www.ecuadorencifras.gob.ec

Reporte de **validaciones de Bases de Datos**

Módulo Ambiental ENESEM 2023

FEBRERO · 2026

INEC | Buenas cifras, mejores vidas

Tabla de Contenido

www.ecuadorencifras.gob.ec

Introducción

Buenas cifras,
mejores vidas

Introducción

Las operaciones estadísticas del Instituto Nacional de Estadística y Censos (INEC) siguen un modelo de producción estándar que sistematiza y estandariza el quehacer estadístico a nivel de la institución, así como de todo el Sistema Estadístico Nacional (SEN). Se trata del **Modelo de Producción Estadística (MPE)**[1], que no es sino:

> "…*el conjunto de fases, procesos y actividades necesarias para producir estadísticas oficiales. A su vez, el modelo permite estandarizar y mejorar los datos y metadatos, y establecer una terminología común en el proceso de producción estadística entre las entidades del Sistema Estadístico Nacional*" (p. 19).

Según las directrices del MPE, existen 8 fases estándar del modelo (en este orden): Planificación, Diseño, Construcción, Recolección, Procesamiento, Análisis, Difusión y Evaluación. Estas fases y sus procesos y actividades serán aplicables a una determinada operación estadística, dependiendo de la naturaleza de la misma.

En el caso particular del Módulo de Información Ambiental Económica en Empresas (ENESEM), edición 2023, todas las fases designadas por el MPE se aplicaron a dicho módulo. En particular, en la fase de Procesamiento, existen 9 subprocesos que configuran esta fase: **(1) Compilar y/o revisar los datos; (2) Integrar los archivos de datos; (3) Clasificar y codificar; (4) Validar los datos; (5) Editar e imputar; (6) Derivar nuevas variables y unidades; (7) Ajustar los factores de expansión; (8) Calcular agregados;** y **(9) Finalizar los archivos de datos**.

[1] INEC, 2024. Modelo de Producción Estadística Versión 2.0. URL: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Sistema_Estadistico_Nacional/Normativas_y_Estandares/modelo_de_produccion_estadistica_v2.pdf; accedido el 04 de febrero de 2026.

El presente reporte brinda información sobre las tareas y resultados obtenidos en los subprocesos ***(4) Validar los datos*** y ***(5) Editar e imputar*** aplicados al Módulo de Información Ambiental Económica de la Encuesta Estructural Empresarial (ENESEM), edición 2023.

www.ecuadorencifras.gob.ec

02.

Fase de Procesamiento del MPE: tipos de validaciones e imputación de la base de datos

Buenas cifras, mejores vidas

En la fase de Procesamiento del Modelo de Producción Estadística,

> "…*se desarrollan procesos y actividades que garantizan el correcto procesamiento de los datos recolectados o recopilados, depuración y generación de la base de datos y su preparación para el análisis y difusión.*" (p. 64).

La fase de Procesamiento:

> "…*está conformada por procesos que compilan, revisan, integran, clasifican, validan, depuran y transforman los datos para que puedan ser analizados y difundidos*". *(p. 64)*.

De toda la fase de Procesamiento del MPE, quizá los procesos de mayor importancia sean los de ***Validar los datos*** y ***Editar e imputar***, debido a que por ellos se garantiza la consistencia y fiabilidad de los datos recabados en la operación estadística. Según el Modelo de Producción Estadística, el proceso de ***Validar los datos***:

> "…*examina los datos para validar la coherencia entre variables, identificar posibles o potenciales inconsistencias, errores o discrepancias como: valores atípicos, respuestas faltantes o errores de codificación. Puede ejecutarse de manera iterativa, validando los datos mediante reglas predefinidas (mallas de validación)*". (p. 66).

Asismismo, el proceso de ***Editar e imputar*** tiene importancia crítica, pues complementa el proceso de ***Validar los datos*** al corregir los errores detectados en este último proceso:

> "*Si bien este proceso [de* **Validar los datos***] se ocupa de la detección y localización de errores reales o potenciales, cualquier actividad de corrección se realiza en el proceso* ***Editar e imputar***". (p.66).

Precisamente, el proceso de ***Editar e imputar*** tiene la finalidad de aplicar métodos correctivos a los datos recolectados,

> "*Cuando los datos se consideran erróneos, faltantes, imprecisos o desactualizados, se pueden asignar o insertar nuevos valores, con el fin de depurarlos y mejorar su precisión y coherencia*". (p. 68).

Con respecto a los métodos que pueden aplicarse en la fase de ***Editar e Imputar*** para corregir datos anómalos que se recolectaron en las fases previas respectivas,

> "*Existe una variedad de métodos para editar e imputar, estos deben ser metodológicamente sólidos y se deben definir en el proceso* ***2.5 Diseñar el procesamiento y análisis****. Los más utilizados son los enfoques basados en reglas, también denominados especificaciones o normas estadísticamente aceptadas, siempre y cuando no se sobrepase el porcentaje óptimo de datos con inconsistencias, ya que, al darse este caso, se debe regresar a la fase de Recolección para realizar las correcciones necesarias*". (p. 68).

En el caso del Módulo de Información Ambiental Económica de la ENESEM 2023, se han implementado no solamente uno, sino cuatro tipos de validaciones con los cuales se minimiza los tipos y eventos de baja calidad de la información recabada en la fase de Recolección (Levantamiento en campo) de la Información. Los tipos de validaciones aplicados al Módulo de Información Ambiental Económica de la ENESEM 2023 son los siguientes:

1. ***Validación estándar (o lógica):*** En este método de validación se valida la información de una variable con referencia a ciertos valores definidos como "válidos" de otras variables relacionadas con la variable a validar. Por ejemplo, si la variable **v7001** debe ser tal que iguale a la variable **v5090**, se crea una variable de control denominada **c7001**, en la cual se marca con "1" a las empresas que no cumplan con la condición lógica: **v7001 == v5090**. En el Anexo A se adjunta el código R que ejecuta este subproceso.
2. ***Control de integridad:*** Por el cual se ejecuta código corrector de valores que no pudieron ser validados en las diferentes rondas de validación, sea porque el aplicativo de ingreso de datos no registró un valor en su formato definido, sea porque no se logró corroborar el dato en campo, sea porque la empresa investigada no quiso entregar el dato, o por cualquier otra razón. El código que realiza el control de integridad ejecuta, efectivamente, una imputación de datos. Sin embargo, esta imputación no aplica valores que pueden resultar de simulaciones o de aproximaciones estadísticas, sino que corresponden a valores finitos y determinados que deberían tener las variables con problemas de falta de integridad. En ese sentido, puede decirse que la imputación es *limitada* a ciertos valores que debería exhibir una variable en condiciones normales. Por

ejemplo, si la regla de validación lógica **c10ii6** determina que debe haber un valor entero entre 0 y 100 para el registro del porcentaje de aguas residuales tratadas (variable **v10ii6**), pero por efecto de algún borrado involuntario del crítico se perdió la información de la variable **v10ii6**, se recupera esta información de otras variables que generan el contexto de validación para la **v10ii6**, como es la variable **v10ii2** ("¿Los procesos productivos de la empresa generaron aguas residuales?"). Así, si la **v10ii2** registra código 2 (= "No"), entonces el porcentaje de aguas residuales tratadas es 0%, valor que debería ir en la **v10ii6**. Además, dado que hay otro grupo de variables cuyos valores dependen del valor de **v10ii6** (como son la secuencia de variables **v10036, …, v10048** que registran el destino de las aguas residuales tratadas; así como la secuencia de variables **v10049, …, v10061** que registran el destino de las aguas residuales no tratadas), se tiene que colocar los valores válidos correspondientes en estas otras variables (cuyo contexto se define parcial o totalmente con la variable bajo escrutinio **v10ii6**) para que todo el subsistema de variables sea coherente. Esta validación suele realizarse, generalmente, después de la última ronda de validaciones, debido a que se dispone de la mayor muestra posible para estimar la distribución teórica de probabilidades de las variables y, por tanto, se disminuye la varianza y se minimiza la cantidad de "falsos positivos" que pudieran generar una carga de trabajo extra innecesaria para el personal de Crítica. En el Anexo B se adjunta el código R que ejecuta este subproceso.

3. ***Validaciones de comparaciones de valores 2023 con valores 2022:*** En este método de validación se suele calcular la variación interanual relativa (2023-2022) para todas las variables escalares del módulo. Luego, se procede a determinar los valores extremos –a veces llamados "atípicos"- de las distribuciones de las antes mencionadas variaciones interanuales relativas. Este método de validación de datos se ejecutó cada dos semanas durante la fase de Recolección de datos, de manera que los valores extremos detectados se enviaron a las diferentes Coordinaciones Zonales para que se revise la causa por la cual se dieron variaciones interanuales muy grandes. El objetivo de esta revisión es corroborar el dato actual –en este caso, recolectado al año 2023-, o bien corregirlo o incluso dejarlo como está, siempre y cuando se brinde una observación de Crítica que incluya una justificación verosímil y verificable por muestreo en una posterior fase de auditoría del trabajo de revisión de las Coordinaciones Zonales, verificación hecha por personal técnico de Planta Central. En el Anexo C se adjunta el código R que ejecuta este subproceso.

A partir del año 2023, durante el cual se realizó el levantamiento de la información de la ENESEM 2022, se aplicó este tipo de validación desde la 2da. ronda de validación hasta la final. Efectivamente, esta mejora implementada para la ENESEM 2022 se deriva de un estudio estadístico empírico profundo de las distribuciones de las variaciones interanuales 2020-2021 y 2021-2022 de las principales variables de escala del Módulo de Información Económica Ambiental de la ENESEM 2022. Aplicando técnicas de aprendizaje automático de máquina, se logró establecer umbrales de variación interanual máxima para dichas variables por cada uno de los tamaños de empresas, de manera que se realizó este tipo de control desde la 2da. ronda de validación, como se manifestó anteriormente.

4. ***Validaciones de verificación de valores extremos:*** Usualmente, las distribuciones de las variables de escala (crematísticas o de cantidad) suelen generar valores atípicos y extremos, tanto inferiores como superiores. Lo que se hace con este tipo de validación es fijar una variable de escala, y luego marcar a las empresas cuyos valores son inferiores al umbral inferior (límite válido inferior), así como a las que tiene valores superiores al umbral superior (límite válido superior). Estos casos tienen que verificarse por parte de los críticos de las zonales quienes, de ser necesario, corregirán los valores erróneos o ratificarán los valores ingresados con la correspondiente justificación. En el Anexo D se adjunta el código R que ejecuta este subproceso.

   Esta validación suele realizarse, generalmente, en la última ronda de validaciones, debido a que se dispone de la mayor muestra posible para estimar la distribución teórica de probabilidades de las variables y, por tanto, se disminuye la varianza y se minimiza la cantidad de "falsos positivos" que pudieran generar una carga de trabajo extra innecesaria para el personal de Crítica.

Una vez descritos en detalle los diferentes métodos o tipos de validaciones aplicados a los datos recolectados para las variables del Módulo de Información Ambiental Económica de la ENESEM 2023, se procede a reportar los resultados encontrados para estas validaciones en los diferentes cortes de la base de datos, correspondientes a los cuatro tipos de validación efectuadas.

www.ecuadorencifras.gob.ec

03.

Reporte de resultados de validaciones por ronda y coordinación zonal

Buenas cifras, mejores vidas

Los resultados de las validaciones lógicas, según los 21 cortes de las bases de datos correspondientes a las rondas de validación planificadas, dada una muestra inicial total de 4860 empresas, fueron los siguientes:

1. **Corte #1 al 16-07-2024:** De un universo de 104 empresas efectivas y criticadas, 104 empresas tuvieron al menos un error de validación lógica. Estas 104 empresas se distribuyen por zonal como: 23=Zonal Centro; 25=Zonal DICA; 37=Zonal Litoral; 19=Zonal Sur. Entre todas las zonales, se generaron 110 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control[2].
2. **Corte #2 al 22-07-2024:** De un universo de 331 empresas efectivas y criticadas, 44 empresas nuevas[3] tuvieron al menos un error de validación lógica. Estas 44 empresas se distribuyen por zonal como: 2=Zonal Centro; 22=Zonal DICA; 12=Zonal Litoral; 8=Zonal Sur. Entre todas las zonales, se generaron 28 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
3. **Corte #3 al 29-07-2024:** De un universo de 482 empresas efectivas y criticadas, 35 empresas nuevas tuvieron al menos un error de validación lógica. Estas 35 empresas se distribuyen por zonal como: 1=Zonal Centro; 13=Zonal DICA; 13=Zonal Litoral; 8=Zonal Sur. Entre todas las zonales, se generaron 23 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
4. **Corte #4 al 05-08-2024:** De un universo de 625 empresas efectivas y criticadas, 72 empresas nuevas tuvieron al menos un error de validación lógica. Estas 72 empresas se distribuyen por zonal como: 4=Zonal Centro; 11=Zonal DICA; 52=Zonal Litoral; 5=Zonal Sur. Entre todas las zonales, se generaron 59 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
5. **Corte #5 al 12-08-2024:** De un universo de 792 empresas efectivas y criticadas, 46 empresas nuevas tuvieron al menos un error de validación lógica. Estas 46 empresas se distribuyen por zonal como: 1=Zonal Centro; 11=Zonal DICA;

[2] Correspondientes a 957 reglas de validación lógica de toda la malla de validación lógica.

[3] Estas empresas sí incluyen a algunas de las existentes en el 1er corte. Esta situación se mantendrá a lo largo de todas las rondas de validación. Las empresas que no fueron justificadas en la anterior ronda de validación o que no fueron gestionadas aparecerán en la siguiente ronda de validación, y así hasta que el error haya sido solventado de forma válida. En general, en cada ronda de validación permanecieron como empresas con "errores persistentes entre rondas" entre el 1% y el 3% de las empresas a ser validadas.

18=Zonal Litoral; 16=Zonal Sur. Entre todas las zonales, se generaron 28 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.

6. **Corte #6 al 19-08-2024:** De un universo de 978 empresas efectivas y criticadas, 37 empresas nuevas tuvieron al menos un error de validación lógica. Estas 37 empresas se distribuyen por zonal como: 2=Zonal Centro; 10=Zonal DICA; 14=Zonal Litoral; 11=Zonal Sur. Entre todas las zonales, se generaron 28 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
7. **Corte #7 al 26-08-2024:** De un universo de 1162 empresas efectivas y criticadas, 1+19+9+19 empresas nuevas tuvieron al menos un error de validación lógica. Estas 1+19+9+19 empresas se distribuyen por zonal como: 1=Zonal Centro; 19=Zonal DICA; 9=Zonal Litoral; 19=Zonal Sur. Entre todas las zonales, se generaron 1+10+6+14 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
8. **Corte #8 al 02-09-2024:** De un universo de 1339 empresas efectivas y criticadas, 59 empresas nuevas tuvieron al menos un error de validación lógica. Estas 59 empresas se distribuyen por zonal como: 5=Zonal Centro; 22=Zonal DICA; 11=Zonal Litoral; 21=Zonal Sur. Entre todas las zonales, se generaron 40 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
9. **Corte #9 al 09-09-2024:** De un universo de 1473 empresas efectivas y criticadas, 44 empresas nuevas tuvieron al menos un error de validación lógica. Estas 44 empresas se distribuyen por zonal como: 1=Zonal Centro; 10=Zonal DICA; 13=Zonal Litoral; 20=Zonal Sur. Entre todas las zonales, se generaron 30 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
10. **Corte #10 al 16-09-2024:** De un universo de 963 empresas efectivas y criticadas, 42 empresas nuevas tuvieron al menos un error de validación lógica. Estas 42 empresas se distribuyen por zonal como: 1=Zonal Centro; 0=Zonal DICA; 24=Zonal Litoral; 17=Zonal Sur. Entre todas las zonales, se generaron 29 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
11. **Corte #11 al 23-09-2024:** De un universo de 1561 empresas efectivas y criticadas, 42 empresas nuevas tuvieron al menos un error de validación lógica. Estas 42 empresas se distribuyen por zonal como: 2=Zonal Centro; 9=Zonal DICA; 12=Zonal

Litoral; 19=Zonal Sur. Entre todas las zonales, se generaron 30 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.

12. **Corte #12 al 30-09-2024:** De un universo de 1842 empresas efectivas y criticadas, 38 empresas nuevas tuvieron al menos un error de validación lógica. Estas 38 empresas se distribuyen por zonal como: 0=Zonal Centro; 10=Zonal DICA; 15=Zonal Litoral; 13=Zonal Sur. Entre todas las zonales, se generaron 27 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
13. **Corte #13 al 07-10-2024:** De un universo de 1954 empresas efectivas y criticadas, 47 empresas nuevas tuvieron al menos un error de validación lógica. Estas 47 empresas se distribuyen por zonal como: 0=Zonal Centro; 32=Zonal DICA; 6=Zonal Litoral; 9=Zonal Sur. Entre todas las zonales, se generaron 33 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
14. **Corte #14 al 14-10-2024:** De un universo de 2056 empresas efectivas y criticadas, 39 empresas nuevas tuvieron al menos un error de validación lógica. Estas 39 empresas se distribuyen por zonal como: 1=Zonal Centro; 14=Zonal DICA; 8=Zonal Litoral; 16=Zonal Sur. Entre todas las zonales, se generaron 37 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
15. **Corte #15 al 21-10-2024:** De un universo de 2199 empresas efectivas y criticadas, 43 empresas nuevas tuvieron al menos un error de validación lógica. Estas 43 empresas se distribuyen por zonal como: 0=Zonal Centro; 16=Zonal DICA; 18=Zonal Litoral; 9=Zonal Sur. Entre todas las zonales, se generaron 32 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
16. **Corte #16 al 28-10-2024:** De un universo de 2295 empresas efectivas y criticadas, 31 empresas nuevas tuvieron al menos un error de validación lógica. Estas 31 empresas se distribuyen por zonal como: 1=Zonal Centro; 8=Zonal DICA; 17=Zonal Litoral; 5=Zonal Sur. Entre todas las zonales, se generaron 28 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
17. **Corte #17 al 04-11-2024:** De un universo de 2389 empresas efectivas y criticadas, 31 empresas nuevas tuvieron al menos un error de validación lógica. Estas 31 empresas se distribuyen por zonal como: 5=Zonal Centro; 12=Zonal DICA; 8=Zonal Litoral; 6=Zonal Sur. Entre todas las zonales, se generaron 26 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
18. **Corte #18 al 11-11-2024:** De un universo de 2466 empresas efectivas y criticadas, 21 empresas nuevas tuvieron al menos un error de validación lógica. Estas 21

empresas se distribuyen por zonal como: 0=Zonal Centro; 5=Zonal DICA; 11=Zonal Litoral; 5=Zonal Sur. Entre todas las zonales, se generaron 21 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.

19. **Corte #19 al 18-11-2024:** De un universo de 2590 empresas efectivas y criticadas, 39 empresas nuevas tuvieron al menos un error de validación lógica. Estas 39 empresas se distribuyen por zonal como: 2=Zonal Centro; 11=Zonal DICA; 22=Zonal Litoral; 4=Zonal Sur. Entre todas las zonales, se generaron 31 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
20. **Corte #20 al 25-11-2024:** De un universo de 2706 empresas efectivas y criticadas, 82 empresas nuevas tuvieron al menos un error de validación lógica. Estas 82 empresas se distribuyen por zonal como: 3=Zonal Centro; 30=Zonal DICA; 35=Zonal Litoral; 14=Zonal Sur. Entre todas las zonales, se generaron 90 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.
21. **Corte #21 al 02-12-2024:** De un universo de 2180 empresas efectivas y criticadas, 12 empresas nuevas tuvieron al menos un error de validación lógica. Estas 12 empresas se distribuyen por zonal como: 0=Zonal Centro; 2=Zonal DICA; 10=Zonal Litoral; 0=Zonal Sur. Entre todas las zonales, se generaron 10 variables de control con al menos un error a verificar, de un total teórico de 957 variables de control.

De las 4860 empresas que conformaron la muestra ENESEM 2023 previa a la fase de levantamiento de información, permanecieron 4435 empresas en la BDD de publicación. Las 425 empresas que salieron de las empresas de publicación tuvieron novedades, como haber sido empresas fusionadas, absorbidas o desintegradas, además de empresas que no entregaron la suficiente información económica y/o ambiental para la ENESEM 2023.

Con respecto a la validación de integridad, cabe señalar que se realizaron imputaciones de ciertos datos en algunas de las variables ambientales de la ENESEM 2023. Todos estos cambios no se realizaron por aplicación de métodos de estimación o de aproximación de datos faltantes o erróneos. El método de imputación usado para los datos ambientales faltantes o erróneos consistió en contactar con los informantes de c/u de las empresas que tenían variables con estas novedades, quienes brindaron la información correcta asociada a valores faltantes o erróneos detectados en el momento de la generación de la BDD

preliminar. A continuación se muestra, variable por variable, el recuento de cambios que se realizaron a los datos faltantes o erróneos del Módulo de Información Económico Ambiental de la ENESEM 2023:

Nombre de variable	v7006	v8096	v8099	v9i2	v9006	v9020	v9029	v9033	v9036
Recuento de cambios realizados	1	4	13	1	2	1	1	1	21

Nombre de variable	v9037	v9038	v9039	v9044	v9052	v9053	v9054	v9055	v9ii1
Recuento de cambios realizados	21	21	21	22	1	1	1	1	3

Nombre de variable	v9078	v9079	v9105	v10i2	v10004	v10i3	v10006	v10009
Recuento de cambios realizados	8	2	2	10	1	1	4	4

Nombre de variable	v10010	v10011	v10012	v10016	v10017	v10019	v10020	v10022
Recuento de cambios realizados	1	2	4	2	3	1	3	4

Nombre de variable	v10030	v10ii3	v10032	v10035	v10127	v10128	v10379	v10380
Recuento de cambios realizados	8	3	14	14	2	20	2	3

Nombre de variable	tipo_desecho_2	v10590	tipo_desecho_9	tipo_desecho_10	tipo_desecho_11	tipo_desecho_12
Recuento de cambios realizados	1	2	1	2	1	1

Nombre de variable	tipo_desecho_13	tipo_desecho_14	tipo_desecho_15	tipo_desecho_16	tipo_desecho_17
Recuento de cambios realizados	1	1	1	2	1

Nombre de variable	tipo_desecho_18	tipo_desecho_19	tipo_desecho_20	tipo_desecho_21	tipo_desecho_22
Recuento de cambios realizados	2	2	2	2	3

TABLA 1. Registro de cambios realizados a datos de la BDD de prepublicación.
Fuente: Módulo de Información Económica Ambiental de la ENESEM 2023.

Se realizaron 279 cambios de información puntuales a 57 variables del Módulo de Información Económica Ambiental de la ENESEM 2023. El código R que realiza esta tarea aparece en el Anexo B del presente documento.

Para las validaciones de comparaciones de valores de las variables ambientales entre los años 2022 y 2023, se controlaron 44 variables del Módulo Ambiental. Los resultados de este tipo de validaciones por comparación, según los cortes 1ro. hasta 10mo. de las bases de datos correspondientes a las rondas de validación planificadas, fueron las siguientes:

1. **Corte #1 al 29-07-2024:** De un universo de 366 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 228 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido[4]. Estas 228 empresas se distribuyen por zonal como: 16=Zonal

[4] Este umbral se había fijado en 30% para las variables de escala del Módulo Ambiental de la ENESEM 2022, después de hacer un análisis descriptivo de las distribuciones de las variaciones interanuales 2021-2022 de todas sus variables escalares. Sin embargo, para el levantamiento de la ENESEM 2023 se realizó una modificación que devino en una innovación técnica, por la cual ya no se fijaba un umbral máximo fijo del 30% de variación interanual para las variables escalares, sino que se tomó al 2% de los valores más altos de las distribuciones de las variaciones interanuales 2022-2023. Esos valores se los envío a revisión a las zonales. La decisión de escoger este valor de corte del 2% se lo debió porque en casi todas las distribuciones de variaciones interanuales 2022-2023 resultaba que el 98% de todos los valores inferiores de las distribuciones generaban una distribución log-normal que pasaban las pruebas estándar de normalidad, como el test de Kolmogorov-Smirnov y el test de Shapiro-Wilk. Cuando se incluía en las distribuciones ese 2% de datos superiores adicionales, casi nunca se pasaban las pruebas de normalidad

Centro; 79=Zonal DICA; 99=Zonal Litoral; 34=Zonal Sur. En total, se generaron 638 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 27=Zonal Centro; 267=Zonal DICA; 264=Zonal Litoral; 80=Zonal Sur.

2. **Corte #2 al 14-08-2024:** De un universo de 669 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 363 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 363 empresas se distribuyen por zonal como: 18=Zonal Centro; 176=Zonal DICA; 122=Zonal Litoral; 47=Zonal Sur. En total, se generaron 1384 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 45=Zonal Centro; 827=Zonal DICA; 400=Zonal Litoral; 112=Zonal Sur.
3. **Corte #3 al 26-08-2024:** De un universo de 872 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 468 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 468 empresas se distribuyen por zonal como: 34=Zonal Centro; 209=Zonal DICA; 170=Zonal Litoral; 55=Zonal Sur. En total, se generaron 1420 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 70=Zonal Centro; 722=Zonal DICA; 500=Zonal Litoral; 128=Zonal Sur.
4. **Corte #4 al 09-09-2024:** De un universo de 1426 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 447 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 447 empresas se distribuyen por zonal como: 32=Zonal Centro; 231=Zonal DICA; 144=Zonal Litoral; 40=Zonal Sur. En total, se generaron 1400 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 68=Zonal Centro; 835=Zonal DICA; 392=Zonal Litoral; 105=Zonal Sur.
5. **Corte #5 al 25-09-2024:** De un universo de 1507 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 482 empresas tuvieron al menos una variable con una variación interanual mayor al

---

para las distribuciones log-normales mencionadas. Este hecho puede interpretarse como que este 2% de valores superiores debían ser valores extremos (superiores incluso que los valores atípicos) de las distribuciones de comparaciones 2022-2023, los cuales ameritaban revisarse en las respectivas zonales. Como efecto del diseño y aplicación de este procedimiento estadístico de detección de valores extremos superiores, en las diferentes variables de escala se generaron varios umbrales máximos de variación interanual permitidos. Este procedimiento se mantuvo con pequeñas variaciones para la ENESEM 2023.

umbral establecido. Estas 482 empresas se distribuyen por zonal como: 5=Zonal Centro; 299=Zonal DICA; 138=Zonal Litoral; 40=Zonal Sur. En total, se generaron 1535 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 9=Zonal Centro; 872=Zonal DICA; 514=Zonal Litoral; 140=Zonal Sur.

6. **Corte #6 al 07-10-2024:** De un universo de 2061 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 732 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 732 empresas se distribuyen por zonal como: 5=Zonal Centro; 352=Zonal DICA; 84=Zonal Litoral; 291=Zonal Sur. En total, se generaron 2665 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 8=Zonal Centro; 1366=Zonal DICA; 302=Zonal Litoral; 989=Zonal Sur.
7. **Corte #7 al 21-10-2024:** De un universo de 2419 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 449 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 449 empresas se distribuyen por zonal como: 8=Zonal Centro; 325=Zonal DICA; 75=Zonal Litoral; 41=Zonal Sur. En total, se generaron 1453 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 25=Zonal Centro; 1008=Zonal DICA; 270=Zonal Litoral; 150=Zonal Sur.
8. **Corte #8 al 07-11-2024:** De un universo de 2719 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 929 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 929 empresas se distribuyen por zonal como: 10=Zonal Centro; 779=Zonal DICA; 116=Zonal Litoral; 24=Zonal Sur. En total, se generaron 2880 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 27=Zonal Centro; 2313=Zonal DICA; 465=Zonal Litoral; 75=Zonal Sur.
9. **Corte #9 al 20-11-2024:** De un universo de 2959 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 541 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 541 empresas se distribuyen por zonal como: 0=Zonal Centro; 402=Zonal DICA; 123=Zonal Litoral; 16=Zonal Sur. En total, se generaron 2235 novedades a verificar para las 44 variables de control establecidas,

distribuidas por zonal como: 0=Zonal Centro; 1721=Zonal DICA; 469=Zonal Litoral; 45=Zonal Sur.

10. **Corte #10 al 03-12-2024:** De un universo de 3242 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2023 y 2022 (publicada), 525 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 525 empresas se distribuyen por zonal como: 0=Zonal Centro; 222=Zonal DICA; 298=Zonal Litoral; 5=Zonal Sur. En total, se generaron 2051 novedades a verificar para las 44 variables de control establecidas, distribuidas por zonal como: 0=Zonal Centro; 912=Zonal DICA; 1115=Zonal Litoral; 24=Zonal Sur.

Finalmente, con respecto a la verificación de valores extremos efectuada sobre el corte de BDD al 16 de diciembre de 2024, se obtuvo los siguientes resultados:

- Se tomó como universo a las 4440 empresas[5] validadas y criticadas hasta el corte de BDD mencionado. El universo de variables a controlar abarca a 49 variables escalares[6] como ingresos, gastos, cantidad de energía, cantidad de residuos generada, etc. Se detectaron 747 empresas con al menos una de las 49 variables a controlar con al menos un valor extremo (sea supremo o ínfimo) de las distribuciones de valores de estas variables, segmentadas por el tamaño de empresa[7]. Se obtuvieron los siguientes resultados por zonal:

    - Zonal Centro: Hubo 27 empresas de 224 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 58 supremos, 1 ínfimo y 59 extremos en total. Se detectaron valores extremos en 43 de las 49 variables escalares a validar.

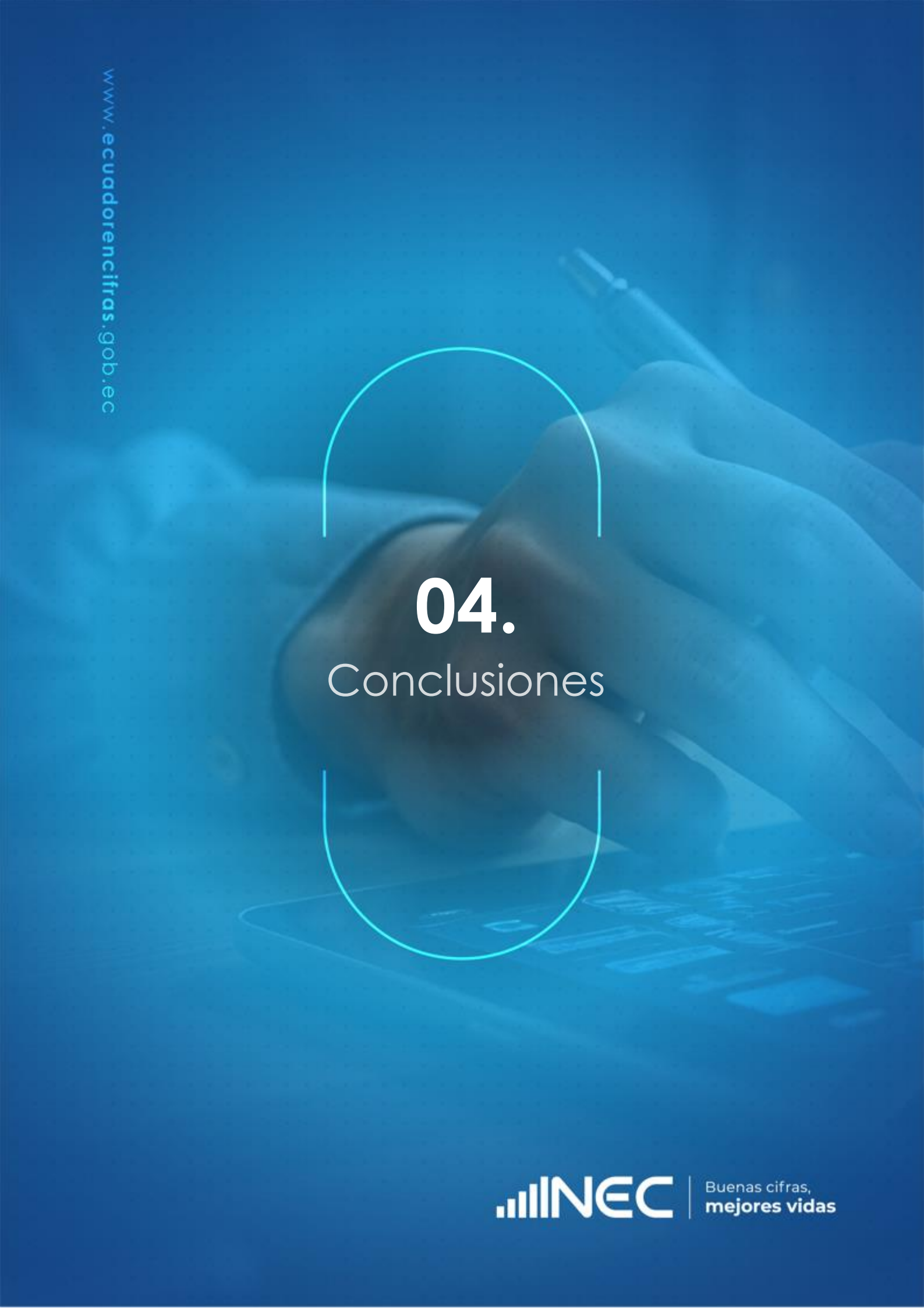
[5] No obstante, se aplicó el algoritmo de valores extremos a las 4435 empresas de publicación de la BDD del Módulo Ambiental de la ENESEM 2023.

[6] Para la ENESEM 2020 se corrió este tipo de validación sobre 681 variables escalares. Este numerario se redujo a apenas 23 variables escalares para la ENESEM 2021, y a 42 variables para la ENESEM 2022, una vez que se determinó el impacto individual del aporte de c/u de las 42 variables originales sobre el Impacto Ambiental Agregado 2022. Para la ENESEM 2023 se incrementaron 7 variables adicionales de importancia.

[7] Esto implica que, por ejemplo, para la variable ***v7001*** no existe únicamente una distribución de los valores logaritmados de esa variable, sino tres: uno para c/u de los tamaños de empresa (Mediana A, Mediana B y Grande). Si una empresa sobrepasa los umbrales por tamaño, determinados por el algoritmo de la función del lenguaje R **robustbase::adjbox()**, entonces se la marca con el código "SUP" para indicar que el valor extremo detectado es un supremo de la distribución logaritmada. Si se lo marca con el código "INF", entonces el valor extremo detectado es un ínfimo de la distribución logaritmada.

- Zonal Litoral: Hubo 336 empresas de 1968 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 667 supremos, 73 ínfimos y 740 extremos en total. Se detectaron valores extremos en 49 de las 49 variables escalares a validar.
- Zonal DICA: Hubo 309 empresas de 1711 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 255 supremos, 62 ínfimos y 317 extremos en total. Se detectaron valores extremos en 49 de las 49 variables escalares a validar.
- Zonal Sur: Hubo 75 empresas de 537 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 55 supremos, 24 ínfimos y 79 extremos en total. Se detectaron valores extremos en 46 de las 49 variables escalares a validar.

Cabe mencionar que todas estas 747 empresas de las 4440 con potencial de ser publicables fueron justificadas en su totalidad sobre sus valores extremos. Aproximadamente el 1% de valores a verificar fueron corregidos, sea porque fueron demasiado bajos, sea porque fueron demasiado altos con respecto a los valores de las empresas similares por tamaño. El resto fueron corroborados por crítica y/o campo.



04.
Conclusiones

Como se ha mencionado anteriormente, los procesos ***Validar los datos*** y ***Editar e imputar*** de la fase de Procesamiento del Modelo de Producción Estadística son cruciales para el aseguramiento de la calidad de la información recolectada en campo para una operación estadística.

En el caso del Módulo de Información Ambiental Económica de la ENESEM 2023, estos procesos se han implementado aplicando diferentes métodos, cada uno de las cuales minimiza la cantidad de errores de levantamiento, garantizando así la mejor calidad de la información disponible para los usuarios de esta operación estadística.

La implementación continua de los distintos tipos de validación que se utilizan para validar el Módulo de Información Ambiental Económica de la ENESEM 2023 presenta un alto grado de innovación, que se manifiesta a través de la automatización casi total de las tareas y actividades. Para este fin, se ha utilizado la plataforma de programación RStudio 2023.12.1, en la cual se han desarrollado varios archivos o módulos de código R en su versión "R Markdown", la cual es ejecutable por bloques de código según la necesidad del programador.

Los resultados obtenidos confirman la tendencia creciente hacia la mejora continua de la calidad de la información generada en la presente operación estadística. De esta manera, el INEC cumple con su misión de garantizar a los usuarios especializados y al público en general el acceso a información robusta, oportuna y accesible universalmente para apoyar al desarrollo efectivo de los sectores productivos del país.

www.ecuadorencifras.gob.ec

05.

Anexos

Buenas cifras,
mejores vidas

ANEXO A. Código R Script para la validación estándar o lógica.

```
#-------------------------------------------------------------------------------------
# title: "Automatización homologada de la validación estándar del Módulo de Información
#         Económica Ambiental (MIEA) de la ENESEM 2023"
# author: "Ramiro Benavides"
# date: "2024-02-29"
# update: "2024-05-28"
#-------------------------------------------------------------------------------------

setwd("C:/VALIDACION_ENESEM")

##############################
# [BLOQUE 1]. CARGA DE DATOS.
##############################

B <- data # Copia del dataframe de datos a validar desde el entorno global.

absorbida <- as.character(data$absorbida)
desintegracion <- as.character(data$desintegracion)
escision <- as.character(data$escision)

absorbida[sapply(absorbida, is.na)] <- ""
desintegracion[sapply(desintegracion, is.na)] <- ""
escision[sapply(escision, is.na)] <- ""

# Las siguientes variables (UNO y CIEN) se utilizarán para las validaciones de sumas de porcentajes.
B$UNO <- 1
B$CIEN <- 100

###############################################
# [BLOQUE 2]. CARGA DE FUNCIONES UTILITARIAS.
###############################################

# La función CalcDiffCol() calcula diferencias de columnas.
CalcDiffCol <- function(colbase, col1, ...) {
  y <- apply(cbind(col1, ...), 1, sum, na.rm = T)
  z <- apply(cbind(-y, colbase), 1, sum, na.rm = T)
  if (sum(z, na.rm = T) != 0) x <- ifelse(z != 0, z, NA_real_)
}

# La función AddRule() agrega una columna de validación de comprobación de sumas
# al frame de datos a validar.
AddRule <- function(label, regla, filtro="") {
  comando1 <- paste0("M <- dim(B)[1]")
  comando2 <- paste0("M <- length(which(", filtro, "))")
  comando <- ifelse(filtro == "", comando1, comando2)
  eval(parse(text = comando))

  comando1 <- paste0("colerr <- ifelse(!(", regla, "), 1, NA)")
  comando2 <- paste0("colerr <- ifelse(!(", regla, ") & (", filtro, "), 1, NA)")
  comando <- ifelse(filtro == "", comando1, comando2)
  eval(parse(text = comando))

  N <- sum(colerr, na.rm = T)
  r <- G

  if (N > 0) {
    comando <- paste0("r$", label, " <- colerr")
    eval(parse(text = comando))
    print(paste0("SÍ se agregó la regla ", label, " al frame de validación. ", "N/M=", N, "/", M))
  } else {
    print(paste0("NO se agregó la regla ", label, " al frame de validación"))
  }
  return(r)
}
```

```
# La función AddValidSumas() agrega una columna de validación al frame de datos a validar.
AddValidSumas <- function(label, colbase, col1, ..., filtro="") {
  r <- G
  p <- CalcDiffCol(colbase, col1, ...)
  p <- as.numeric(p)

  if (abs(sum(p, na.rm=T)) > 1e-4) {
    comando1 <- paste0("M <- dim(B)[1]")
    comando2 <- paste0("M <- length(which(", filtro, "))")
    comando <- ifelse(filtro == "", comando1, comando2)
    eval(parse(text = comando))

    comando1 <- paste0("colerr <- ifelse(abs(p) > 0, 1, NA)")
    comando2 <- paste0("colerr <- ifelse(", filtro, ", ifelse(abs(p) > 0, 1, NA), NA)")
    comando <- ifelse(filtro == "", comando1, comando2)
    eval(parse(text = comando))

    N <- sum(colerr, na.rm = T)

    if (N > 0) {
      comando <- paste0("r$", label, " <- colerr")
      eval(parse(text = comando))

      print(paste0("SÍ se agregó la regla ", label, " al frame de validación. ", "N/M=", N, "/", M))
    } else {
      print(paste0("NO se agregó la regla ", label, " al frame de validación"))
    }
    return(r)
  } else {
    print(paste0("NO se agregó la regla ", label, " al frame de validación"))
  }
  return(r)
}

###############################################################################################
# [BLOQUE 3]. CREACIÓN DEL FRAME 'G' CONTENEDOR DE LAS VARIABLES DE ERROR.
# APLICACIÓN DE TRANSFORMACIONES NATURALES DE TIPO DE DATOS A VARIABLES AMBIENTALES.
###############################################################################################
G <- data.frame(B$inec_identificador_empresa) # G es el dataframe de reporte de errores de validación.
names(G) <- "inec_identificador_empresa"

# Detección del rango de índices donde se encuentran las variables ambientales.
var_inicio <- which(names(B) == "v7001")
var_final <- which(names(B) == "v10iii42")
vars_AMB <- names(B)[var_inicio:var_final]
B[, vars_AMB] <- type.convert(B[, vars_AMB], as.is = TRUE)
rm(var_inicio, var_final)

#############################################################
# [BLOQUE 4]. EJECUCION DE LAS REGLAS DE VALIDACION DEL MIEA.
#############################################################

############################
# A. VALIDACIONES DE SUMAS
# 1. Validación de sumas: Capítulo 7
G <- AddValidSumas("e1001", B$v7002, B$v7003, B$v7004)

# 2. Validación de sumas: Capítulo 8
G <- AddValidSumas("e1011", B$v8098, B$v8087, B$v8093)
G <- AddValidSumas("e1012", B$v8099, B$v8089, B$v8095)
G <- AddValidSumas("e1013", B$v8100, B$v8091, B$v8097)

# 3. Validación de sumas: Capítulo 9
G <- AddValidSumas("e1014", B$v9052, B$v9005, B$v9013, B$v9021, B$v9029, B$v9037,
                   B$v9045, filtro="B$v9i2==1")
G <- AddValidSumas("e1015", B$v9053, B$v9006, B$v9014, B$v9022, B$v9030, B$v9038,
                   B$v9046, filtro="B$v9i2==1")
```

```
G <- AddValidSumas("e1016", B$v9054, B$v9007, B$v9015, B$v9023, B$v9031, B$v9039,
            B$v9047, filtro="B$v9i2==1")
G <- AddValidSumas("e1017", B$v9055, B$v9009, B$v9017, B$v9025, B$v9033, B$v9041,
            B$v9049, filtro="B$v9i2==1")
G <- AddValidSumas("e1018", B$v9056, B$v9010, B$v9018, B$v9026, B$v9034, B$v9042,
            B$v9050, filtro="B$v9i2==1")
G <- AddValidSumas("e1019", B$v9105, B$v9059, B$v9063, B$v9067, B$v9071, B$v9075, B$v9079,
            B$v9083, B$v9087, B$v9091, B$v9095, B$v9099, B$v9102, filtro="B$v9ii1==1")

# 4. Validación de sumas: Capítulo 10. Aguas de captación y residuales.
G <- AddValidSumas("e1020", B$v10030, B$v10012, B$v10020, B$v10028, filtro="B$v10i3==1")
G <- AddValidSumas("e1021", B$v10031, B$v10013, B$v10021, B$v10029, filtro="B$v10i3==1")

# 5. Validación de sumas: Capítulo 10.1 Residuos No Peligrosos.
G <- AddValidSumas("e1039", B$CIEN, B$v10079, B$v10080, B$v10081, filtro="B$v10073 >= 0")
G <- AddValidSumas("e1040", B$CIEN, B$v10100, B$v10101, B$v10102, filtro="B$v10094 >= 0")
G <- AddValidSumas("e1041", B$CIEN, B$v10121, B$v10122, B$v10123, filtro="B$v10115 >= 0")
G <- AddValidSumas("e1042", B$CIEN, B$v10142, B$v10143, B$v10144, filtro="B$v10136 >= 0")
G <- AddValidSumas("e1043", B$CIEN, B$v10163, B$v10164, B$v10165, filtro="B$v10157 >= 0")
G <- AddValidSumas("e1044", B$CIEN, B$v10184, B$v10185, B$v10186, filtro="B$v10178 >= 0")
G <- AddValidSumas("e1045", B$CIEN, B$v10205, B$v10206, B$v10207, filtro="B$v10199 >= 0")
G <- AddValidSumas("e1046", B$CIEN, B$v10226, B$v10227, B$v10228, filtro="B$v10220 >= 0")
G <- AddValidSumas("e1047", B$CIEN, B$v10247, B$v10248, B$v10249, filtro="B$v10241 >= 0")
G <- AddValidSumas("e1048", B$CIEN, B$v10268, B$v10269, B$v10270, filtro="B$v10262 >= 0")
G <- AddValidSumas("e1049", B$CIEN, B$v10289, B$v10290, B$v10291, filtro="B$v10283 >= 0")
G <- AddValidSumas("e1051", B$CIEN, B$v10331, B$v10332, B$v10333, filtro="B$v10325 >= 0")

# 6. Validación de sumas: Capítulo 10.2 Desechos especiales.
G <- AddValidSumas("e1062", B$CIEN, B$v10394, B$v10395, B$v10396, filtro="B$v10388 >= 0")
G <- AddValidSumas("e1063", B$CIEN, B$v10415, B$v10416, B$v10417, filtro="B$v10409 >= 0")
G <- AddValidSumas("e1064", B$CIEN, B$v10436, B$v10437, B$v10438, filtro="B$v10430 >= 0")
G <- AddValidSumas("e1065", B$CIEN, B$v10457, B$v10458, B$v10459, filtro="B$v10451 >= 0")
G <- AddValidSumas("e1066", B$CIEN, B$v10478, B$v10479, B$v10480, filtro="B$v10472 >= 0")
G <- AddValidSumas("e1069", B$CIEN, B$v10541, B$v10542, B$v10543, filtro="B$v10535 >= 0")

# 7. Validación de sumas: Capítulo 10.3 Desechos peligrosos.
G <- AddValidSumas("e1092", B$CIEN, B$v10562, B$v10563, B$v10564, filtro="B$v10556 >= 0")
G <- AddValidSumas("e1093", B$CIEN, B$v10583, B$v10584, B$v10585, filtro="B$v10577 >= 0")
G <- AddValidSumas("e1094", B$CIEN, B$v10604, B$v10605, B$v10606, filtro="B$v10598 >= 0")
G <- AddValidSumas("e1095", B$CIEN, B$v10625, B$v10626, B$v10627, filtro="B$v10619 >= 0")
G <- AddValidSumas("e1096", B$CIEN, B$v10646, B$v10647, B$v10648, filtro="B$v10640 >= 0")
G <- AddValidSumas("e1097", B$CIEN, B$v10667, B$v10668, B$v10669, filtro="B$v10661 >= 0")
G <- AddValidSumas("e1098", B$CIEN, B$v10688, B$v10689, B$v10690, filtro="B$v10682 >= 0")
G <- AddValidSumas("e1099", B$CIEN, B$v10709, B$v10710, B$v10711, filtro="B$v10703 >= 0")
G <- AddValidSumas("e1100", B$CIEN, B$v10730, B$v10731, B$v10732, filtro="B$v10724 >= 0")
G <- AddValidSumas("e1101", B$CIEN, B$v10751, B$v10752, B$v10753, filtro="B$v10745 >= 0")
G <- AddValidSumas("e1102", B$CIEN, B$v10772, B$v10773, B$v10774, filtro="B$v10766 >= 0")
G <- AddValidSumas("e1103", B$CIEN, B$v10793, B$v10794, B$v10795, filtro="B$v10787 >= 0")
G <- AddValidSumas("e1104", B$CIEN, B$v10814, B$v10815, B$v10816, filtro="B$v10808 >= 0")
G <- AddValidSumas("e1105", B$CIEN, B$v10835, B$v10836, B$v10837, filtro="B$v10829 >= 0")
G <- AddValidSumas("e1106", B$CIEN, B$v10856, B$v10857, B$v10858, filtro="B$v10850 >= 0")
G <- AddValidSumas("e1107", B$CIEN, B$v10877, B$v10878, B$v10879, filtro="B$v10871 >= 0")
G <- AddValidSumas("e1108", B$CIEN, B$v10898, B$v10899, B$v10900, filtro="B$v10892 >= 0")
G <- AddValidSumas("e1109", B$CIEN, B$v10919, B$v10920, B$v10921, filtro="B$v10913 >= 0")
G <- AddValidSumas("e1110", B$CIEN, B$v10940, B$v10941, B$v10942, filtro="B$v10934 >= 0")
G <- AddValidSumas("e1111", B$CIEN, B$v10961, B$v10962, B$v10963, filtro="B$v10955 >= 0")
G <- AddValidSumas("e1112", B$CIEN, B$v10982, B$v10983, B$v10984, filtro="B$v10976 >= 0")
G <- AddValidSumas("e1113", B$CIEN, B$v11003, B$v11004, B$v11005, filtro="B$v10997 >= 0")

##################################################
# B. VALIDACIONES LOGICAS, DE CRUCE O RELACIONALES.

################################
# Capítulo 7. GESTION AMBIENTAL
################################

#-------------------------
```

```
# REGLA val_7_001
#-------------------------
# En la casilla 7001, el aplicativo debe recuperar y desplegar el número de personas ocupadas que se registra en la Línea 238, variable 5090.

G <- AddRule("e1401", "B$v7001==B$v5090")

#-------------------------
# REGLA val_7_002
#-------------------------
# Si contesta "Sí" en la Pregunta 1, pase a la Pregunta 2. Si contesta "No", pasar al Capítulo 8.

G <- AddRule("e1402", "B$v71 %in% 1:2")

#-------------------------
# REGLA val_7_003
#-------------------------
# Obligatorio si respondió que "Sí" en la Pregunta 1. Validar si este valor es mayor que 0 y menor que VAR_7001. En caso contrario, mostrar mensaje de error.

G <- AddRule("e1403", "B$v7002 > 0 & B$v7002 < B$v7001", "B$v71==1")

#-------------------------
# REGLA val_7_004
#-------------------------
# Obligatorio si respondió que "Sí" en la Pregunta 1. Validar si este valor es menor que el 50%  del valor de la VAR_7001. En caso contrario, mostrar mensaje de advertencia.

G <- AddRule("e1404", "B$v7002 > 0 & B$v7002 <= 0.5*B$v7001", "B$v71==1")

#-------------------------
# REGLA val_7_005
#-------------------------
# Obligatorio si respondió a la Pregunta 7002. Validar si v7003 es mayor o igual que 0 y menor o igual que valor Pregunta 7002. En caso contrario, mostrar mensaje de error.

G <- AddRule("e1405", "B$v7003 >= 0 & B$v7003 <= B$v7002", "B$v7002 > 0")

#-------------------------
# REGLA val_7_006
#-------------------------
# Obligatorio si respondió a la Pregunta 7002. Validar si v7004 es mayor o igual que 0 y menor o igual que valor Pregunta 7002. En caso contrario, mostrar mensaje de error.

G <- AddRule("e1406", "B$v7004 >= 0 & B$v7004 <= B$v7002", "B$v7002 > 0")

#-------------------------
# REGLA val_7_007
#-------------------------
# La suma de v7003 y v7004 debe ser = al valor numérico ingresado en la Pregunta 7002.

G <- AddValidSumas("e1407", B$v7002, B$v7003, B$v7004, filtro="B$v7002 > 0")

#-------------------------
# REGLA val_7_008
#-------------------------
# El valor numérico de la Pregunta 7005 debe ser mayor o igual que el sueldo básico por el personal (450*12* Valor(Pregunta 7003)) y no puede ser mayor que la variable 5180.

G <- AddRule("e1408", "B$v7005 >= 450*12*B$v7003 & B$v7005 <= B$v5180", "B$v7003 > 0")

##-------------------------
# REGLA val_7_009
#-------------------------
# Si el valor es menor al sueldo básico (450*12* Valor(v7003), mostrar advertencia y desplegar el campo de observaciones. No pasar sin observación.
```

```
cadena_OK <- c("CAPÍTULO 7", "CAPITULO 7", "CAPÍTULO7", "CAPÍTULO7", "CAP 7", "CAP. 7", "CAP7", "CAP.7")

c1 <- unlist(sapply(cadena_OK, function(x) grep(x, B$observaciones)))
c2 <- unlist(sapply(cadena_OK, function(x) grep(x, B$observaciones_critica)))
c3 <- unlist(sapply(cadena_OK, function(x) grep(x, B$observaciones_investigador)))

c1 <- unique(as.numeric(c1))
c2 <- unique(as.numeric(c2))
c3 <- unique(as.numeric(c3))

cond_1 <- cond_2 <- cond_3 <- rep(FALSE, dim(B)[1])

cond_1[c1] <- TRUE
cond_2[c2] <- TRUE
cond_3[c3] <- TRUE
condicion <- (cond_1 | cond_2 | cond_3)

G <- AddRule("e1409", "!(B$v7005 < 450*12*B$v7003) | condicion", "B$v7003 > 0")

rm(cadena_OK, c1, c2, c3, cond_1, cond_2, cond_3, condicion)

#---------------------------
# REGLA val_7_010
#---------------------------
# Si el valor de la suma de las variables 7005+7006 es mayor que el registrado en la celda 5180, mostrar mensaje de error.

z <- apply(cbind(B$v7005, B$v7006), 1, sum, na.rm = T)
G <- AddRule("e1410", "z <= B$v5180", "B$v7002 > 0")
rm(z)

##############################################################################
# Capítulo 8. OFERTA Y UTILIZACION DE BIENES Y SERVICIOS AMBIENTALES
##############################################################################

#---------------------------
# REGLA val_8_001
#---------------------------
# Obligatorio recuperar y desplegar información bajo la condición:
# v8001 = v4146 + v4147 + v4148 + v4158(si v4157==2023) + v4164(si v4163==2023) + v4170(si v4169==2023) + v4176(si
v4175==2023) + v4179(si v4178==2023) + v4185(si v4184==2023) + v4191(si v4190==2023) + v4161(si v4160==2023) +
v4167(si v4166==2023) + v4173(si v4172==2023) + v4182(si v4181==2023) + v4188(si v4187==2023) + v4194(si v4193==2023).
# NOTA: si no se da la igualdad, muestre el mensaje de advertencia.

# Cálculo de sumas condicionales.
z1 <- ifelse(B$v4157 == 2023, B$v4158, NA_integer_)
z2 <- ifelse(B$v4163 == 2023, B$v4164, NA_integer_)
z3 <- ifelse(B$v4169 == 2023, B$v4170, NA_integer_)
z4 <- ifelse(B$v4175 == 2023, B$v4176, NA_integer_)
z5 <- ifelse(B$v4178 == 2023, B$v4179, NA_integer_)
z6 <- ifelse(B$v4184 == 2023, B$v4185, NA_integer_)
z7 <- ifelse(B$v4190 == 2023, B$v4191, NA_integer_)
z8 <- ifelse(B$v4160 == 2023, B$v4161, NA_integer_)
z9 <- ifelse(B$v4166 == 2023, B$v4167, NA_integer_)
z10 <- ifelse(B$v4172 == 2023, B$v4173, NA_integer_)
z11 <- ifelse(B$v4181 == 2023, B$v4182, NA_integer_)
z12 <- ifelse(B$v4187 == 2023, B$v4188, NA_integer_)
z13 <- ifelse(B$v4193 == 2023, B$v4194, NA_integer_)

# Cálculo de la suma global (excepto v4146, v4147, v4148).
z <- apply(cbind(z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12, z13), 1, sum, na.rm = T)

# Ejecución de regla de validación.
G <- AddValidSumas("e1411", B$v8001, B$v4146, B$v4147, B$v4148, z)

# Eliminación de variables temporales.
rm(z, z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11, z12, z13)
```

```
#---------------------------------------------
# REGLAS val_8_002...val_8_007
#---------------------------------------------
# En las columnas 1,3 y 5 debe existir obligatoriamente una respuesta (1=SI o 2=NO).

seq_i <- seq(8086, 8096, 2)
for (i in seq_i) {
  etiqueta <- paste0("e", 1411+(i-8084)/2)
  variable <- paste0("B$v", i)
  regla <- paste0(variable, " %in% 1:2")
  G <- AddRule(etiqueta, regla)
}
rm(etiqueta, i, regla, seq_i, variable)

#---------------------------------------------
# REGLAS val_8_008...val_8_013
#---------------------------------------------
# En el caso de que conteste que "SI" en las columnas 1,3,5, se debe registrar un valor > 0 en las columnas 2,4,6.

seq_i <- seq(8087, 8097, 2)
for (i in seq_i) {
  etiqueta <- paste0("e", 1417+(i-8085)/2)
  variable <- paste0("B$v", i)
  previa <- paste0("B$v", i-1)
  regla <- paste0(variable, " > 0")
  filtrado <- paste0(previa, "==1")
  G <- AddRule(etiqueta, regla, filtrado)
}
rm(etiqueta, filtrado, i, previa, regla, seq_i, variable)

#---------------------------
# REGLA val_8_014
#---------------------------
# El total de la columna 2 (línea 328, VAR_8098) no puede ser superior que VAR_2006.

G <- AddRule("e1424", "B$v8098 <= B$v2006", "B$v8098 > 0 & B$v2006 > 0")

#---------------------------
# REGLA val_8_015
#---------------------------
# El total de la columna 4 (línea 328, VAR_8099) no puede ser superior que v8001.

G <- AddRule("e1425", "B$v8099 <= B$v8001", "B$v8088 == 1 | B$v8094 == 1")

#---------------------------
# REGLA val_8_016
#---------------------------
# La variable v8091 deberá ser mayor que cero si en el Cap. 7, v7002 es mayor que 0.

G <- AddRule("e1959", "B$v8091 > 0", "B$v7002 > 0")


##################################################
# Capítulo 9. ENERGIA, COMBUSTIBLES Y LUBRICANTES
##################################################

#---------------------------
# REGLA val_9_001
#---------------------------
# En la Tabla 9.i.1, Pregunta 1, llenar obligatoriamente los campos de cantidad (2) y valor (3). La v9003 (Observación) debe ser
llenada obligatoriamente cuando las variables: v9001 (cantidad) y v9002 (valor USD) no tengan registro "missing", o el valor sea
"0".
filtro <- "(is.na(B$v9001) | B$v9001 == 0) | (is.na(B$v9002) | B$v9002 == 0)"
G <- AddRule("e1426", "nchar(trimws(as.character(B$v9003))) > 2", filtro)
rm(filtro)

#---------------------------
```

```
# REGLA val_9_002
#-------------------------
# En la Tabla 9.i.1, Columna 3, se aplica la siguiente condición: v9002 < v1173. Formato númerico (F10.0).

G <- AddRule("e1427", "B$v9002 < B$v1173", "B$v9002 > 0 & B$v1173 > 0")

#-------------------------
# REGLA val_9_003
#-------------------------
# Verificar que el Valor USD (Tabla 9.i.1, Columna 3) dividido para la Cantidad (Tabla 9.i.1, Columna 2) debe estar dentro del rango de $0.05 a $0.44.

G <- AddRule("e1428", "B$v9002 >= 0.05*B$v9001 & B$v9002 <= 0.44*B$v9001", "B$v9001 > 0")

#-------------------------
# REGLA val_9_004
#-------------------------
# En la Tabla 9.i.1, Pregunta 2, debe existir una sola respuesta : "Sí" o "No". Si contesta "No", pasar a la Sección II. COMBUSTIBLES Y LUBRICANTES.

G <- AddRule("e1429", "B$v9i2 %in% 1:2")

#-------------------------
# REGLA val_9_005
#-------------------------
# Si la variable v9044 esta marcada con "SI", el aplicativo deberá solicitar información de forma obligatoria en la variable v9057.

G <- AddRule("e1430", "nchar(trimws(as.character(B$v9057))) > 2", "B$v9044==1")

#-------------------------
# REGLA val_9_006
#-------------------------
# La energía total producida (v9052) debe ser igual a la suma de la energía consumida (v9054) + la energía vendida (v9055).

G <- AddValidSumas("e1431", B$v9052, B$v9054, B$v9055, filtro="B$v9i2==1")

#-------------------------
# REGLA val_9_007
#-------------------------
# En la Columna 2 de la Tabla 9.i.3 debe existir al menos un "Sí" (v9004, v9012, v9020, v9028, v9036 o v9044).

G <- AddRule("e1432", "B$v9004==1 | B$v9012==1 | B$v9020==1 | B$v9028==1 | B$v9036==1
      | B$v9044==1", "B$v9i2==1")

#-------------------------
# REGLA val_9_008
#-------------------------
# Si en la Columna 2 de la Tabla 9.i.3 se contesta al menos con un "Sí", debe obligatoriamente registrar que "Sí" en la Pregunta 2.

G <- AddRule("e1433", "B$v9i2==1", "B$v9004==1 | B$v9012==1 | B$v9020==1 | B$v9028==1
      | B$v9036==1 | B$v9044==1")

#-------------------------------------------
# REGLAS val_9_009...val_9_014
#-------------------------------------------
# Si responde "No" en la columna (2), bloquear la línea y pasar al siguiente tipo de energía.

z <- apply(cbind(B$v9004, B$v9012, B$v9020, B$v9028, B$v9036, B$v9044), 1, prod, na.rm = T)
z <- ifelse(z == 1, NA_integer_, 1)

seq_i <- seq(9004, 9044, 8)
for (i in seq_i) {
  etiqueta <- paste0("e", 1434+(i-9004)/8)
  variable <- paste0("B$v", i)
  secuencia <- sapply(c(1:3, 5:6), function(j) paste0("v", i + j))
  All_NA_0 <- apply(B[, secuencia], 1, function(x) {all(is.na(x) | x == 0)})
```

```
  regla <- paste0("!(", variable, "==2) | All_NA_0==TRUE")
  G <- AddRule(etiqueta, regla, "z==1 & B$v9i2==1")
}
rm(All_NA_0, etiqueta, i, regla, seq_i, secuencia, variable, z)

#---------------------------------------------
# REGLAS val_9_015...val_9_020
#---------------------------------------------
# Obligatorio si se registra información en la columna 7 debe existir información en la columna 8.

seq_i <- seq(9010, 9050, 8)
for (i in seq_i) {
  etiqueta <- paste0("e", 1440+(i-9010)/8)
  variable <- paste0("B$v", i)
  anterior <- paste0("B$v", i-1)
  regla <- paste0(variable, " > 0")
  condicion <- paste0(anterior, " > 0")
  G <- AddRule(etiqueta, regla, condicion)
}
rm(condicion, etiqueta, i, regla, seq_i, anterior, variable)

#---------------------------------------------
# REGLAS val_9_021...val_9_026
#---------------------------------------------
# La suma de los valores de las columnas (5) y (7) debe ser igual al valor registrado en la columna (3).

seq_i <- seq(9005, 9045, 8)
for (i in seq_i) {
  etiqueta <- paste0("e", 1446+(i-9005)/8)
  variable <- paste0("B$v", i)
  sumando_1 <- paste0("B$v", i+2)
  sumando_2 <- paste0("B$v", i+4)
  filtrado <- paste0("B$v", i-1, "==1")
  comando <- paste0("G <- AddValidSumas('", etiqueta, "', ", variable, ", ", sumando_1,
                    ", ", sumando_2, ", filtro='", filtrado, "')")
  # eval(parse(text = comando), envir = .GlobalEnv)
  eval(parse(text = comando))
}
rm(comando, etiqueta, filtrado, i, seq_i, sumando_1, sumando_2, variable)
#---------------------------------------------
# REGLAS val_9_027...val_9_032
#---------------------------------------------
# Si existe valor en la columna 8, éste debe ser >= que el valor de la columna 4.

seq_i <- seq(9010, 9050, 8)
for (i in seq_i) {
  etiqueta <- paste0("e", 1452+(i-9010)/8)
  variable <- paste0("B$v", i)
  anterior <- paste0("B$v", i - 4)
  siguiente <- paste0("B$v", i + 1)
  regla1 <- paste0("(", variable, " >= ", anterior, ")")
  regla2 <- paste0("(", variable, " < ", anterior, ") & (nchar(trimws(as.character(", siguiente, "))) > 2)")
  regla <- paste0(regla1, " | ", regla2)
  filtrado <- paste0("(B$v", i-6, " ==1) & (B$v", i-5, " > B$v", i-3, ")")
  G <- AddRule(etiqueta, regla, filtrado)
}
rm(anterior, etiqueta, filtrado, i, seq_i, regla1, regla2, regla, siguiente, variable)

#---------------------------------------------
# REGLAS val_9_033...val_9_038
#---------------------------------------------
# Si los valores de la columna (7) son < que los valores de la columna (3), entonces obligatoriamente debe existir valor positivo
en la columna (5).

seq_i <- seq(9008, 9048, 8)
for (i in seq_i) {
  etiqueta <- paste0("e", 1458+(i-9008)/8)
```

```
  anterior <- paste0("B$v", i-1)
  regla <- paste0(anterior, " > 0")
  condicion <- paste0("B$v", i+1, " < ", "B$v", i-3)
  G <- AddRule(etiqueta, regla, condicion)
}
rm(anterior, etiqueta, condicion, i, seq_i, regla)

#---------------------------------------------
# REGLAS val_9_039...val_9_044
#---------------------------------------------
# En la pregunta 3 validar los valores de la columna VALOR (4) con respecto a la columna KWH/año (3), según la regla: "El Valor USD(4) dividido para los KWH/año(3) debe estar dentro del rango de 0.05 a 1.00".

G <- AddRule("e1464", "B$v9006 >= 0.04*B$v9005 & B$v9006 <= 1.01*B$v9005", "B$v9005 > 0")
G <- AddRule("e1465", "B$v9014 >= 0.04*B$v9013 & B$v9014 <= 1.01*B$v9013", "B$v9013 > 0")
G <- AddRule("e1466", "B$v9022 >= 0.04*B$v9021 & B$v9022 <= 1.01*B$v9021", "B$v9021 > 0")
G <- AddRule("e1467", "B$v9030 >= 0.04*B$v9029 & B$v9030 <= 1.01*B$v9029", "B$v9029 > 0")
G <- AddRule("e1468", "B$v9038 >= 0.04*B$v9037 & B$v9038 <= 1.01*B$v9037", "B$v9037 > 0")
G <- AddRule("e1469", "B$v9046 >= 0.04*B$v9045 & B$v9046 <= 1.01*B$v9045", "B$v9045 > 0")

#---------------------------------------------
# REGLAS val_9_045...val_9_050
#---------------------------------------------
# Si se registran valores en la columna 7, no podrán ser mayores que los valores de la columna 3.

G <- AddRule("e1470", "B$v9009 <= B$v9005", "B$v9005 > 0")
G <- AddRule("e1471", "B$v9017 <= B$v9013", "B$v9013 > 0")
G <- AddRule("e1472", "B$v9025 <= B$v9021", "B$v9021 > 0")
G <- AddRule("e1473", "B$v9033 <= B$v9029", "B$v9029 > 0")
G <- AddRule("e1474", "B$v9041 <= B$v9037", "B$v9037 > 0")
G <- AddRule("e1475", "B$v9049 <= B$v9045", "B$v9045 > 0")

#---------------------------
# REGLA val_9_051
#---------------------------
# Si al menos una de las variables v9010, v9018, v9026, v9034 y v9050 es > 0, entonces en la Línea 327 la variable v8093 debe ser > 0.

G <- AddRule("e1476", "B$v8093 > 0", "B$v9010 > 0 | B$v9018 > 0 | B$v9026 > 0 | B$v9034 > 0 | B$v9050 > 0")

#---------------------------
# REGLA val_9_081
#---------------------------
# Verificar que v9105 (linea 341, columna 3) sea <= a la suma de las v1130 + v1146.

z <- apply(cbind(B$v1130, B$v1146), 1, sum, na.rm = T)
G <- AddRule("e1477", "B$v9105 <= z", "B$v9ii1==1")
rm(z)

#---------------------------
# REGLA val_9_094
#---------------------------
# Línea 329 (Súper) -> Verificar que el cociente de las variables v9059 y v9058 debe estar en el rango entre 3.37 y 5.20 (incluyendo los extremos).

G <- AddRule("e1478", "B$v9059 >= 3.36 * B$v9058 & B$v9059 <= 5.21 * B$v9058", "B$v9058 > 0")

#---------------------------
# REGLA val_9_095
#---------------------------
# Línea 330 (Extra) -> Verificar que el cociente de las variables v9063 y v9062 debe estar en el rango entre 2.4 y 3.00 (incluyendo los extremos).

G <- AddRule("e1479", "B$v9063 >= 2.39 * B$v9062 & B$v9063 <= 3.01 * B$v9062", "B$v9062 > 0")

#---------------------------
# REGLA val_9_096
```

```
#---------------------------
# Línea 331 (Jet Fuel) -> Verificar que el cociente de las variables v9067 y v9066 debe estar en el rango entre 2.44 y 4.00 (incluyendo los extremos).

G <- AddRule("e1480", "B$v9067 >= 2.43 * B$v9066 & B$v9067 <= 4.01 * B$v9066", "B$v9066 > 0")

#---------------------------
# REGLA val_9_097
#---------------------------
# Línea 332 (Diesel) -> Verificar que el cociente de las variables v9071 y v9070 debe estar en el rango entre 1.75 y 3.13 (incluyendo los extremos).

G <- AddRule("e1481", "B$v9071 >= 1.74 * B$v9070 & B$v9071 <= 3.14 * B$v9070", "B$v9070 > 0")

#---------------------------
# REGLA val_9_098
#---------------------------
# Línea 333 (Gas Licuado GLP) -> Verificar que el cociente de las variables v9075 y v9074 debe estar en el rango entre 0.11 y 2.0 (incluyendo los extremos).
G <- AddRule("e1482", "B$v9075 >= 0.10 * B$v9074 & B$v9075 <= 2.01 * B$v9074", "B$v9074 > 0")

#---------------------------
# REGLA val_9_099
#---------------------------
# Línea 334 (Gas Natural) -> Verificar que el cociente de las variables v9079 y v9078 debe estar en el rango entre 0.33 y 10.1 (incluyendo los extremos).

G <- AddRule("e1483", "B$v9079 >= 0.32 * B$v9078 & B$v9079 <= 10.11 * B$v9078", "B$v9078 > 0")

#---------------------------
# REGLA val_9_100
#---------------------------
# Línea 335 (Residuo Fuel Oil) -> Verificar que el cociente de las variables v9083 y v9082 debe estar en el rango entre 0.50 y 1.68 (incluyendo los extremos).

G <- AddRule("e1484", "B$v9083 >= 0.49 * B$v9082 & B$v9083 <= 1.69 * B$v9082", "B$v9082 > 0")

#---------------------------
# REGLA val_9_101
#---------------------------
# Línea 336 (Crudo Residual) -> Verificar que el cociente de las variables v9087 y v9086 debe estar en el rango entre 0.05 y 1.15 (incluyendo los extremos).

G <- AddRule("e1485", "B$v9087 >= 0.04 * B$v9086 & B$v9087 <= 1.16 * B$v9086", "B$v9086 > 0")

#---------------------------
# REGLA val_9_102
#---------------------------
# Línea 337 (Carbón) -> Verificar que el cociente de las variables v9091 y v9090 debe estar en el rango entre 0.10 y 0.4 (incluyendo los extremos).

G <- AddRule("e1486", "B$v9091 >= 0.09 * B$v9090 & B$v9091 <= 0.41 * B$v9090", "B$v9090 > 0")

#---------------------------
# REGLA val_9_103
#---------------------------
# Línea 338 (Ecopaís)- > Verificar que el cociente de las variables v9095 y v9094 debe estar en el rango entre 2.4 y 3.0 (incluyendo los extremos).

G <- AddRule("e1487", "B$v9095 >= 2.39 * B$v9094 & B$v9095 <= 3.01 * B$v9094", "B$v9094 > 0")

#---------------------------
# REGLA val_9_104
#---------------------------
# Línea 339 (Aceites) -> Verificar que el cociente de las variables v9099 y v9098 debe estar en el rango entre 3.1 y 49.17 (incluyendo los extremos).
```

```
G <- AddRule("e1488", "B$v9099 >= 3.09 * B$v9098 & B$v9099 <= 49.18 * B$v9098", "B$v9098 > 0")

######################
# Capítulo 10.1. AGUA
######################

#---------------------------------
# REGLA val_10_1_001
#---------------------------------
# En la pregunta 1, llenar obligatoriamente los campos de cantidad (2) y valor (3). La variable v10002 (Observación) debe ser llenada obligatoriamente cuando las variables: v10000 (cantidad) y v10001 (valor USD) no tengan registro "missing" o el valor sea "0".

filtro <- "(is.na(B$v10000) | B$v10000 == 0) | (is.na(B$v10001) | B$v10001 == 0)"
G <- AddRule("e1489", "nchar(trimws(as.character(B$v10002))) > 2", filtro)
rm(filtro)

#---------------------------------
# REGLA val_10_1_002
#---------------------------------
# En la columna (3) se aplica la siguiente condición: VAR_10001 < VAR_1173.

G <- AddRule("e1490", "B$v10001 < B$v1173", "B$v10001 > 0")

#---------------------------------
# REGLA val_10_1_003
#---------------------------------
# Incluir en la pregunta 1. un mensaje de advertencia que valide los valores de la columna VALOR con respecto de la columna CANTIDAD. El mensaje de advertencia sería: "El Valor USD(3) dividido para la Cantidad(2) debe estar dentro del rango de $0.40 a $1.30".

G <- AddRule("e1491", "B$v10001 >= 0.39 * B$v10000 & B$v10001 <= 1.31 * B$v10000 & is.finite(B$v10001 / B$v10000)",
"B$v10000 > 0")

#---------------------------------
# REGLA val_10_1_004
#---------------------------------
# En la Pregunta 2, en el caso de contestar "SI" se sigue a la pregunta 2.1. En el caso de contestar "NO", se pasa a la Pregunta 3 (Fuentes de captación de agua).

G <- AddRule("e1492", "B$v10i2 %in% 1:2")

#---------------------------------
# REGLA val_10_1_005
#---------------------------------
# En la pregunta 2.1 Debe existir una sola respuesta numérica en las columnas (1), (2) y (3). Esto para los que contestaron "SI" en la pregunta 2.

G <- AddRule("e1493", "(B$v10003 %in% 1:2) & (B$v10004 > 0) & (B$v10005 > 0)", "B$v10i2==1")

#---------------------------------
# REGLA val_10_1_006
#---------------------------------
# En la Pregunta 3, en el caso de contestar "SI" sigue el flujo. En el caso de contestar "NO", pasar a la Sección II. Aguas Residuales/desechadas.

G <- AddRule("e1494", "B$v10i3 %in% 1:2")

#---------------------------------
# REGLA val_10_1_007
#---------------------------------
# Si en la pregunta 3 se respondió que SI, debe haber por lo menos un "SI" por respuesta en la columna (1).

G <- AddRule("e1495", "B$v10006==1 | B$v10014==1 | B$v10022==1", "B$v10i3==1")


#-----------------------------------------------------
```

```
# REGLAS val_10_1_011...val_10_1_013
#----------------------------------------------------------
# Si en la columna (3) contesta "SI" debe haber valor en las columnas (4), (5) y (6).

G <- AddRule("e1496", "B$v10009 > 0 & B$v10010 > 0 & B$v10011 > 0", "B$v10008==1")
G <- AddRule("e1497", "B$v10017 > 0 & B$v10018 > 0 & B$v10019 > 0", "B$v10016==1")
G <- AddRule("e1498", "B$v10025 > 0 & B$v10026 > 0 & B$v10027 > 0", "B$v10024==1")

#----------------------------------------------------------
# REGLAS val_10_1_014...val_10_1_016
#----------------------------------------------------------
# Si en la columna (1) contesta "NO", automáticamente se deberá pasar a la siguiente fuente de captación de agua.

z <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, prod, na.rm = T)
z <- ifelse(z == 1, NA_integer_, 1)

for (i in seq(10006, 10022, 8)) {
  etiqueta <- paste0("e", 1499+(i-10006)/8)
  variable <- paste0("B$v", i)
  secuencia <- sapply(2:7, function(j) paste0("v", i + j))
  All_NA_0 <- apply(B[, secuencia], 1, function(x) {all(is.na(x) | x == 0)})
  regla <- paste0("!(", variable, "==2) | All_NA_0==TRUE")
  G <- AddRule(etiqueta, regla, "z!=1 & B$v10i3==1")
}
rm(All_NA_0, etiqueta, i, regla, secuencia, variable, z)

#----------------------------------------------------------
# REGLAS val_10_1_017...val_10_1_019
#----------------------------------------------------------
# Si en la columna (3) contesta "NO", debe bloquear el ingreso de valores en las columnas (4),(5),(6) y (7).

z <- apply(cbind(B$v10008, B$v10016, B$v10024), 1, prod, na.rm = T)
z <- ifelse(z == 1, NA_integer_, 1)

for (i in seq(10008, 10024, 8)) {
  etiqueta <- paste0("e", 1502+(i-10008)/8)
  variable <- paste0("B$v", i)
  secuencia <- sapply(1:4, function(j) paste0("v", i + j))
  All_NA_0 <- apply(B[, secuencia], 1, function(x) {all(is.na(x) | x == 0)})
  regla <- paste0("!(", variable, "==2) | All_NA_0==TRUE")
  G <- AddRule(etiqueta, regla, "z!=1 & (B$v10006 == 1 | B$v10014 == 1 | B$v10022 == 1)")
}
rm(All_NA_0, etiqueta, i, regla, secuencia, variable, z)

#----------------------------------------------------------
# REGLAS val_10_1_020...val_10_1_022
#----------------------------------------------------------
# La información registrada en las variables v10012, v10020 y v10028 debe existir respuesta de formato numérico de escala
(F10.2).

G <- AddRule("e1505", "(100*B$v10012) %% 100 >= 0", "B$v10008==1 & B$v10012 > 0")
G <- AddRule("e1506", "(100*B$v10020) %% 100 >= 0", "B$v10016==1 & B$v10020 > 0")
G <- AddRule("e1507", "(100*B$v10028) %% 100 >= 0", "B$v10024==1 & B$v10028 > 0")


#----------------------------------------------------------
# REGLAS val_10_1_023...val_10_1_025
#----------------------------------------------------------
# Las variables v10010, v10018, v10026 deben ser menores o iguales que 24.

G <- AddRule("e1508", "B$v10010 %in% 1:24", "B$v10008==1")
G <- AddRule("e1509", "B$v10018 %in% 1:24", "B$v10016==1")
G <- AddRule("e1510", "B$v10026 %in% 1:24", "B$v10024==1")

#----------------------------------------------------------
# REGLAS val_10_1_026...val_10_1_028
#----------------------------------------------------------
```

```
# Las variables v10011, v10019, v10027 deben ser menores o iguales que 30.

G <- AddRule("e1511", "B$v10011 %in% 1:30", "B$v10008==1")
G <- AddRule("e1512", "B$v10019 %in% 1:30", "B$v10016==1")
G <- AddRule("e1513", "B$v10027 %in% 1:30", "B$v10024==1")

#-----------------------------------------------------------
# REGLAS val_10_1_029...val_10_1_031
#-----------------------------------------------------------
# Tabla 10.i.3, Columna (4), Caudal de agua captada que usó la empresa m3/h", se debe permitir el ingreso de valores mayores
a cero con dos decimales.

G <- AddRule("e1514", "(100*B$v10009) %% 100 >= 0", "B$v10008==1 & B$v10009 > 0")
G <- AddRule("e1515", "(100*B$v10017) %% 100 >= 0", "B$v10016==1 & B$v10017 > 0")
G <- AddRule("e1516", "(100*B$v10025) %% 100 >= 0", "B$v10024==1 & B$v10025 > 0")

#---------------------------------
# REGLA val_10_1_032
#---------------------------------
# Línea 342 -> En el casillero 10012 se despliega el cálculo de la fórmula: Var(10012) = 12 * Var(10009) * Var(10010) *
Var(10011).

B$DOCE <- 12
z <- apply(cbind(B$DOCE, B$v10009, B$v10010, B$v10011), 1, prod, na.rm =T)
z <- round(z, 2)
G <- AddRule("e1517", "round(B$v10012, 2) == z", "B$v10008==1")
rm(z)

#---------------------------------
# REGLA val_10_1_033
#---------------------------------
# Línea 343 -> En el casillero 10020 se despliega el cálculo de la fórmula: Var(10020) = 12 * Var(10017) * Var(10018) *
Var(10019).

z <- apply(cbind(B$DOCE, B$v10017, B$v10018, B$v10019), 1, prod, na.rm =T)
z <- round(z, 2)
G <- AddRule("e1518", "round(B$v10020, 2) == z", "B$v10016==1")
rm(z)

#---------------------------------
# REGLA val_10_1_034
#---------------------------------
# Línea 344 -> En el casillero 10028 se despliega el cálculo de la fórmula: Var(10028) = 12 * Var(10025) * Var(10026) *
Var(10027).

z <- apply(cbind(B$DOCE, B$v10025, B$v10026, B$v10027), 1, prod, na.rm =T)
z <- round(z, 2)
G <- AddRule("e1519", "round(B$v10028, 2) == z", "B$v10024==1")
rm(z)

#---------------------------------
# REGLA val_10_1_035
#---------------------------------
# Verificar la sumatoria de las variables: v9002 + v10001 <= v1173.

z <- apply(cbind(B$v9002, B$v10001), 1, sum, na.rm =T)
G <- AddRule("e1520", "z <= B$v1173")
rm(z)

#---------------------------------
# REGLA val_10_1_036
#---------------------------------
# Línea 342 -> Si en la columna (1) contesta que "SI" (v10006 == 1), debe haber obligatoriamente valor positivo en la columna (8)
(v10013 > 0).

G <- AddRule("e1521", "B$v10013 > 0", "B$v10006 == 1")
```

```
#---------------------------------
# REGLA val_10_1_037
#---------------------------------
# Línea 343 -> Si en la columna (1) contesta que "SI" (v10014 == 1), debe haber obligatoriamente valor positivo en la columna (8) (v10021 > 0).

G <- AddRule("e1522", "B$v10021 > 0", "B$v10014 == 1")

#---------------------------------
# REGLA val_10_1_038
#---------------------------------
# Línea 344 -> Si en la columna (1) contesta que "SI" (v10022 == 1), debe haber obligatoriamente valor positivo en la columna (8) (v10029 > 0).

G <- AddRule("e1523", "B$v10029 > 0", "B$v10022 == 1")


##############################################
# Capítulo 10.2. AGUAS RESIDUALES / DESECHADAS
##############################################

#---------------------------------
# REGLA val_10_2_001
#---------------------------------
# Obligatorio responder a la Pregunta 10.ii.1. En el caso de contestar "Si", ir a la Pregunta 10.ii.1.1; en el caso de contestar "No", pasar a la Pregunta 10.ii.2.

G <- AddRule("e1524", "B$v10ii1 %in% 1:2")

#---------------------------------
# REGLA val_10_2_002
#---------------------------------
# Obligatorio responder a la Pregunta 10.ii.1.1 si respondió que "Sí" a la Pregunta 10.ii.1. Esta variable admite como valor mínimo 0.01. Formato numérico de escala. Formato numérico (F10.2)

G <- AddRule("e1525", "B$v10ii11 >= 0.01", "B$v10ii1 == 1")


#---------------------------------
# REGLA val_10_2_003
#---------------------------------
# Obligatorio responder a la Pregunta 10.ii.2. Debe existir una sola respuesta "Sí" ó "No". Si el informante contesta "No", pasar directamente al Capítulo 10, Sección III. Otros Residuos y/o desechos.

G <- AddRule("e1526", "B$v10ii2 %in% 1:2")

#---------------------------------
# REGLA val_10_2_004
#---------------------------------
# Obligatorio responder a la Pregunta 10.ii.3 si respondió que "Si" a la Pregunta 10.ii.2 es "Si". Debe existir una sola respuesta "Sí" ó "No". Si el informante contesta "No" pasar directamente a la Pregunta 10.ii.5.

G <- AddRule("e1527", "B$v10ii3 %in% 1:2", "B$v10ii2 == 1")

#---------------------------------
# REGLA val_10_2_005
#---------------------------------
# Debe existir respuesta de formato numérico de escala (F10.2) en las Columnas 1 y 4 de la Tabla 10.ii.4 (Var10032 y Var10033, respectivamente).

oper <- function(x) {((100*x) %% 100) >= 0}
regla <- "oper(B$v10032) & oper(B$v10035)"
G <- AddRule("e1528", regla, "B$v10ii3 == 1")
rm(oper, regla)

#---------------------------------
# REGLA val_10_2_006
```

```
#---------------------------------
# En el casillero v10035 se despliega el cálculo de la fórmula: v10035 = 12 * v10032 * v10033 * v10034.

z <- apply(cbind(B$DOCE, B$v10032, B$v10033, B$v10034), 1, prod, na.rm =T)
z <- round(z, 2)
G <- AddRule("e1529", "round(B$v10035, 2) == z", "B$v10032 > 0")
rm(z)
B$DOCE <- NULL

#---------------------------------
# REGLA val_10_2_007
#---------------------------------
# Además, para la verificación siguiente, comprobar que Var 10004 (Preg. 3.1) esté en m3. Si no, convertir previamente a m3 la cantidad que está en la unidad de volumen actual. Verificar que: Var10035 <= Var10000 + Var10004 + Var10030.

Agua_Tank <- ifelse(B$v10003 == 2, B$v10004, ifelse(B$v10003 == 1, B$v10004/264.17205, NA)) # 1=US gal; 2=m3.
Agua_consumo <- apply(cbind(B$v10000, Agua_Tank, B$v10030), 1, sum, na.rm = T)
G <- AddRule("e1530", "B$v10035 <= Agua_consumo", "B$v10ii3 == 1")
rm(Agua_consumo, Agua_Tank)

#---------------------------------
# REGLA val_10_2_008
#---------------------------------
# Si en la Pregunta 10.ii.3 se contestó que "Si", verificar que v10032 > 0, v10033 > 0, v10034 > 0, v10035 > 0.

G <- AddRule("e1531", "B$v10032 > 0 & B$v10033 > 0 & B$v10034 > 0", "B$v10ii3 == 1")


#---------------------------------
# REGLA val_10_2_009
#---------------------------------
# Las variables: v10033 <= 24 y v10034 <= 30.

G <- AddRule("e1532", "B$v10033 %in% 1:24 & B$v10034 %in% 1:30", "B$v10ii3 == 1")

#---------------------------------
# REGLAS val_10_2_010
#---------------------------------
# Si el proceso productivo de la empresa generó aguas residuales (v10ii2 == 1) y no dio tratamiento a las mismas (Var10ii5 == 2), verificar que no se haya seleccionado ningún tipo de tratamiento de aguas residuales.

z <- apply(cbind(B$v10ii5111, B$v10ii5112, B$v10ii5113, B$v10ii5114), 1, min)
G <- AddRule("e1533", "is.na(z)", "B$v10ii2==1 & B$v10ii5==2")

#---------------------------------
# REGLAS val_10_2_011
#---------------------------------
# Si la empresa ha dado algún tipo de tratamiento a sus aguas residuales (v10ii5 == 1), verificar que al menos uno de los cuatro tipos de tratamiento ha sido marcado.

G <- AddRule("e1534", "z == 1", "B$v10ii5 == 1")

#---------------------------------
# REGLAS val_10_2_015
#---------------------------------
# Si el informante responde "SI" para al menos un tipo de tratamiento de aguas residuales, la variable 2"Ninguno" no deberá estar marcada.

G <- AddRule("e1535", "B$v10ii5 == 1", "z == 1")
rm(z)

#---------------------------------
# REGLA val_10_2_017
#---------------------------------
# Si en el Capítulo 10, Sección II,  Pregunta 5, se seleccionó algún tipo de tratamiento a sus aguas residuales, entonces en la Línea 326 del Capítulo 8, celdas 8089 y 8091, al menos una de ellas deberá tener valores mayores que 0.
```

```
G <- AddRule("e1536", "B$v8089 > 0 | B$v8091 > 0", "B$v10ii5 == 1")

#---------------------------------
# REGLA val_10_2_019
#---------------------------------
# En la Pregunta 6 debe existir una respuesta numérica entre 1% y 100%. Si no se cumple con esta restricción, mostrar mensaje de error.

G <- AddRule("e1537", "B$v10ii6 %in% 1:100", "B$v10ii2 == 1 & B$v10ii5 == 1")


##############################################
# Capítulo 10.3. OTROS RESIDU0S Y/O DESECHOS
##############################################

# Bloque de código que crea variables "globales" (dentro del entorno de ejecución local) que sirven para realizar las iteraciones de las reglas de validación de residuos y/o desechos. Las variables creadas en este bloque de código se consideran como variables
# "globales" para la ejecución de todas las reglas de validación del Capítulo 10, Sección III (Residuos y Desechos).
seq_cad <- c("seq(10062, 10272, 21)", "10314", "seq(10377, 10461, 21)", "seq(10524, 10986, 21)")
# Cadenas que contienen los rangos de secuenciales de las variables de la columna (1) de las Tablas de Residuos y/o Desechos.

num_var <- as.integer(unlist(sapply(seq_cad, function(x) eval(parse(text = x)))))
# Este vector de cadenas contiene los secuenciales numéricos de las variables de la columna (1) de las Tablas de Residuos y/o Desechos.

seq_var <- paste0("v", num_var) # Vector de cadenas con las variables de la columna (1) de las Tablas de Residuos y/o Desechos.
# if (isFALSE("package:foreach" %in% search())) library(foreach) # Carga condicional de la librería que ofrece la posibilidad de construir objetos iterables.

#-----------------------------------------------------
# REGLAS val_10_3_001...val_10_3_040
#-----------------------------------------------------
# En la Columna 1 de las Tablas 1,2,3 de Residuos y/o Desechos se debe responder obligatoriamente con "Si" o "No".

etiqueta <- paste0("e", 1538:1577)
# 'etiqueta' es un vector de cadenas que contiene las etiquetas de las variables de control correspondiente a las variables de control a generar.

variable <- sapply(1:length(num_var), function(i) paste0("B$", seq_var[i])) # Vector de cadenas con los nombres de variable de la columna (1) de las Tablas de Residuos y/o Desechos que se validarán.

regla <- paste0(variable, " %in% 1:2") # Vector de cadenas con las reglas de validación a ejecutar.

comando <- sapply(1:length(regla), function(k) paste0("G <- AddRule('", etiqueta[k], "', '", regla[k], "')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_001 hasta val_10_3_040. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación. No se usa filtro porque cada regla de validación se aplica a todas las empresas sin excepción.

rm(etiqueta, variable, regla, comando)
# Eliminación de las variables temporales creadas en el bloque de código actual.

#-----------------------------------------------------
# REGLAS val_10_3_041...val_10_3_080
#-----------------------------------------------------
# Si responde "No" en esta variable, bloquear toda la línea (incluyendo la Columna 9) y pasar al siguiente residuo y/o desecho.

B$inec_ciiu4 <- B$ciiu4_actividad_principal

activ_econ <- substr(B$inec_ciiu4, 1, 1) # Letra de la actividad económica.

filtro <- 'activ_econ %in% c("B", "C", "D", "E", "F", "G", "H", "I")'
# "filtro" sirve como condición de definición de las empresas que obligatoriamente debe escrutar la presente regla de validación.

etiqueta <- paste0("e", 1578:1617)
```

```
variable <- paste0("B$", seq_var)
sucesivas <- sapply(c(2,3,5,7,9,11,17,18,19), function(j) paste0("v", num_var+j))

All_NA <- rep(NA, dim(B)[1])
for (j in 1:length(num_var)) {
  All_NA <- cbind(All_NA, apply(B[, sucesivas[j, ]], 1, function(x) all(is.na(x))))
}
All_NA <- All_NA[, -1]
All_NA <- data.frame(All_NA)
# La matriz 'All_NA' genera 40 columnas, una por cada residuo y/o desecho. Cada columna tiene el mismo número de componentes que empresas a validar. Cada celda de esta matriz tiene un valor lógico (Verdadero o Falso) que indica si la empresa de la misma fila tiene todos los valores 'missing' (en caso Verdadero) para el residuo y/o desecho de la columna 1...40.
names(All_NA) <- paste0("desecho_", 1:length(num_var))

regla1 <- paste0("!(", variable, " == 2)")

regla <- paste0("(", regla1, " | All_NA[, ", 1:length(num_var), "] == TRUE)")
# Cadena con la regla de validación concreta.

comando <- sapply(1:length(regla), function(k) paste0("G <- AddRule('", etiqueta[k], "', '", regla[k], "', '", filtro, "')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_041 hasta val_10_3_080. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(activ_econ, filtro, etiqueta, variable, sucesivas, All_NA, j, regla1, regla, comando)
# Eliminación de las variables temporales creadas en el bloque de código actual.

#---------------------------------------------------------
# REGLAS val_10_3_081...val_10_3_120
#---------------------------------------------------------
# Si en esta variable (Cantidad) se registra valores > 0, entonces verificar la siguiente suma: col 3.1 + col 3.2 + col 3.3 + col 4.1 = col 2.2.

seq_vnum <- num_var + 3
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna 2.2. Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v2_2 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables pivote (col 2.2).
seq_v3_1 <- paste0("v", seq_vnum + 2) # Vector cadena con la secuencia de variables columna 3.1.
seq_v3_2 <- paste0("v", seq_vnum + 4) # Vector cadena con la secuencia de variables columna 3.2.
seq_v3_3 <- paste0("v", seq_vnum + 6) # Vector cadena con la secuencia de variables columna 3.3.
seq_v4_1 <- paste0("v", seq_vnum + 8) # Vector cadena con la secuencia de variables columna 4.1.

z_condic <- data.frame(apply(B[, seq_v2_2], 2, function(x) x > 0))
# El frame z_condic contiene en sus columnas los filtros de positividad para las variables de la columna pivote (col 2.2).

z_2_2 <- B[, seq_v2_2]
# El frame Z_2_2 contiene en sus columnas los valores concretos de las variables de la columna pivote (col 2.2).

z_suma <- apply(B[, c(seq_v3_1[1], seq_v3_2[1], seq_v3_3[1], seq_v4_1[1])], 1, sum, na.rm = T)
for (j in 2:length(seq_vnum)) {
  z_suma <- cbind(z_suma, apply(B[, c(seq_v3_1[j], seq_v3_2[j], seq_v3_3[j], seq_v4_1[j])], 1, sum, na.rm = T))
}
z_suma <- data.frame(z_suma[, 1:40])
names(z_suma) <- names(z_2_2)
# z_suma es un dataframe de 40 columnas (una por cada residuo / desecho), donde cada celda contiene el valor de la suma de las variables de las columnas 3.1, 3.2, 3.3 y 4.1 para el residuo / desecho correspondiente a la columna elegida y para la empresa de la fila correspondiente.

diff <- z_2_2 - z_suma # Este dataframe es la diferencia entre la columna 2.2 y el vector suma de las columnas 3.1, 3.2, 3.3 y 4.1. Teóricamente, debe ser igual a {0, NA}.

etiqueta <- paste0("e", 1618:1657) # Vector con etiquetas de las reglas de validación actuales.

All_NA_0 <- data.frame(ifelse(diff == 0 | is.na(diff), TRUE, FALSE))
# All_NA_0 es el vector lógico que marca con TRUE a las componentes de 'diff' que son nulas o "missing".
```

```
comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                        "', 'All_NA_0[, ", k, "]==TRUE', 'z_condic[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_081 hasta val_10_3_120. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1, z_condic, z_2_2, j, z_suma)
rm(diff, etiqueta, All_NA_0, comando)
# Eliminación de las variables temporales creadas en el bloque de código actual.

#---------------------------------------------------------
# REGLAS val_10_3_121...val_10_3_160
#---------------------------------------------------------
# En esta variable de las Tablas 10.III, se puede seleccionar únicamente las opciones de unidades de medida siguientes: 1 Kilogramo; 2 Tonelada (Líneas 361 a 383), 1 Kilogramo; 2 Tonelada; 3 Galones americanos (Líneas 377; 384 a 405).

seq_vnum_2 <- c(seq(10064, 10274, 21), 10316, 10379, seq(10421, 10463, 21), 10526)
# Secuenciales de variables de cantidad con opciones 1 Kilogramo; 2 Tonelada.

seq_vnum_3 <- c(10400, seq(10547, 10988, 21))
# Secuenciales de variables de cantidad con opciones 1 Kilogramo; 2 Tonelada; 3 US Gal.

seq_vnum <- sort(union(seq_vnum_2, seq_vnum_3)) # Secuenciales de todas las variables de cantidad.

seq_v2_2 <- paste0("v", seq_vnum_2) # Vector cadena con la secuencia de variables pivote (col 2.2), con dos unidades.
seq_v2_3 <- paste0("v", seq_vnum_3) # Vector cadena con la secuencia de variables de la (col 2.2), con tres unidades.

# z_regla_2 <- data.frame(matrix(ncol = length(seq_vnum_2), nrow = dim(B)[1]))
# for (j in 1:17) z_regla_2[, j] <- ifelse(B[, seq_v2_2[j]] %in% 1:2, TRUE, FALSE)

z_regla_2 <- data.frame(sapply(B[, seq_v2_2], function(x) (x %in% 1:2)))
#
# z_regla_3 <- data.frame(matrix(ncol = length(seq_vnum_3), nrow = dim(B)[1]))
# for (j in 1:23) z_regla_3[, j] <- ifelse(B[, seq_v2_3[j]] %in% 1:3, TRUE, FALSE)

z_regla_3 <- data.frame(sapply(B[, seq_v2_3], function(x) (x %in% 1:3)))

w_regla <- cbind.data.frame(z_regla_2, z_regla_3)
nombres_w <- sort(names(w_regla))

z_regla <- w_regla[, nombres_w]
# "z_regla" es el frame lógico que marca con TRUE a las variables de la columna 2.1 de todas las Tabla 10.III que tienen las unidades de medida con código correcto.

z_condic <- data.frame(apply(B[, names(z_regla)], 2, function(x) x > 0))
# "z_condic" es el vector lógico que marca con TRUE a las variables de la columna 2.1 de la Tabla 1 que tienen valores positivos. Esto es, se conoce la unidad de medida del residuo generado.

etiqueta <- paste0("e", 1658:1697) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                        "', 'z_regla[, ", k, "]==TRUE', 'z_condic[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_121 hasta val_10_3_160. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum_2, seq_vnum_3, seq_vnum, seq_v2_2, seq_v2_3, z_regla_2, z_regla_3, j)
rm(w_regla, nombres_w, z_regla, z_condic, etiqueta, comando)


#---------------------------------------------------------
# REGLAS val_10_3_161...val_10_3_200
#---------------------------------------------------------
# Si en esta variable (Col. 1) se respondió que "Sí" y no se conoce ni la unidad (columna 2.1) ni la cantidad del residuo (columna 2.2), entonces en las columnas 3.1, 3.2, 3.3 y 4.1 debe ingresarse el valor 0 únicamente para la(s) gestión(es) realizada(s).

seq_vnum <- num_var
```

```
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna 1. Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 1.
seq_v2_1 <- paste0("v", seq_vnum + 2) # Vector cadena con la secuencia de variables columna 2.1.
seq_v2_2 <- paste0("v", seq_vnum + 3) # Vector cadena con la secuencia de variables columna 2.2.
seq_v3_1 <- paste0("v", seq_vnum + 5) # Vector cadena con la secuencia de variables columna 3.1.
seq_v3_2 <- paste0("v", seq_vnum + 7) # Vector cadena con la secuencia de variables columna 3.2.
seq_v3_3 <- paste0("v", seq_vnum + 9) # Vector cadena con la secuencia de variables columna 3.3.
seq_v4_1 <- paste0("v", seq_vnum + 11) # Vector cadena con la secuencia de variables columna 4.1.

z_1 <- data.frame(apply(B[, seq_v1], 2, function(x) x==1))
# El frame z_1 contiene en sus columnas los filtros de igualdad a 1 para las variables de la columna pivote (col 1).

z_2_1 <- data.frame(apply(B[, seq_v2_1], 2, is.na))
# El frame z_2_1 contiene en sus columnas los filtros de valor vacío para las variables de la columna 2.1.

z_2_2 <- data.frame(apply(B[, seq_v2_2], 2, is.na))
# El frame z_2_2 contiene en sus columnas los filtros de valor nulo para las variables de la columna 2.2.

z_condic <- data.frame(z_1 & z_2_1 & z_2_2)
# "z_condic" es el vector lógico que marca con TRUE a las variables de la columna 2.2 de todas las Tablas de desechos / residuos que cumplen con la condición de tener en la columna 1 el valor "Sí" y en las columnas 2.1 y 2.2 tener valores vacíos (missing) y nulos, respectivamente.

z_ceros <- apply(B[, c(seq_v3_1[1], seq_v3_2[1], seq_v3_3[1], seq_v4_1[1])], 1, function(x) length(which(x == 0)))
for (i in 2:length(seq_vnum)) {
  z_ceros <- cbind(z_ceros, apply(B[, c(seq_v3_1[i], seq_v3_2[i], seq_v3_3[i], seq_v4_1[i])], 1, function(x) length(which(x == 0))))
}
z_ceros <- data.frame(z_ceros)
names(z_ceros) <- paste0("Desecho_", 1:40)
# z_ceros es una matriz de 40 filas (una por cada residuo / desecho), donde cada fila hace un recuento de la cantidad de ceros que tiene una empresa para la columna dada (desecho dado), únicamente entre las variables de las columnas 3.1, 3.2, 3.3 y 4.1.

etiqueta <- paste0("e", 1698:1737) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                         "', 'sapply(z_ceros[, ", k, "], function(x) x %in% 1:3)', 'z_condic[, ",
                                                         k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_161 hasta val_10_3_200. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v1, seq_v2_1, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1)
rm(z_1, z_2_1, z_2_2, z_condic, z_ceros, etiqueta, comando)

#---------------------------------------------------------
# REGLAS val_10_3_201...val_10_3_240
#---------------------------------------------------------
# Si en esta variable (Col. 1) se respondió que "Sí" y se conoce la unidad (Columna 2.1) y la cantidad Columna (2.2) del residuo, entonces en las columnas 3.1, 3.2, 3.3 y 4.1 debe ingresarse valores mayores que 0 únicamente para la(s) gestión(es) realizada(s), y missing (blanco) para las gestiones no realizadas.

# NOTA: Bajo la condición habilitante, se deberá revisar que la suma de las columnas 3.1, 3.2, 3.3 y 4.1 sea mayor que cero, pero que ninguna de estas columnas sea nula (sí se admiten "missing").

seq_vnum <- num_var + 2
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna 2.1. Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v1 <- paste0("v", seq_vnum - 2) # Vector cadena con la secuencia de variables columna 1.
seq_v2_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 2.1.
seq_v2_2 <- paste0("v", seq_vnum + 1) # Vector cadena con la secuencia de variables columna 2.2.
seq_v3_1 <- paste0("v", seq_vnum + 3) # Vector cadena con la secuencia de variables columna 3.1.
seq_v3_2 <- paste0("v", seq_vnum + 5) # Vector cadena con la secuencia de variables columna 3.2.
seq_v3_3 <- paste0("v", seq_vnum + 7) # Vector cadena con la secuencia de variables columna 3.3.
seq_v4_1 <- paste0("v", seq_vnum + 9) # Vector cadena con la secuencia de variables columna 4.1.
```

```
data_2_2 <- data.frame(apply(B[, seq_v2_2], 2, function(x) x))
# El frame data_2_2 contiene en sus columnas copias de las variables de la columna 2.2.

z_1 <- data.frame(apply(B[, seq_v1], 2, function(x) x == 1))
# El frame z_1 contiene en sus columnas los filtros de unicidad para las variables de la columna pivote de generación de residuos / desechos (col 1).

z_2_1 <- data.frame(apply(B[, seq_v2_1], 2, function(x) x > 0))
# El frame z_2_1 contiene en sus columnas los filtros de positividad para las variables de la columna 2.1.

z_2_2 <- data.frame(apply(B[, seq_v2_2], 2, function(x) x > 0))
# El frame z_2_1 contiene en sus columnas los filtros de positividad para las variables de la columna 2.2.

z_condic <- data.frame(z_1 & z_2_1 & z_2_2)
# "z_condic" es la matriz lógica que marca con TRUE a las variables de la columna 2.1 de todas las Tablas de desechos / residuos que cumplen con la condición de tener valores positivos en las columnas 2.1 y 2.2.

z_suma <- apply(B[, c(seq_v3_1[1], seq_v3_2[1], seq_v3_3[1], seq_v4_1[1])], 1, function(x) sum(x, na.rm=T))
for (j in 2:length(seq_vnum)) {
  z_suma <- cbind(z_suma, apply(B[, c(seq_v3_1[j], seq_v3_2[j], seq_v3_3[j], seq_v4_1[j])], 1, function(x) sum(x, na.rm=T)))
}
z_suma <- data.frame(z_suma)
names(z_suma) <- paste0("Desecho_", 1:40)
# z_suma es una matriz de 40 filas (una por cada residuo / desecho), donde cada fila es igual a la suma de las columnas 3.1, 3.2, 3.3 y 4.1.

z_ceros <- apply(B[, c(seq_v3_1[1], seq_v3_2[1], seq_v3_3[1], seq_v4_1[1])], 1, function(x) length(which(is.na(x) | x == 0)))
for (i in 2:length(seq_vnum)) {
  z_ceros <- cbind(z_ceros, apply(B[, c(seq_v3_1[i], seq_v3_2[i], seq_v3_3[i], seq_v4_1[i])], 1, function(x) length(which(is.na(x) | x == 0))))
}
z_ceros <- data.frame(z_ceros)
names(z_ceros) <- paste0("Desecho_", 1:40)
# z_ceros es una matriz de 40 filas (una por cada residuo / desecho), donde cada fila hace un recuento de la cantidad de valores iguales a cero o missing que tiene una empresa entre las variables de las columnas 3.1, 3.2, 3.3 y 4.1.

z_regla <- data.frame(sapply(1:length(seq_vnum), function(j) (z_ceros[, j] %in% 0:3 & z_suma[, j] == data_2_2[, j])))
# z_regla es una matriz de 40 filas (una por cada residuo / desecho), donde cada fila hace una comprobación si el recuento de ceros entre las columnas 3.1, 3.2, 3.3 y 4.1 está entre 0 y 3, al tiempo que la suma de las columnas 3.1, 3.2, 3.3 y 4.1 es igual a la columna 2.2.

etiqueta <- paste0("e", 1738:1777) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                          "', 'z_regla[, ", k, "]==TRUE', 'z_condic[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_201 hasta val_10_3_240. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(i, j, seq_vnum, seq_v1, seq_v2_1, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1)
rm(data_2_2, z_1, z_2_1, z_2_2, z_condic, z_suma, z_ceros, z_regla, etiqueta, comando)

#---------------------------------------------------------
# REGLAS val_10_3_241...val_10_3_280
#---------------------------------------------------------
# Si esta variable (Col. 4.1) es >= 0, entonces en las columnas 8.1, 8.2 y 8.3 debe haber valores enteros entre 1 y 100 para las gestiones externas realizadas. Registrar missing para las gestiones no realizadas. La suma de esas columnas debe ser 100%.

# NOTA: La condición habilitante para esta regla (esto es, para que haya valores válidos en las columnas 8.1, 8.2 y 8.3) es que la columna 4.1 (Gestión Externa) sea mayor o igual que cero. El valor "cero" en la columna 4.1 significa que hay gestión externa, pero no se conoce la cantidad de residuo gestionada externamente.

seq_vnum <- num_var + 11
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna 2.1. Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.
```

```
seq_v4_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 4.1.
seq_v8_1 <- paste0("v", seq_vnum + 6) # Vector cadena con la secuencia de variables columna 8.1.
seq_v8_2 <- paste0("v", seq_vnum + 7) # Vector cadena con la secuencia de variables columna 8.2.
seq_v8_3 <- paste0("v", seq_vnum + 8) # Vector cadena con la secuencia de variables columna 8.3.

z_4_1 <- data.frame(apply(B[, seq_v4_1], 2, function(x) x >= 0))
# El frame z_4_1 contiene en sus columnas los filtros de positividad o nulidad para las variables de la columna 4.1.

z_8_1 <- data.frame(apply(B[, seq_v8_1], 2, function(x) x %in% 1:100 | is.na(x)))
# El frame z_8_1 contiene en sus columnas los filtros de no nulidad o vaciedad para las variables de la columna 8.1.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) x %in% 1:100 | is.na(x)))
# El frame z_8_2 contiene en sus columnas los filtros de no nulidad o vaciedad para las variables de la columna 8.2.
z_8_3 <- data.frame(apply(B[, seq_v8_3], 2, function(x) x %in% 1:100 | is.na(x)))
# El frame z_8_3 contiene en sus columnas los filtros de no nulidad o vaciedad para las variables de la columna 8.3.

z_suma <- apply(B[, c(seq_v8_1[1], seq_v8_2[1], seq_v8_3[1])], 1, sum, na.rm=T)
for (i in 2:length(seq_vnum)) {
  z_suma <- cbind(z_suma, apply(B[, c(seq_v8_1[i], seq_v8_2[i], seq_v8_3[i])], 1, sum, na.rm=T))
}
z_suma <- data.frame(z_suma)
names(z_suma) <- seq_v4_1
# z_suma es una lista de 40 componentes (una por cada residuo / desecho), donde cada componente encapsula a un vector de tamaño igual a las filas de la BDD a validar. Esta componente se  corresponde con la suma de las variables de las columnas 8.1, 8.2 y 8.3 para el residuo de la componente que corresponda en la lista "z_suma".

z_regla <- data.frame(z_suma) == 100
colnames(z_regla) <- seq_v4_1
z_regla <- data.frame(z_regla)
# z_regla es el frame que contiene los vectores lógicos (una columna por cada residuo) cuyos valores cumplen con la condición que define la regla de validación.

etiqueta <- paste0("e", 1778:1817) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                         "', 'z_regla[, ", k, "]==TRUE', 'z_4_1[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_241 hasta val_10_3_280. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v4_1, seq_v8_1, seq_v8_2, seq_v8_3, z_4_1, z_8_1, z_8_2, z_8_3)
rm(i, z_suma, z_regla, etiqueta, comando)

#------------------------------------------------------------
# REGLAS val_10_3_281...val_10_3_320
#------------------------------------------------------------
# Si esta variable (col. 4.1) está vacía (missing), debe bloquearse las columnas 8.1, 8.2, 8.3 y 9.

seq_vnum <- num_var + 11
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (4.1). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v4_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 4.1.
seq_v8_1 <- paste0("v", seq_vnum + 6) # Vector cadena con la secuencia de variables columna 8.1.
seq_v8_2 <- paste0("v", seq_vnum + 7) # Vector cadena con la secuencia de variables columna 8.2.
seq_v8_3 <- paste0("v", seq_vnum + 8) # Vector cadena con la secuencia de variables columna 8.3.
seq_v9 <- paste0("v", seq_vnum + 9) # Vector cadena con la secuencia de variables columna 9.

z_4_1 <- data.frame(apply(B[, seq_v4_1], 2, is.na))
# El frame z_4_1 contiene en sus columnas los filtros de vaciedad para las variables de la columna 4.1.

z_8_1 <- data.frame(apply(B[, seq_v8_1], 2, function(x) (is.na(x) | x == "")))
# El frame z_8_1 contiene en sus columnas los filtros de nulidad o vaciedad para las variables de la columna 8.1.

z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) (is.na(x) | x == "")))
# El frame z_8_2 contiene en sus columnas los filtros de nulidad o vaciedad para las variables de la columna 8.2.

z_8_3 <- data.frame(apply(B[, seq_v8_3], 2, function(x) (is.na(x) | x == "")))
```

```
# El frame z_8_3 contiene en sus columnas los filtros de nulidad o vaciedad para las variables de la columna 8.3.

z_9 <- data.frame(apply(B[, seq_v9], 2, function(x) (is.na(x) | x == "")))
# El frame z_9 contiene en sus columnas los filtros de vaciedad para las variables de la columna 9.

z_regla <- data.frame(z_8_1 & z_8_2 & z_8_3)
# z_regla es el frame que contiene los vectores lógicos (una columna por cada residuo) cuyos valores cumplen con la condición que define la regla de validación.

etiqueta <- paste0("e", 1818:1857) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                         "', 'z_regla[, ", k, "]==TRUE', 'z_4_1[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_281 hasta val_10_3_320. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v4_1, seq_v8_1, seq_v8_2, seq_v8_3, seq_v9, z_4_1, z_8_1, z_8_2, z_8_3, z_9)
rm(z_regla, etiqueta, comando)
#---------------------------------------------------------
# REGLAS val_10_3_321...val_10_3_360
#---------------------------------------------------------
# Si esta variable es > 0 (Col. 8.2), entonces verificar que las columnas 2.2 y 4.1 sean mayores o iguales que 0.

seq_vnum <- num_var + 18
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (8.2). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v2_2 <- paste0("v", seq_vnum - 15) # Vector cadena con la secuencia de variables columna 2.2.
seq_v4_1 <- paste0("v", seq_vnum - 7) # Vector cadena con la secuencia de variables columna 4.1.
seq_v8_2 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 8.2.

# El frame z_2_2 contiene en sus columnas los filtros de positividad para las variables de la columna 2.2.
z_2_2 <- data.frame(apply(B[, seq_v2_2], 2, function(x) x >= 0))

# El frame z_4_1 contiene en sus columnas los filtros de positividad para las variables de la columna 4.1.
z_4_1 <- data.frame(apply(B[, seq_v4_1], 2, function(x) x >= 0))

# El frame z_8_2 contiene en sus columnas los filtros de positividad para las variables de la columna 8.2.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) as.numeric(x) > 0))

# z_regla es el frame que contiene los vectores lógicos (una columna por cada residuo) cuyos valores cumplen con la condición que define la regla de validación.
z_regla <- data.frame(z_2_2 & z_4_1)

etiqueta <- paste0("e", 1858:1897) # Vector con etiquetas de las reglas de validación actuales.

comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                         "', 'z_regla[, ", k, "]==TRUE', 'z_8_2[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_321 hasta val_10_3_360. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v2_1, seq_v2_2, seq_v4_1, seq_v8_2, z_2_1, z_2_2, z_4_1, z_8_2)
rm(z_regla, etiqueta, comando)

#---------------------------------------------------------
# REGLAS val_10_3_361...val_10_3_400
#---------------------------------------------------------
# Si esta variable es > 0 (Col. 8.3), entonces verificar que la variable de la columna 9 no es un texto vacío. La columna (9) (Observación: Especifique otro tipo de recolector) se llena cuando en la columna (8.3) (Otro) se ha respondido con un porcentaje mayor que 0%.

seq_vnum <- num_var + 20
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (9). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.
```

```
seq_v8_3 <- paste0("v", seq_vnum - 1) # Vector cadena con la secuencia de variables columna 8.3.
seq_v9 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 9.

z_8_3 <- data.frame(apply(B[, seq_v8_3], 2, function(x) as.numeric(x) > 0))
# El frame z_8_3 contiene en sus columnas los filtros de positividad para las variables de la columna 8.3.

z_9 <- data.frame(apply(B[, seq_v9], 2, function(x) nchar(trimws(as.character(x))) > 2))
# El frame z_9 contiene en sus columnas los filtros de no vaciedad para las variables de la columna 9.

etiqueta <- paste0("e", 1898:1937) # Vector con etiquetas de las reglas de validación actuales.
comando <- sapply(1:length(etiqueta), function(k) paste0("G <- AddRule('", etiqueta[k],
                                                         "', 'z_9[, ", k, "]==TRUE', 'z_8_3[, ", k, "]==TRUE')"))
eval(parse(text = comando)) # Ejecución de todas y c/u de las reglas de validación, desde val_10_3_361 hasta val_10_3_400. Se agregarán al dataframe de reporte únicamente las reglas que presenten al menos una empresa que no cumpla con la condición de validación.

rm(seq_vnum, seq_v8_3, seq_v9, z_8_3, z_9, etiqueta, comando)

#---------------------------------------------------
# REGLAS val_10_3_401...val_10_3_403
#---------------------------------------------------
# Si en el Capítulo 10, Sección III (Otros residuos y/o desechos), Preguntas 1.1, 2.1 y 3.1 se respondió que "Sí", verificar que en las preguntas 1.1.2, 2.1.2 y 3.1.2 existan valores mayores que cero, según corresponda.

G <- AddRule("e1938", "B$v10076_1 > 0", "B$v10iii1_1 == 1")
G <- AddRule("e1939", "B$v10076_2 > 0", "B$v10iii2_1 == 1")
G <- AddRule("e1940", "B$v10076_3 > 0", "B$v10iii3_1 == 1")

#---------------------------------------------------
# REGLAS val_10_3_404...val_10_3_406
#---------------------------------------------------
# Si en el Capítulo 10, Sección III (Otros residuos y/o desechos), Preguntas 1.1, 2.1 y 3.1 se respondió que "No", pasar a las preguntas 1.2, 2.2 y 3.2, respectivamente.

G <- AddRule("e1941", "B$v10076_1 == 0 | is.na(B$v10076_1)", "B$v10iii1_1 == 2")
G <- AddRule("e1942", "B$v10076_2 == 0 | is.na(B$v10076_2)", "B$v10iii2_1 == 2")
G <- AddRule("e1943", "B$v10076_3 == 0 | is.na(B$v10076_3)", "B$v10iii3_1 == 2")

#---------------------------------------------------
# REGLAS val_10_3_407...val_10_3_409
#---------------------------------------------------
# Si en el Capítulo 10, Sección III (Otros residuos y/o desechos), Preguntas 1.2, 2.2 y 3.2 se respondió que "Sí", verificar que en las preguntas 1.2.1, 2.2.1 y 3.2.1 existan valores mayor que cero, según corresponda.

G <- AddRule("e1944", "B$v10078_1 > 0", "B$v10iii1_2 == 1")
G <- AddRule("e1945", "B$v10078_2 > 0", "B$v10iii2_2 == 1")
G <- AddRule("e1946", "B$v10078_3 > 0", "B$v10iii3_2 == 1")

#---------------------------------------------------
# REGLAS val_10_3_410...val_10_3_412
#---------------------------------------------------
# Si en el Capítulo 10, Sección III (Otros residuos y/o desechos), Preguntas 1.2, 2.2 y 3.2
# se respondió que "No", continuar el flujo de preguntas.

G <- AddRule("e1947", "B$v10078_1 == 0 | is.na(B$v10078_1)", "B$v10iii1_2 == 2")
G <- AddRule("e1948", "B$v10078_2 == 0 | is.na(B$v10078_2)", "B$v10iii2_2 == 2")
G <- AddRule("e1949", "B$v10078_3 == 0 | is.na(B$v10078_3)", "B$v10iii3_2 == 2")

#-------------------------------
# REGLA val_10_3_413
#-------------------------------
# Si en el Capítulo 10, Sección III (Otros residuos y/o desechos), al menos una de las
# variables v10076_1, v10076_2, v10076_3 es mayor que cero, entonces VAR_8091 > 0.

G <- AddRule("e1950", "B$v8091 > 0", "B$v10076_1 > 0 | B$v10076_2 > 0 | B$v10076_3 > 0")
```

```
#---------------------------------
# REGLA val_10_3_414
#---------------------------------
# La suma de las variables v10076_1, v10076_2, v10076_3 no puede ser mayor que VAR_8091 > 0.

z <- apply(cbind(B$v10076_1, B$v10076_2, B$v10076_3), 1, sum, na.rm = T)
G <- AddRule("e1951", "z <= B$v8091", "B$v8091 > 0")
rm(z)

#---------------------------------
# REGLA val_10_3_415
#---------------------------------
# En el Capítulo 10, Sección III, Pregunta 4, es Obligatorio el llenado. En caso de contestar "Sí", ir a la Pregunta 4.1. En el caso de contestar "No", pasar al siguiente Capítulo.

G <- AddRule("e1952", "!(B$v10iii4 == 2) | (is.na(B$v10iii41) & is.na(B$v10iii42))")

#---------------------------------
# REGLA val_10_3_416
#---------------------------------
# Para la Pregunta 4.1, obligatorio su llenado si respondió que "Sí" en la Pregunta 4. En el caso de contestar "Sí", ir a la Pregunta 4.2. En el caso de contestar "No", pasar al capítulo siguiente.

G <- AddRule("e1953", "!(B$v10iii41 == 2) | is.na(B$v10iii42)", "B$v10iii4 == 1")

#---------------------------------
# REGLA val_10_3_417
#---------------------------------
# Para la Pregunta 4.2, obligatorio su llenado si respondió que "Sí" en la Pregunta 4.1.

G <- AddRule("e1954", "B$v10iii42 > 0", "B$v10iii41 == 1")

#---------------------------------
# REGLA val_10_3_418
#---------------------------------
# El valor de la variable de la Pregunta 4.2 debe ser menor que la v3181.

G <- AddRule("e1955", "B$v10iii42 < B$v3181", "B$v10iii41 == 1")

#---------------------------------
# REGLA val_10_3_419
#---------------------------------
# Si existe valor > 0 en la Col. 8.2 de la Tabla 10.III.1, debe existir valor > 0 en la variable v10076_1.

seq_vnum <- num_var[1:12] + 18
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (8.2). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v8_2 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 8.2, para la Tabla 10.III.1.

# El frame z_8_2 contiene en sus columnas valores TRUE si la respectiva variable de la columna 8.2 es positiva.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) x > 0))

# El frame Posit_8_2 contiene, por cada empresa, el valor lógico TRUE si al menos una de las variables de la columna 8.2 es positiva.
Posit_8_2 <- apply(B[, seq_v8_2], 1, function(x) any(x))

G <- AddRule("e1956", "B$v10076_1 > 0", "Posit_8_2==TRUE")

rm(seq_vnum, seq_v8_2, z_8_2, Posit_8_2)

#---------------------------------
# REGLA val_10_3_420
#---------------------------------
# Si existe valor > 0 en la Col. 8.2 de la Tabla 10.III.2, debe existir valor > 0 en la variable v10076_2.

seq_vnum <- num_var[13:18] + 18
```

```
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (8.2). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v8_2 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 8.2, para la Tabla 10.III.2.

# El frame z_8_2 contiene en sus columnas valores TRUE si la respectiva variable de la columna 8.2 es positiva.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) x > 0))

# El frame Posit_8_2 contiene, por cada empresa, el valor lógico TRUE si al menos una de las variables de la columna 8.2 es positiva.
Posit_8_2 <- apply(B[, seq_v8_2], 1, function(x) any(x))

G <- AddRule("e1957", "B$v10076_2 > 0", "Posit_8_2==TRUE")

rm(seq_vnum, seq_v8_2, z_8_2, Posit_8_2)

#------------------------------
# REGLA val_10_3_421
#------------------------------
# Si existe valor > 0 en la Col. 8.2 de la Tabla 10.III.3, debe existir valor > 0 en la variable v10076_3.

seq_vnum <- num_var[19:40] + 18
# La variable seq_vnum contiene los sufijos numéricos de las variables pivote en la validación, esto es, las variables de la columna (8.2). Su dimensión es igual al recuento de residuos de cualquier tipo del formulario.

seq_v8_2 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables columna 8.2, para la Tabla 10.III.3.

# El frame z_8_2 contiene en sus columnas valores TRUE si la respectiva variable de la columna 8.2 es positiva.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) x > 0))

# El frame Posit_8_2 contiene, por cada empresa, el valor lógico TRUE si al menos una de las variables de la columna 8.2 es positiva.
Posit_8_2 <- apply(B[, seq_v8_2], 1, function(x) any(x))

G <- AddRule("e1958", "B$v10076_3 > 0", "Posit_8_2==TRUE")

rm(seq_vnum, seq_v8_2, z_8_2, Posit_8_2)


################################################################################
# [BLOQUE 5]. RUTINA DE EXPORTACIÓN DE LOS ERRORES DE VALIDACIÓN AL TIPO DE
# OBJETOS QUE GENERA DECON PARA LAS VALIDACIONES ECONÓMICAS.
################################################################################

vect_log <- (B$estado == 'C' & B$efectividad == 1)
vect_log <- (vect_log & absorbida != '1.8')
vect_log <- (vect_log & desintegracion != '1.9')
vect_log <- (vect_log & escision != '1.10')

for (i in 1:length(vect_log)) {
  if (!isFALSE(vect_log[i])) vect_log[i] <- TRUE
}

N <- length(which(vect_log == TRUE))
if (N > 0) {
  G <- data.frame(G[which(vect_log == TRUE), ])

  if (dim(G)[2] == 1) {
    nombre_columna <- "e1999"
    alerta_1999 <- data.frame(cbind(G[1, 1], "e1999"))
    names(alerta_1999)[1] <- "inec_identificador_empresa"
    names(alerta_1999)[2] <- "error_n"
    variables_juntura <- aux_variables_alertas[-which("error_n" == aux_variables_alertas)]
    alerta_1999 <- merge(B[, variables_juntura], alerta_1999, by = "inec_identificador_empresa")
    alerta_1999 <- alerta_1999[, aux_variables_alertas]
  } else {
    err_x_var <- apply(G[, 2:dim(G)[2]], 2, function(x) sum(x, na.rm = T))
```

```
    col_a_retener <- names(sort(err_x_var[err_x_var > 0]))
    G <- data.frame(cbind(G$inec_identificador_empresa, G[, col_a_retener]))
    names(G) <- c("inec_identificador_empresa", col_a_retener)
    rm(err_x_var, col_a_retener)

    G[, 2:dim(G)[2]] <- sapply(G[, 2:dim(G)[2]], as.character) # Conversión forzosa de todas las columnas de errores a tipo
'character'.

    for (j in 2:dim(G)[2]) G[, j] <- ifelse(G[, j] == "1", gsub("e", "", names(G)[j]), NA_character_)
    # Transformación de los 1's de las columnas de G al sufijo numérico del nombre de la columna que contiene los 1's.

    vars <- names(G)[2:dim(G)[2]] # Vector con todos los nombres de variables con error.
    sec_alertas <- sapply(vars, function(x) as.numeric(sub("e", "", x))) # Versión numérica de "vars".

    variables_juntura <- aux_variables_alertas[-which("error_n" == aux_variables_alertas)]
    # Vector con todas las variables de reporte, excepto "error_n", la cual se construirá en el código DECON de construcción del
reporte general (Económico + Ambiental).

    rm(j, vars_AMB)  # Borrado de algunas variables temporales.

    e_AMB <- environment()
    # Puntero al entorno local de ejecución (el que contiene todos los objetos intermedios del proceso de validación estándar).
Servirá como entorno de ejecución de las instrucciones incluidas en la siguiente función 'Escribir_Alerta()'.

    # Definición de función de escritura de frames de tipo 'alerta_N'.
    Escribir_Alerta <- function(num_alerta) {
      nombre_columna <- paste0("e", num_alerta)
      comando <- paste0("alerta_", num_alerta, " <- G[which(!is.na(G[, '", nombre_columna,
                        "'])), c('inec_identificador_empresa', '", nombre_columna, "')]")
      eval(parse(text = comando), envir = e_AMB)

      comando <- paste0("names(alerta_", num_alerta, ")[2] <- 'error_n'")
      eval(parse(text = comando), envir = e_AMB)

      comando <- paste0("alerta_", num_alerta," <- merge(B[, variables_juntura], alerta_",
                        num_alerta,", by = 'inec_identificador_empresa')")
      eval(parse(text = comando), envir = e_AMB)

      comando <- paste0("alerta_", num_alerta, " <- alerta_", num_alerta, "[, aux_variables_alertas]")
      eval(parse(text = comando), envir = e_AMB)
    }

    sapply(sec_alertas, function(x) Escribir_Alerta(x)) # Llamada a creación de todos los frames de tipo "alerta_N" para todos los
errores ambientales existentes en el frame G.

    rm(vars, sec_alertas, variables_juntura, e_AMB, G, vect_log)
    # Borrado del frame de generación inicial de errores ambientales y objetos temporales.
  }
} else {
  print(paste0("Para el corte actual de la BDD, la zonal ", aux_seleccion, " no tiene empresas para validar."))
}

##################################################################################################
# CIERRE DEL ENTORNO DE DATOS Y FINALIZACION DE LAS VALIDACIONES AMBIENTALES ESTÁNDAR
##################################################################################################
```

ANEXO B. Código R Markdown y Código SPSS para el control de integridad de la base de datos.

```
---
title: "SX Integridad e Imputación 2023"
author: "Ramiro Benavides"
date: "2024-02-23"
output: html_document
---

1. Imputación de casos particulares de variables corregidas en ciertas empresas.
```{r}
# B$inec_identificador_empresa <- B$id_empresa
# B$id <- as.character(B$inec_identificador_empresa)
# B_BK <- B
B$v8099[which(B$id == '13705179098')] <- 110533
B$v8099[which(B$id == '13826254178')] <- 25200
B$v8099[which(B$id == '13602946015')] <- 1000
B$v8099[which(B$id == '13602124015')] <- 52000
B$v8099[which(B$id == '13602997019')] <- 1000
B$v8099[which(B$id == '13826010171')] <- 1804
B$v8099[which(B$id == '14783702178')] <- 21535
B$v8099[which(B$id == '14793919175')] <- 153885
B$v8099[which(B$id == '13602122012')] <- 800
B$v8099[which(B$id == '14773721174')] <- 19270
B$v8099[which(B$id == '14596814175')] <- 1942187
B$v8099[which(B$id == '14713026078')] <- 5300
B$v8099[which(B$id == '44381467012')] <- 2748

#-----------------------------------------------------

# B$v9i2[which(B$id == '44528781176')] <- 1
# B$v9036[which(B$id == '44528781176')] <- 1
# B$v9037[which(B$id == '44528781176')] <- 635180
# B$v9038[which(B$id == '44528781176')] <- 127036
# B$v9039[which(B$id == '44528781176')] <- 635180
# B$v9052[which(B$id == '44528781176')] <- 635180
# B$v9053[which(B$id == '44528781176')] <- 127036
# B$v9054[which(B$id == '44528781176')] <- 635180
#
# B$v9i2[which(B$id == '14658916174')] <- 1
# B$v9028[which(B$id == '14658916174')] <- 1
# B$v9029[which(B$id == '14658916174')] <- 19174
# B$v9030[which(B$id == '14658916174')] <- 959
# B$v9031[which(B$id == '14658916174')] <- 19174
# B$v9052[which(B$id == '14658916174')] <- 19174
# B$v9053[which(B$id == '14658916174')] <- 959
# B$v9054[which(B$id == '14658916174')] <- 19174
#
# B$v9i2[which(B$id == '14633469173')] <- 1
# B$v9036[which(B$id == '14633469173')] <- 1
# B$v9037[which(B$id == '14633469173')] <- 102583
# B$v9038[which(B$id == '14633469173')] <- 25646
# B$v9039[which(B$id == '14633469173')] <- 102583
# B$v9052[which(B$id == '14633469173')] <- 102583
# B$v9053[which(B$id == '14633469173')] <- 25646
# B$v9054[which(B$id == '14633469173')] <- 102583
#
# B$v9i2[which(B$id == '13829624171')] <- 1
# B$v9036[which(B$id == '13829624171')] <- 1
# B$v9037[which(B$id == '13829624171')] <- 11347
# B$v9038[which(B$id == '13829624171')] <- 227
# B$v9039[which(B$id == '13829624171')] <- 11347
# B$v9052[which(B$id == '13829624171')] <- 11347
# B$v9053[which(B$id == '13829624171')] <- 227
# B$v9052[which(B$id == '13829624171')] <- 11347
```
```

```
# B$v9i2[which(B$id == '13826419177')] <- 1
# B$v9036[which(B$id == '13826419177')] <- 1
# B$v9037[which(B$id == '13826419177')] <- 49050
# B$v9038[which(B$id == '13826419177')] <- 9810
# B$v9039[which(B$id == '13826419177')] <- 49050
# B$v9052[which(B$id == '13826419177')] <- 49050
# B$v9053[which(B$id == '13826419177')] <- 9810
# B$v9052[which(B$id == '13826419177')] <- 49050
#
# B$v9i2[which(B$id == '13824602179')] <- 1
# B$v9036[which(B$id == '13824602179')] <- 1
# B$v9037[which(B$id == '13824602179')] <- 148492
# B$v9038[which(B$id == '13824602179')] <- 29700
# B$v9039[which(B$id == '13824602179')] <- 148492
# B$v9052[which(B$id == '13824602179')] <- 148492
# B$v9053[which(B$id == '13824602179')] <- 29700
# B$v9054[which(B$id == '13824602179')] <- 148492
#
# B$v9i2[which(B$id == '13710735098')] <- 1
# B$v9036[which(B$id == '13710735098')] <- 1
# B$v9037[which(B$id == '13710735098')] <- 25000
# B$v9038[which(B$id == '13710735098')] <- 6250
# B$v9039[which(B$id == '13710735098')] <- 25000
# B$v9052[which(B$id == '13710735098')] <- 25000
# B$v9053[which(B$id == '13710735098')] <- 6250
# B$v9054[which(B$id == '13710735098')] <- 25000
#
# B$v9i2[which(B$id == '13708424093')] <- 1
# B$v9036[which(B$id == '13708424093')] <- 1
# B$v9037[which(B$id == '13708424093')] <- 48500
# B$v9038[which(B$id == '13708424093')] <- 12125
# B$v9039[which(B$id == '13708424093')] <- 48500
# B$v9052[which(B$id == '13708424093')] <- 48500
# B$v9053[which(B$id == '13708424093')] <- 12125
# B$v9054[which(B$id == '13708424093')] <- 48500
#
# B$v9i2[which(B$id == '13602237014')] <- 1
# B$v9036[which(B$id == '13602237014')] <- 1
# B$v9037[which(B$id == '13602237014')] <- 14867823
# B$v9038[which(B$id == '13602237014')] <- 2973565
# B$v9039[which(B$id == '13602237014')] <- 14867823
# B$v9052[which(B$id == '13602237014')] <- 14867823
# B$v9053[which(B$id == '13602237014')] <- 2973565
# B$v9054[which(B$id == '13602237014')] <- 14867823

# B$v9i2[which(B$id == '13824570170')] <- 1
# B$v9029[which(B$id == '13824570170')] <- 7756883
# B$v9030[which(B$id == '13824570170')] <- 1318670
# B$v9033[which(B$id == '13824570170')] <- 7756883
# B$v9034[which(B$id == '13824570170')] <- 1939221
# B$v9052[which(B$id == '13824570170')] <- 7756883
# B$v9053[which(B$id == '13824570170')] <- 1318670
# B$v9055[which(B$id == '13824570170')] <- 7756883
# B$v9056[which(B$id == '13824570170')] <- 1939221
#
# B$v9i2[which(B$id == '13824498177')] <- 1
# B$v9037[which(B$id == '13824498177')] <- 496689825
# B$v9038[which(B$id == '13824498177')] <- 74503474
# B$v9039[which(B$id == '13824498177')] <- 496689825
# B$v9052[which(B$id == '13824498177')] <- 496689825
# B$v9053[which(B$id == '13824498177')] <- 74503474
# B$v9054[which(B$id == '13824498177')] <- 496689825
#
# B$v9i2[which(B$id == '14804557076')] <- 1
# B$v9029[which(B$id == '14804557076')] <- 257000
# B$v9030[which(B$id == '14804557076')] <- 25700
# B$v9031[which(B$id == '14804557076')] <- 257000
```

```
# B$v9052[which(B$id == '14804557076')] <- 257000
# B$v9053[which(B$id == '14804557076')] <- 25700
# B$v9054[which(B$id == '14804557076')] <- 257000
#
# B$v9i2[which(B$id == '14654385174')] <- 1
# B$v9037[which(B$id == '14654385174')] <- 10032
# B$v9038[which(B$id == '14654385174')] <- 1204
# B$v9039[which(B$id == '14654385174')] <- 10032
# B$v9052[which(B$id == '14654385174')] <- 10032
# B$v9053[which(B$id == '14654385174')] <- 1204
# B$v9054[which(B$id == '14654385174')] <- 10032

# B$v9i2[which(B$id == '14749765179')] <- 1
# B$v9036[which(B$id == '14749765179')] <- 1
# B$v9037[which(B$id == '14749765179')] <- 13404
# B$v9038[which(B$id == '14749765179')] <- 2681
# B$v9039[which(B$id == '14749765179')] <- 13404
# B$v9052[which(B$id == '14749765179')] <- 13404
# B$v9053[which(B$id == '14749765179')] <- 2681
# B$v9054[which(B$id == '14749765179')] <- 13404
#
# B$v9i2[which(B$id == '14768927091')] <- 1
# B$v9036[which(B$id == '14768927091')] <- 1
# B$v9037[which(B$id == '14768927091')] <- 305800
# B$v9038[which(B$id == '14768927091')] <- 45870
# B$v9039[which(B$id == '14768927091')] <- 305800
# B$v9052[which(B$id == '14768927091')] <- 305800
# B$v9053[which(B$id == '14768927091')] <- 45870
# B$v9054[which(B$id == '14768927091')] <- 305800
#
# B$v9i2[which(B$id == '14800457174')] <- 1
# B$v9036[which(B$id == '14800457174')] <- 1
# B$v9037[which(B$id == '14800457174')] <- 581594
# B$v9038[which(B$id == '14800457174')] <- 98871
# B$v9039[which(B$id == '14800457174')] <- 581594
# B$v9052[which(B$id == '14800457174')] <- 581594
# B$v9053[which(B$id == '14800457174')] <- 98871
# B$v9054[which(B$id == '14800457174')] <- 581594

#------------------------------------------------------------------------

# B$v9037[which(B$id == '13627012050')] <- 14540
# B$v9038[which(B$id == '13627012050')] <- 85000
# B$v9039[which(B$id == '13627012050')] <- 14540
# B$v9052[which(B$id == '13627012050')] <- 14540
# B$v9053[which(B$id == '13627012050')] <- 85000
# B$v9054[which(B$id == '13627012050')] <- 14540

# B$v9037[which(B$id == '13824726170')] <- 19200
# B$v9038[which(B$id == '13824726170')] <- 14320
# B$v9039[which(B$id == '13824726170')] <- 19200
# B$v9052[which(B$id == '13824726170')] <- 19200
# B$v9053[which(B$id == '13824726170')] <- 14320
# B$v9054[which(B$id == '13824726170')] <- 19200

# B$v9037[which(B$id == '13826194175')] <- 1190
# B$v9038[which(B$id == '13826194175')] <- 100
# B$v9039[which(B$id == '13826194175')] <- 1190
# B$v9052[which(B$id == '13826194175')] <- 1190
# B$v9053[which(B$id == '13826194175')] <- 100
# B$v9054[which(B$id == '13826194175')] <- 1190

# B$v9037[which(B$id == '14666765177')] <- 32081280
# B$v9038[which(B$id == '14666765177')] <- 2887310
# B$v9039[which(B$id == '14666765177')] <- 32081280
# B$v9052[which(B$id == '14666765177')] <- 32081280
# B$v9053[which(B$id == '14666765177')] <- 2887310
```

```
# B$v9054[which(B$id == '14666765177')] <- 32081280

# B$v9037[which(B$id == '14772262176')] <- 16800
# B$v9038[which(B$id == '14772262176')] <- 16800
# B$v9039[which(B$id == '14772262176')] <- 16800
# B$v9052[which(B$id == '14772262176')] <- 16800
# B$v9053[which(B$id == '14772262176')] <- 16800
# B$v9054[which(B$id == '14772262176')] <- 16800

# B$v9037[which(B$id == '14773721174')] <- 1837260
# B$v9038[which(B$id == '14773721174')] <- 92000
# B$v9039[which(B$id == '14773721174')] <- 1837260
# B$v9052[which(B$id == '14773721174')] <- 1837260
# B$v9053[which(B$id == '14773721174')] <- 92000
# B$v9054[which(B$id == '14773721174')] <- 1837260

# B$v9037[which(B$id == '14820987170')] <- 920700
# B$v9038[which(B$id == '14820987170')] <- 82860
# B$v9039[which(B$id == '14820987170')] <- 920700
# B$v9052[which(B$id == '14820987170')] <- 920700
# B$v9053[which(B$id == '14820987170')] <- 82860
# B$v9054[which(B$id == '14820987170')] <- 920700

# B$v9005[which(B$id == '44403938073')] <- 1531136
# B$v9006[which(B$id == '44403938073')] <- 261366
# B$v9007[which(B$id == '44403938073')] <- 1531136
# B$v9052[which(B$id == '44403938073')] <- 1531136
# B$v9053[which(B$id == '44403938073')] <- 261366
# B$v9054[which(B$id == '44403938073')] <- 1531136

# B$v9037[which(B$id == '47031077172')] <- 11778
# B$v9038[which(B$id == '47031077172')] <- 1060
# B$v9039[which(B$id == '47031077172')] <- 11778
# B$v9052[which(B$id == '47031077172')] <- 11778
# B$v9053[which(B$id == '47031077172')] <- 1060
# B$v9054[which(B$id == '47031077172')] <- 11778

#-----------------------------------------------------------------------

B$v9078[which(B$id == '13602151012')] <- 2799093
B$v9078[which(B$id == '13602012012')] <- 1160679
B$v9078[which(B$id == '14716250173')] <- 548335
B$v9078[which(B$id == '13824893173')] <- 494584
B$v9078[which(B$id == '13824656171')] <- 105459
B$v9078[which(B$id == '13827278178')] <- 103606
B$v9078[which(B$id == '13602002017')] <- 80000
B$v9078[which(B$id == '13824482173')] <- 50932

#-----------------------------------------------------------------------

B$v10004[which(B$id == '13763386133')] <- 31330

B$v10003[which(B$id == '13705804090')] <- 2
B$v10004[which(B$id == '13705804090')] <- 531871

B$v10003[which(B$id == '13706667096')] <- 2
B$v10004[which(B$id == '13706667096')] <- 8963

B$v10003[which(B$id == '13826419177')] <- 2
B$v10004[which(B$id == '13826419177')] <- 24013

B$v10003[which(B$id == '14637707176')] <- 2
B$v10004[which(B$id == '14637707176')] <- 10440

B$v10003[which(B$id == '49380785098')] <- 2
B$v10004[which(B$id == '49380785098')] <- 10216
```

```
B$v10003[which(B$id == '14635138093')] <- 2
B$v10004[which(B$id == '14635138093')] <- 21429

B$v10003[which(B$id == '14837690090')] <- 2
B$v10004[which(B$id == '14837690090')] <- 6528656

B$v10003[which(B$id == '14873419084')] <- 1
B$v10004[which(B$id == '14873419084')] <- 64000

B$v10003[which(B$id == '46773657092')] <- 1
B$v10004[which(B$id == '46773657092')] <- 12246

#---------------------------------------------------------------------------

B$v10i3[which(B$id == '13705179098')] <- 1
B$v10006[which(B$id == '13705179098')] <- 2
B$v10014[which(B$id == '13705179098')] <- 1
B$v10022[which(B$id == '13705179098')] <- 2
B$v10016[which(B$id == '13705179098')] <- 1
B$v10017[which(B$id == '13705179098')] <- 2599200
B$v10018[which(B$id == '13705179098')] <- 24
B$v10019[which(B$id == '13705179098')] <- 30
B$v10020[which(B$id == '13705179098')] <- 1871424000
B$v10030[which(B$id == '13705179098')] <- 1871424000

#---------------------------------------------------------------------------

B$v10017[which(B$id == '13655728079')] <- 146012
B$v10020[which(B$id == '13655728079')] <- 630771840
B$v10030[which(B$id == '13655728079')] <- 630771840

#---------------------------------------------------------------------------

B$v10009[which(B$id == '14720447178')] <- 3978
B$v10012[which(B$id == '14720447178')] <- 17375904
B$v10030[which(B$id == '14720447178')] <- 18201888

#---------------------------------------------------------------------------

B$v10i3[which(B$id == '14856916171')] <- 1
B$v10006[which(B$id == '14856916171')] <- 1
B$v10008[which(B$id == '14856916171')] <- 1
B$v10009[which(B$id == '14856916171')] <- 3229
B$v10010[which(B$id == '14856916171')] <- 12
B$v10011[which(B$id == '14856916171')] <- 25
B$v10012[which(B$id == '14856916171')] <- 11624400
B$v10014[which(B$id == '14856916171')] <- 2
B$v10022[which(B$id == '14856916171')] <- 2
B$v10030[which(B$id == '14856916171')] <- 11624400

#---------------------------------------------------------------------------

B$v10014[which(B$id == '14869478173')] <- 1
B$v10016[which(B$id == '14869478173')] <- 1
B$v10017[which(B$id == '14869478173')] <- 2027
B$v10018[which(B$id == '14869478173')] <- 8
B$v10019[which(B$id == '14869478173')] <- 20
B$v10020[which(B$id == '14869478173')] <- 3891840
B$v10030[which(B$id == '14869478173')] <- 3891840

#---------------------------------------------------------------------------

B$v10014[which(B$id == '14873389096')] <- 1
B$v10016[which(B$id == '14873389096')] <- 1
B$v10017[which(B$id == '14873389096')] <- 4240
B$v10018[which(B$id == '14873389096')] <- 8
B$v10019[which(B$id == '14873389096')] <- 22
```

```
B$v10020[which(B$id == '14873389096')] <- 8954880
B$v10030[which(B$id == '14873389096')] <- 8954880

#---------------------------------------------------------------------

B$v10009[which(B$id == '13734909119')] <- 223
B$v10012[which(B$id == '13734909119')] <- 1926720
B$v10030[which(B$id == '13734909119')] <- 1926720

#---------------------------------------------------------------------

B$v10016[which(B$id == '14822828095')] <- 1
B$v10017[which(B$id == '14822828095')] <- 703.47
B$v10018[which(B$id == '14822828095')] <- 12
B$v10019[which(B$id == '14822828095')] <- 20
B$v10020[which(B$id == '14822828095')] <- 2025994
B$v10030[which(B$id == '14822828095')] <- 2025994

#---------------------------------------------------------------------

B$v10009[which(B$id == '13829533175')] <- 1383
B$v10010[which(B$id == '13829533175')] <- 24
B$v10011[which(B$id == '13829533175')] <- 20
B$v10012[which(B$id == '13829533175')] <- 7966080
B$v10030[which(B$id == '13829533175')] <- 7966080

#---------------------------------------------------------------------

B$v10009[which(B$id == '14640234173')] <- 216
B$v10010[which(B$id == '14640234173')] <- 24
B$v10011[which(B$id == '14640234173')] <- 30
B$v10012[which(B$id == '14640234173')] <- 1866240

B$v10017[which(B$id == '14640234173')] <- 72
B$v10018[which(B$id == '14640234173')] <- 24
B$v10019[which(B$id == '14640234173')] <- 30
B$v10020[which(B$id == '14640234173')] <- 622080

B$v10030[which(B$id == '14640234173')] <- 2488320

#---------------------------------------------------------------------

i <- which(B$id == '14863718172')
B$v10035[i] <- 238332
B$v10032[i] <- 238332 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13627013056')
B$v10035[i] <- 136344
B$v10032[i] <- 136344 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13706438097')
B$v10035[i] <- 309640
B$v10032[i] <- 309640 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13710879090')
B$v10035[i] <- 86400
B$v10032[i] <- 86400 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13828259177')
B$v10035[i] <- 61560
B$v10032[i] <- 61560 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14783702178')
B$v10035[i] <- 563155
B$v10032[i] <- 563155 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14829783135')
```

```
B$v10035[i] <- 18569088
B$v10032[i] <- 18569088 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14870885173')
B$v10035[i] <- 33466
B$v10032[i] <- 33466 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14867791095')
B$v10035[i] <- 8186400
B$v10032[i] <- 8186400 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13602151012')
B$v10035[i] <- 1002240
B$v10032[i] <- 1002240 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13710289098')
B$v10035[i] <- 172800
B$v10032[i] <- 172800 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '13829533175')
B$v10035[i] <- 271200
B$v10032[i] <- 271200 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14673975098')
B$v10035[i] <- 76291
B$v10032[i] <- 76291 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14785339178')
B$v10035[i] <- 11218176
B$v10032[i] <- 11218176 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '44537311174')
B$v10035[i] <- 75085
B$v10032[i] <- 75085 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '46716813177')
B$v10035[i] <- 354322
B$v10032[i] <- 354322 / 12 / B$v10033[i] / B$v10034[i]

#------------------------------------------------------------------------

B$v10128[which(B$id == '14869185133')] <- 123969
B$v10128[which(B$id == '13706047098')] <- 20422
B$v10128[which(B$id == '13711040093')] <- 69405
B$v10128[which(B$id == '13826143171')] <- 645929
B$v10128[which(B$id == '13831547175')] <- 25
B$v10128[which(B$id == '13846643189')] <- 2650
B$v10128[which(B$id == '14675257176')] <- 1572
B$v10128[which(B$id == '14873389096')] <- 9118
B$v10128[which(B$id == '44403938073')] <- 15042
B$v10128[which(B$id == '44603454135')] <- 8040
B$v10128[which(B$id == '46994508172')] <- 5800
B$v10128[which(B$id == '47050948133')] <- 19007
B$v10128[which(B$id == '22314163099')] <- 1600
B$v10128[which(B$id == '14800439176')] <- 2760
B$v10128[which(B$id == '14602560016')] <- 4808
B$v10128[which(B$id == '13705210092')] <- 4693496
B$v10128[which(B$id == '13824845179')] <- 4105
B$v10128[which(B$id == '13827316177')] <- 173
B$v10127[which(B$id == '14869592171')] <- 1
B$v10127[which(B$id == '47055666098')] <- 2
B$v10128[which(B$id == '47055666098')] <- 12
B$v10128[which(B$id == '47275990171')] <- 6776
B$v10380[which(B$id == '14865994177')] <- 34383
B$v10380[which(B$id == '14783510015')] <- 7800
B$v10379[which(B$id == '13705123092')] <- 1
B$v10380[which(B$id == '13763885130')] <- 82
```

```
#-***********************************************************

# Valores corregidos por las coordinaciones zonales

i <- which(B$id == '44437645091')
B$v10035[i] <- 25344
B$v10032[i] <- 25344 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14813284171')
B$v10035[i] <- 418
B$v10032[i] <- 418 / 12 / B$v10033[i] / B$v10034[i]

i <- which(B$id == '14832550176')
B$v10035[i] <- 250
B$v10032[i] <- 250 / 12 / B$v10033[i] / B$v10034[i]
```

2. Código de integridad para todas las variables del MIEAE 2023 (excepto 10.3 Residuos y/o desechos).

```{r}
i_inicial <- which(names(B) == "v7001")
i_final <- which(names(B) == "v10iii42")
vars <- names(B)[i_inicial:i_final]

# Capítulo 7
B$v71[is.na(B$v71)] <- 2

i <- which(B$v7002 == 0)
B$v7002[i] <- NA_integer_
B$v7003[i] <- NA_integer_
B$v7004[i] <- NA_integer_

i <- which(B$v7003 == 0)
B$v7003[i] <- NA_integer_

i <- which(B$v7004 == 0)
B$v7004[i] <- NA_integer_

B$v7005[which(is.na(B$v7003))] <- NA_integer_
B$v7006[which(is.na(B$v7004))] <- NA_integer_

# Capítulo 8
B$v8086[which(is.na(B$v8086))] <- 2
B$v8086[which(B$v8086 == 1 & is.na(B$v8087))] <- 2
B$v8088[which(is.na(B$v8088))] <- 2
B$v8090[which(is.na(B$v8090))] <- 2
B$v8092[which(is.na(B$v8092))] <- 2
B$v8092[which(B$v8092 == 1 & is.na(B$v8093))] <- 2
B$v8093[which(B$v8092 == 2 & B$v8093 == 0)] <- NA_integer_

B$v8095[which(B$v8095 == 2 & B$v8095 == 0)] <- NA_integer_

B$v8096[which(B$v8096 == 1 & is.na(B$v8097))] <- 2
B$v8097[which(B$v8096 == 2 & B$v8097 == 0)] <- NA_integer_

B$v8098[which(B$v8098 == 0)] <- NA_integer_
B$v8099[which(B$v8099 == 0)] <- NA_integer_
B$v8100[which(B$v8100 == 0)] <- NA_integer_

# Capítulo 9
B$v9001[which(B$v9001 == 0)] <- NA_integer_
B$v9002[which(B$v9002 == 0)] <- NA_integer_

B$v9i2[which(is.na(B$v9i2))] <- 2

m <- apply(cbind(B$v9004, B$v9012, B$v9020, B$v9028, B$v9036, B$v9044), 1, min)
i <- which(B$v9i2 == 1 & is.na(m))
```

```
B$v9i2[i] <- 2

B$v9005[which(B$v9005 == 0)] <- NA_integer_
B$v9006[which(B$v9006 == 0)] <- NA_integer_
B$v9007[which(B$v9007 == 0)] <- NA_integer_
B$v9009[which(B$v9009 == 0)] <- NA_integer_
B$v9010[which(B$v9010 == 0)] <- NA_integer_

B$v9013[which(B$v9013 == 0)] <- NA_integer_
B$v9014[which(B$v9014 == 0)] <- NA_integer_
B$v9015[which(B$v9015 == 0)] <- NA_integer_
B$v9017[which(B$v9017 == 0)] <- NA_integer_
B$v9018[which(B$v9018 == 0)] <- NA_integer_

B$v9021[which(B$v9021 == 0)] <- NA_integer_
B$v9022[which(B$v9022 == 0)] <- NA_integer_
B$v9023[which(B$v9023 == 0)] <- NA_integer_
B$v9025[which(B$v9025 == 0)] <- NA_integer_
B$v9026[which(B$v9026 == 0)] <- NA_integer_

B$v9029[which(B$v9029 == 0)] <- NA_integer_
B$v9030[which(B$v9030 == 0)] <- NA_integer_
B$v9031[which(B$v9031 == 0)] <- NA_integer_
B$v9033[which(B$v9033 == 0)] <- NA_integer_
B$v9034[which(B$v9034 == 0)] <- NA_integer_

B$v9037[which(B$v9037 == 0)] <- NA_integer_
B$v9038[which(B$v9038 == 0)] <- NA_integer_
B$v9039[which(B$v9039 == 0)] <- NA_integer_
B$v9041[which(B$v9041 == 0)] <- NA_integer_
B$v9042[which(B$v9042 == 0)] <- NA_integer_

B$v9045[which(B$v9045 == 0)] <- NA_integer_
B$v9046[which(B$v9046 == 0)] <- NA_integer_
B$v9047[which(B$v9047 == 0)] <- NA_integer_
B$v9049[which(B$v9049 == 0)] <- NA_integer_
B$v9050[which(B$v9050 == 0)] <- NA_integer_

B$v9052[which(B$v9052 == 0)] <- NA_integer_
B$v9053[which(B$v9053 == 0)] <- NA_integer_
B$v9054[which(B$v9054 == 0)] <- NA_integer_
B$v9055[which(B$v9055 == 0)] <- NA_integer_
B$v9056[which(B$v9056 == 0)] <- NA_integer_

i <- which(B$v9044 == 1)
B$v9036[i] <- 1
B$v9037[i] <- B$v9045[i]
B$v9038[i] <- B$v9046[i]
B$v9039[i] <- B$v9047[i]
B$v9041[i] <- B$v9049[i]
B$v9042[i] <- B$v9050[i]
B$v9051[i] <- ""

B$v9045[i] <- NA_integer_
B$v9046[i] <- NA_integer_
B$v9047[i] <- NA_integer_
B$v9049[i] <- NA_integer_
B$v9050[i] <- NA_integer_

s <- apply(cbind(B$v9058, B$v9062, B$v9066, B$v9070, B$v9074, B$v9078, B$v9082,
                 B$v9086, B$v9090, B$v9094, B$v9098, B$v9102), 1, sum, na.rm=T)
s1 <- s > 0
table(B$v9ii1, s1, useNA = "ifany")
i <- which(B$v9ii1 == 1 & !s1)
B$v9ii1[i] <- 2

B$v9058[which(B$v9058 == 0)] <- NA_integer_
```

```
B$v9059[which(B$v9059 == 0)] <- NA_integer_

B$v9062[which(B$v9062 == 0)] <- NA_integer_
B$v9063[which(B$v9063 == 0)] <- NA_integer_

B$v9066[which(B$v9066 == 0)] <- NA_integer_
B$v9067[which(B$v9067 == 0)] <- NA_integer_

B$v9070[which(B$v9070 == 0)] <- NA_integer_
B$v9071[which(B$v9071 == 0)] <- NA_integer_

B$v9074[which(B$v9074 == 0)] <- NA_integer_
B$v9075[which(B$v9075 == 0)] <- NA_integer_

B$v9078[which(B$v9078 == 0)] <- NA_integer_
B$v9079[which(B$v9079 == 0)] <- NA_integer_

B$v9082[which(B$v9082 == 0)] <- NA_integer_
B$v9083[which(B$v9083 == 0)] <- NA_integer_

B$v9082[which(B$v9082 == 0)] <- NA_integer_
B$v9083[which(B$v9083 == 0)] <- NA_integer_

B$v9086[which(B$v9086 == 0)] <- NA_integer_
B$v9087[which(B$v9087 == 0)] <- NA_integer_

B$v9090[which(B$v9090 == 0)] <- NA_integer_
B$v9091[which(B$v9091 == 0)] <- NA_integer_

B$v9094[which(B$v9094 == 0)] <- NA_integer_
B$v9095[which(B$v9095 == 0)] <- NA_integer_

B$v9098[which(B$v9098 == 0)] <- NA_integer_
B$v9099[which(B$v9099 == 0)] <- NA_integer_

B$v9102[which(B$v9102 == 0)] <- NA_integer_
B$v9105[which(B$v9105 == 0)] <- NA_integer_

# Capítulo 10
B$v10000[which(B$v10000 == 0)] <- NA_integer_
B$v10001[which(B$v10001 == 0)] <- NA_integer_

table(B$v10i2, useNA = "ifany")

B$v10i2[which(B$inec_identificador_empresa == '14773646091')] <- 2
B$v10003[which(B$inec_identificador_empresa == '14773646091')] <- NA_integer_
B$v10i3[which(B$inec_identificador_empresa == '14773646091')] <- 1

i <- which(B$v10i2 == 2 & B$v10004 == 0 & B$v10005 == 0)
B$v10003[i] <- B$v10004[i] <- B$v10005[i] <- NA_integer_
B$v10i2[which(B$inec_identificador_empresa == '49476634091')] <- 1

table(B$v10i3, useNA = "ifany")

i <- which(B$v10006 == 2 & B$v10014 == 2 & B$v10022 == 2)
B$v10003 [i] <- 2
B$v10006 [i] <- NA_integer_
B$v10014 [i] <- NA_integer_
B$v10022 [i] <- NA_integer_

m <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, min)

i <- which(B$v10006 == 0 & B$v10014 == 0 & B$v10022 == 0)
ind_inicial <- which(vars %in% "v10006")
ind_final <- which(vars %in% "v10031")
for (k in i)
  for (var in vars[ind_inicial:ind_final])
```

```
   B[k, var] <- NA_integer_

m <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, min)
i <- which(B$v10006 == 0 | B$v10014 == 0 | B$v10022 == 0)

B$v10014[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10016[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10020[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10022[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10024[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10028[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10030[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_
B$v10031[which(B$inec_identificador_empresa == "13831642178")] <- NA_integer_

m <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, min)
i <- which(m == 0)
B$v10006[i] <- B$v10022[i] <- 2
B$v10008[i] <- B$v10012[i] <- B$v10024[i] <- B$v10028[i] <- NA_integer_

m <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, min)
table(B$v10i3, m, useNA = "ifany")

B$v10i3[which(B$inec_identificador_empresa == "14773646091")] <- 1
B$v10024[which(B$inec_identificador_empresa == "14773646091")] <- NA_integer_

m <- apply(cbind(B$v10006, B$v10014, B$v10022), 1, min)
table(B$v10i3, m, useNA = "ifany")

i <- which(B$v10i3 == 1 & is.na(m))
B$v10014[which(B$inec_identificador_empresa == "13831642178")] <- 2
B$v10022[which(B$inec_identificador_empresa == "13831642178")] <- 2

B$v10i3[is.na(B$v10i3)] <- 2

#---------------------------------------

table(B$v10ii1, useNA = "ifany")
m <- (B$v10ii11 > 0)
table(B$v10ii1, m, useNA = "ifany")

i <- which(B$v10ii1 == 2 & B$v10ii11 == 0)
B$v10ii11[i] <- NA_integer_

m <- (B$v10ii11 > 0)
table(B$v10ii1, m, useNA = "ifany")

B$v10ii1[which(B$inec_identificador_empresa == "14641254135")] <- 2
B$v10ii11[which(B$inec_identificador_empresa == "14641254135")] <- NA_integer_

m <- (B$v10ii11 > 0)
table(B$v10ii1, m, useNA = "ifany")

B$v10ii1[which(is.na(B$v10ii1))] <- 2
B$v10ii2[which(is.na(B$v10ii2))] <- 2
B$v10ii3[which(is.na(B$v10ii3))] <- 2

B$v10032[which(B$inec_identificador_empresa == "44537311174")] <- 16
B$v10033[which(B$inec_identificador_empresa == "44537311174")] <- 17
B$v10034[which(B$inec_identificador_empresa == "44537311174")] <- 23
B$v10035[which(B$inec_identificador_empresa == "44537311174")] <- 75072

B$v10032[which(B$inec_identificador_empresa == "13627013056")] <- 38
B$v10033[which(B$inec_identificador_empresa == "13627013056")] <- 13
B$v10034[which(B$inec_identificador_empresa == "13627013056")] <- 23
B$v10035[which(B$inec_identificador_empresa == "13627013056")] <- 136344

B$v10032[which(B$inec_identificador_empresa == "14783702178")] <- 291.5
```

```
B$v10033[which(B$inec_identificador_empresa == "14783702178")] <- 7
B$v10034[which(B$inec_identificador_empresa == "14783702178")] <- 23
B$v10035[which(B$inec_identificador_empresa == "14783702178")] <- 563178

B$v10032[which(B$inec_identificador_empresa == "14829783135")] <- 2388
B$v10033[which(B$inec_identificador_empresa == "14829783135")] <- 24
B$v10034[which(B$inec_identificador_empresa == "14829783135")] <- 27
B$v10035[which(B$inec_identificador_empresa == "14829783135")] <- 18569088

B$v10032[which(B$inec_identificador_empresa == "13706438097")] <- 43
B$v10033[which(B$inec_identificador_empresa == "13706438097")] <- 20
B$v10034[which(B$inec_identificador_empresa == "13706438097")] <- 30
B$v10035[which(B$inec_identificador_empresa == "13706438097")] <- 309600

#-------------------------------------------------------------------------------------------------------

m <- apply(cbind(B$v10ii5111, B$v10ii5112, B$v10ii5113, B$v10ii5114), 1, min)
B$v10ii5[which(is.na(B$v10ii5))] <- 2
```

3. Código de integridad para las variables del Capítulo 10.3 (Residuos y/o desechos).

```{r}
B$v10064[which(B$v10062 == 2)] <- NA_integer_

B$v10062[which(is.na(B$v10062))] <- 2

#-------------------------------------------------------------------------------------------------------

B$v10085[which(B$inec_identificador_empresa == "13711280094")] <- 1
i <- which(B$v10083 == 1 & B$v10085 == 1 & B$v10086 == 0)
B$v10085[i] <- NA_integer_
i <- which(B$v10083 == 1 & B$v10085 == 1 & is.na(B$v10086))
B$v10085[i] <- NA_integer_
B$v10086[i] <- 0

i <- which(B$v10083 == 1 & is.na(B$v10085) & is.na(B$v10086))
B$v10086[i] <- 0

B$v10083[which(is.na(B$v10083))] <- 2

#-------------------------------------------------------------------------------------

i <- which(B$v10104 == 1 & B$v10106 == 1 & B$v10107 == 0)
B$v10106[i] <- NA_integer_

i <- which(B$v10104 == 1 & is.na(B$v10106) & is.na(B$v10107))
B$v10107[i] <- 0

B$v10104[which(is.na(B$v10104))] <- 2

#-------------------------------------------------------------------------------------

i <- which(B$v10125 == 1 & B$v10127 == 1 & B$v10128 == 0)
B$v10127[i] <- NA_integer_

i <- which(B$v10125 == 1 & B$v10127 == 1 & is.na(B$v10128))
B$v10127[i] <- NA_integer_
B$v10128[i] <- 0
B$v10142[i] <- 100

i <- which(B$v10125 == 1 & is.na(B$v10127) & is.na(B$v10128))
B$v10128[i] <- 0

i <- which(B$v10125 == 2 & B$v10127 == 2)
B$v10127[i] <- NA_integer_

B$v10125[which(is.na(B$v10125))] <- 2
```

```
#-------------------------------------------------------------------------------
i <- which(B$v10146 == 1 & B$v10148 == 1 & B$v10149 == 0)
B$v10148[i] <- NA_integer_

i <- which(B$v10146 == 1 & is.na(B$v10148) & B$v10149 == 150)
B$v10148[i] <- 1

i <- which(B$v10146 == 1 & is.na(B$v10148) & is.na(B$v10149))
B$v10149[i] <- 0

B$v10146[which(is.na(B$v10146))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10167 == 1 & B$v10169 == 1 & B$v10170 == 0)
B$v10169[i] <- NA_integer_

i <- which(B$v10167 == 1 & B$v10169 == 2 & is.na(B$v10170))
B$v10169[i] <- NA_integer_
B$v10170[i] <- 0
B$v10178[i] <- 0
B$v10184[i] <- 100

B$v10167[which(is.na(B$v10167))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10188 == 1 & is.na(B$v10190) & is.na(B$v10191))
B$v10191[i] <- 0

i <- which(B$v10188 == 2 & B$v10190 == 2)
B$v10190[i] <- NA_integer_
B$v10191[i] <- 0

B$v10188[which(is.na(B$v10188))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10209 == 1 & is.na(B$v10211) & is.na(B$v10212))
B$v10212[i] <- 0

i <- which(B$v10209 == 1 & B$v10211 == 1 & B$v10212 == 0)
B$v10211[i] <- NA_integer_

i <- which(B$v10209 == 2 & B$v10211 == 2)
B$v10211[i] <- NA_integer_

B$v10209[which(is.na(B$v10209))] <- 2

#---------------------------------------------------------

i <- which(B$v10230 == 2 & B$v10232 == 2)
B$v10232[i] <- NA_integer_

B$v10230[which(is.na(B$v10230))] <- 2

#---------------------------------------------------------
i <- which(B$v10251 == 2 & B$v10253 == 2)
B$v10253[i] <- NA_integer_

B$v10251[which(is.na(B$v10251))] <- 2

#---------------------------------------------------------

i <- which(B$v10272 == 2 & B$v10274 == 2)
B$v10274[i] <- NA_integer_
```

```
B$v10272[which(is.na(B$v10272))] <- 2

#---------------------------------------------------------

i <- which(B$v10314 == 2 & B$v10316 == 2)
B$v10316[i] <- NA_integer_

B$v10272[which(is.na(B$v10272))] <- 2

#---------------------------------------------------------

B$v10iii1_1[which(is.na(B$v10iii1_1))] <- 2

#---------------------------------------------------------

B$v10iii1_2[which(is.na(B$v10iii1_2))] <- 2

#---------------------------------------------------------

B$v10iii1_3[which(is.na(B$v10iii1_3))] <- 2

#-----------------------------------------------------------------------------------

i <- which(B$v10377 == 1 & is.na(B$v10379) & (B$v10380 > 0))
B$v10379[i] <- 1

i <- which(B$v10377 == 1 & is.na(B$v10379) & is.na(B$v10380))
B$v10380[i] <- 0

B$v10377[which(is.na(B$v10377))] <- 2

#---------------------------------------------------------

i <- which(B$v10398 == 2 & B$v10400 == 3)
B$v10400[i] <- NA_integer_

B$v10398[which(is.na(B$v10398))] <- 2

#---------------------------------------------------------

i <- which(B$v10419== 2 & B$v10421 == 3)
B$v10421[i] <- NA_integer_

B$v10419[which(is.na(B$v10419))] <- 2

#-----------------------------------------------------------------------------------

i <- which(B$v10440== 1 & is.na(B$v10442) & is.na(B$v10443))
B$v10443[i] <- B$v10451[i] <- 0
B$v10457[i] <- 100

i <- which(B$v10440== 2 & B$v10442 == 3)
B$v10442[i] <- NA_integer_

B$v10440[which(is.na(B$v10440))] <- 2

#-----------------------------------------------------------------------------------

i <- which(B$v10461== 1 & is.na(B$v10463) & is.na(B$v10464))
B$v10464[i] <- B$v10472[i] <- 0
B$v10457[i] <- 100

i <- which(B$v10461== 2 & B$v10463 == 3)
B$v10463[i] <- NA_integer_

B$v10461[which(is.na(B$v10461))] <- 2
```

```
#-----------------------------------------------------------
i <- which(B$v10524 == 2 & B$v10526 == 3)
B$v10526[i] <- NA_integer_

B$v10461[which(is.na(B$v10461))] <- 2

#-----------------------------------------------------------

B$v10iii2_1[which(is.na(B$v10iii2_1))] <- 2

#-----------------------------------------------------------

B$v10iii2_2[which(is.na(B$v10iii2_2))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10545 == 1 & is.na(B$v10547) & is.na(B$v10548))
B$v10548[i] <- 0

i <- which(B$v10545 == 1 & is.na(B$v10547) & B$v10548 > 0)
B$v10547[i] <- 3

i <- which(B$v10545 == 1 & B$v10547 > 0 & B$v10548 == 0)
B$v10548[i] <- NA_integer_

B$v10545[which(is.na(B$v10545))] <- 2

#-------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_2) | B$tipo_desecho_2 == "")
B$tipo_desecho_2[i] <- "(NE-42) - Material adsorbente contaminado con hidrocarburos: waipes, paños, trapos, aserrín, barreras
adsorbentes y otros materiales sólidos adsorbentes"

i <- which(B$v10566 == 1 & is.na(B$v10568) & B$v10569 > 0)
B$v10568[i] <- 1

i <- which(B$v10566 == 2 & B$v10568 == 3)
B$v10568[i] <- NA_integer_

B$v10545[which(is.na(B$v10545))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10587 == 1 & is.na(B$v10589) & is.na(B$v10569))
B$v10590[i] <- 0

i <- which(B$v10587 == 1 & is.na(B$v10589) & B$v10569 > 0)
B$v10589[i] <- 3

B$v10587[which(is.na(B$v10587))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10608 == 1 & is.na(B$v10610) & B$v10611 > 0)
B$v10610[i] <- 1

B$v10608[which(is.na(B$v10608))] <- 2

#-------------------------------------------------------------------------------

i <- which(B$v10629 == 1 & is.na(B$v10631) & is.na(B$v10632))
B$v10632[i] <- 0

i <- which(B$v10629 == 2 & B$v10631 == 3 & is.na(B$v10632))
B$v10631[i] <- NA_integer_

B$v10629[which(is.na(B$v10629))] <- 2
```

```
#------------------------------------------------------------------------------

i <- which(B$v10650 == 1 & is.na(B$v10652) & is.na(B$v10653))
B$v10653[i] <- 0

i <- which(B$v10650 == 1 & is.na(B$v10652) & B$v10653 == 4)
B$v10652[i] <- 2

i <- which(B$v10650 == 1 & is.na(B$v10652) & B$v10653 == 2)
B$v10652[i] <- 1

B$v10650[which(is.na(B$v10650))] <- 2

#------------------------------------------------------------------------------

i <- which(B$v10671 == 1 & is.na(B$v10673) & is.na(B$v10674))
B$v10674[i] <- 0

i <- which(B$v10671 == 2 & B$v10673 == 3)
B$v10673[i] <- NA_integer_

B$v10671[which(is.na(B$v10671))] <- 2

#-------------------------------------------------------

i <- which(B$v10692 == 2 & B$v10694 == 3)
B$v10694[i] <- NA_integer_

B$v10692[which(is.na(B$v10692))] <- 2

#------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_9) | B$tipo_desecho_9 == "")
B$tipo_desecho_9[i] <- "(NE-53) - Cartuchos de impresión de tinta o tóner usados"

i <- which(B$v10713 == 1 & is.na(B$v10715) & is.na(B$v10716))
B$v10716[i] <- 0

B$v10713[which(is.na(B$v10713))] <- 2

#------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_10) | B$tipo_desecho_10 == "")
B$tipo_desecho_10[i] <- "(NE-30) - Equipo de protección personal contaminado con materiales peligrosos"

i <- which(B$v10734 == 2 & B$v10736 == 3)
B$v10736[i] <- NA_integer_

B$v10734[which(is.na(B$v10734))] <- 2

#------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_11) | B$tipo_desecho_11 == "")
B$tipo_desecho_11[i] <- "(NE-43) - Material adsorbente contaminado con sustancias químicas peligrosas: waipes, paños, trapos,
aserrín, barreras adsorbentes y otros materiales sólidos adsorbentes"

i <- which(B$v10755 == 1 & is.na(B$v10757) & B$v10758 == 50)
B$v10757[i] <- 1

i <- which(B$v10755 == 2 & B$v10757 == 3)
B$v10757[i] <- NA_integer_

B$v10755[which(is.na(B$v10755))] <- 2

#------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_12) | B$tipo_desecho_12 == "")
```

```
B$tipo_desecho_12[i] <- "(NE-08) - Baterías usadas que contengan Hg, Ni, Cd u otros materiales peligrosos y que exhiban características de peligrosidad."

i <- which(B$v10776 == 1 & is.na(B$v10778) & B$v10779 == 276)
B$v10778[i] <- 1

i <- which(B$v10776 == 2 & B$v10778 == 3)
B$v10778[i] <- NA_integer_

B$v10776[which(is.na(B$v10776))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_13) | B$tipo_desecho_13 == "")
B$tipo_desecho_13[i] <- "(Q.86.08) - Fármacos caducados o fuera de especificaciones"

i <- which(B$v10797 == 1 & is.na(B$v10799) & B$v10800 == 9)
B$v10799[i] <- 1

B$v10797[which(is.na(B$v10797))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_14) | B$tipo_desecho_14 == "")
B$tipo_desecho_14[i] <- "(NE-10) - Desechos biopeligrosos activos resultantes de la atención médica prestados en centros médicos de empresas"

i <- which(B$v10818 == 1 & is.na(B$v10820) & B$v10821 > 0)
B$v10820[i] <- 1

B$v10818[which(is.na(B$v10818))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_15) | B$tipo_desecho_15 == "")
B$tipo_desecho_15[i] <- "(NE-35) - Hidrocarburos sucios o contaminados con otras sustancias"

B$v10839[which(is.na(B$v10839))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_16) | B$tipo_desecho_16 == "")
B$tipo_desecho_16[i] <- "(Q.86.03) - Sangre, sus derivados e insumos usados para procedimientos de análisis y administración de los mismos."

B$v10860[which(is.na(B$v10860))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_17) | B$tipo_desecho_17 == "")
B$tipo_desecho_17[i] <- "(NE-38) - Lodos de tanques de almacenamiento de hidrocarburos"

B$v10881[which(is.na(B$v10881))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_18) | B$tipo_desecho_18 == "")
B$tipo_desecho_18[i] <- "(NE-34) - Aceites, grasas y ceras usadas o fuera de especificaciones"

i <- which(B$v10902 == 1 & is.na(B$v10904) & B$v10905 > 0)
B$v10904[i] <- 1

B$v10902[which(is.na(B$v10902))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_19) | B$tipo_desecho_19 == "")
```

```
B$tipo_desecho_19[i] <- "(G.46.01) - Lodos de las plantas de tratamiento de aguas residuales industriales  que contienen sustancias peligrosas"

B$v10923[which(is.na(B$v10923))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_20) | B$tipo_desecho_20 == "")
B$tipo_desecho_20[i] <- "(NE-09) - Chatarra contaminada con materiales peligrosos"

B$v10944[which(is.na(B$v10944))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_21) | B$tipo_desecho_21 == "")
B$tipo_desecho_21[i] <- "(NE-49) - Residuos de tintas, pinturas, resinas que contengan sustancias peligrosas y exhiban características de peligrosidad"

B$v10965[which(is.na(B$v10965))] <- 2

#---------------------------------------------------------------------------------

i <- which(is.na(B$tipo_desecho_22) | B$tipo_desecho_22 == "")
B$tipo_desecho_22[i] <- "(NE-36) - Lodos de aceite"

B$v10986[which(is.na(B$v10986))] <- 2

#---------------------------------------------------------

B$v10iii3_1[which(is.na(B$v10iii2_1))] <- 2

#---------------------------------------------------------

B$v10iii3_2[which(is.na(B$v10iii2_2))] <- 2

#---------------------------------------------------------

rm(i, ind_inicial, ind_final, k, m, s, s1)
```

4. Código de integridad de las variables imputadas para la empresa "Petroecuador".

```
id <- "13824423177"
i <- which(B$inec_identificador_empresa == id)

B$v9036[which(B$id == '13824423177')] <- 1
B$v9037[which(B$id == '13824423177')] <- 1832365004
B$v9038[which(B$id == '13824423177')] <- 209507301
B$v9039[which(B$id == '13824423177')] <- 1832365004
B$v9052[which(B$id == '13824423177')] <- 1832365004
B$v9053[which(B$id == '13824423177')] <- 209507301
B$v9054[which(B$id == '13824423177')] <- 1832365004

B$v10379[i] <- 2 # Se cambia la unidad de kg a toneladas.
# No se imputa la variable v10422, porque el valor es consistente con los de la serie de años 2018 a 2021.
# En el año 2022 se ingresó erróneamente el valor de 1147 toneladas, cuando debió ser 1147 kg.
B$v10iii3_1[i] <- 1
B$v10076_3[i] <- 2522165
rm(i, id)
```

5. Código de integridad de las variables imputadas para la empresa "GasGreen".

```
id <- "14832210171"
i <- which(B$inec_identificador_empresa == id)
B$v9021[i] <- NA_integer_
B$v9045[i] <- 33853764
B$v9057 <- "BIOGAS"
```

```
rm(i, id)
```

6. Exportación a SPSS y Excel de la BDD MIEA 2023 imputada.

```{r}
writexl::write_xlsx(J, "C:/Users/rbenavides/Documents/EMPRESAS 2023/Integridad 2023/BDD_MIEAE_2023_INTEGRA.xlsx")
haven::write_sav(J, "C:/Users/rbenavides/Documents/EMPRESAS 2023/Integridad 2023/BDD_MIEAE_2023_INTEGRA.sav",
compress = TRUE)
```

#----------

 # FIN DE LA SINTAXIS DE CONTROL DE INTEGRIDAD DE LA BDD AMBIENTAL ENESEM 2023.

#----------

ANEXO C. Código R Markdown para las comparaciones de valores 2023-2022.

---
title: "SX Comparación BDD 2023-2022"
author: "Ramiro Benavides"
date: "10/08/2023"
update: "20/06/2024"
output: html_document

A. Carga de las BDD 2023 y 2022 del Módulo Ambiental ENESEM.

```{r}
# B_2023 es el frame que carga la BDD a validar del último corte criticado.
B_2023 <- haven::read_sav("~/EMPRESAS 2023/BDD/Empresas_03_12_2024.sav")

# B_2022 es el frame de la base de publicación 2022.
B_2022 <- haven::read_sav ("~/EMPRESAS 2022/BDD_MIEA_2022_PUB_03_2024.sav")

# "Origen" es la BDD que tiene la información de zonal de crítica por empresa.
# Origen <- readxl::read_excel("~/EMPRESAS 2023/BDD /Empresas_val_03_12_2024.xlsx")

# B_2022_BKUP es el respaldo del frame "B_2022".
# B_2023_BKUP es el respaldo del frame "B_2023".

B_2022 <- as.data.frame(B_2022)
B_2023 <- as.data.frame(B_2023)

B_2023_BK <- B_2023
B_2022_BK <- B_2022

# El frame C es la juntura de los frames 2022 y 2023. Sirve para determinar la intersección de identificadores entre las empresas 2023 criticadas y las empresas 2022 publicadas.

C <- merge(B_2023, B_2022, by.x="inec_identificador_empresa", by.y="inec_identificador_empresa")
id <- C$inec_identificador_empresa
rm(C)
```

B. Determinación de las variables a transformar y las que no se deben transformar.

```{r}
# Creación de vector con variables de identificación de empresas para los frames de reporte.

lista_vars_identif <- c("inec_identificador_empresa", "secuencial_sistema", "Zonal_DEAGA", "Estado", "ciiu4_actividad_principal")

# Creación de vector con variables de control (numéricas) para los frames de reporte.

# sec_var_sin_trans <- c(7003, 7004, 7005, 7006, 8098, 8099, 8100, 9002, 9053, 9054, 9056, 9105, 10001, 10030, 10035)
sec_var_sin_trans <- c(7002, 7003, 7004, 7005, 7006, 8100, 8098, 8099, 8091, 8097, 9001, 9002, 9005, 9007, 9013, 9015, 9021, 9023, 9029, 9031, 9037, 9039, 9045, 9047, 9052, 9053, 9054, 9055, 9056, 9058, 9059, 9062, 9063, 9066, 9067, 9070, 9071, 9074, 9075, 9078, 9079, 9082, 9083, 9086, 9087, 9090, 9091, 9094, 9095, 9098, 9099, 9102, 9105, 10000, 10001, 10012, 10020, 10028, 10030, 10035)
var_sin_trans <- paste0("v", sec_var_sin_trans)
# var_sin_trans <- c(var_sin_trans, "v10ii11", "v10ii5", "v10ii6")

sec_var_trans <- c(10004, 10065, 10086, 10107, 10128, 10149, 10170, 10191, 10212, 10233, 10254, 10275, 10317, 10380, 10401, 10422, 10443, 10464, 10527)

var_trans <- paste0("v", sec_var_trans)
var_chk <- c(var_sin_trans, var_trans)
```

C. Creación de variables transformadas a unidades estándar para el frame del año 2023.

```{r}
B <- B_2023
# Creación de variables transformadas para las pertenecientes al vector de cadenas "var_trans".
```

```
B$v10004 <- ifelse(B$v10003 == 1, B$v10004 * 0.00378541, B$v10004) # B$v10003 == 1 (US gal).
for (j in 1:length(sec_var_trans)) {
#for (j in 2:length(sec_var_trans)) {
  for (i in 1:dim(B)[1]) {
    B[i, var_trans[j]] <- as.numeric(ifelse(B[i, paste0("v", sec_var_trans[j]-1)] == 2, B[i, var_trans[j]] * 1000, ifelse(B[i, paste0("v",
sec_var_trans[j]-1)] == 3, B[i, var_trans[j]] * 3.78541, B[i, var_trans[j]])))
  }
  print(paste("Columna", var_trans[j], "procesada."))
}
D_2023 <- B
```

D. Creación de variables transformadas a unidades estándar para el frame del año 2022.

```{r}
B <- B_2022
# Creación de variables transformadas para las pertenecientes al vector de cadenas "var_trans".
B$v10004 <- ifelse(B$v10003 == 1, B$v10004 * 0.00378541, B$v10004) # B$v10003 == 1 (US gal).
for (j in 2:length(sec_var_trans)) {
  for (i in 1:dim(B)[1]) {
    B[i, var_trans[j]] <- ifelse(B[i, paste0("v", sec_var_trans[j]-1)] == 2, B[i, var_trans[j]] * 1000, ifelse(B[i, paste0("v",
sec_var_trans[j]-1)] == 3, B[i, var_trans[j]] * 3.78541, B[i, var_trans[j]]))
  }
  print(paste("Columna", var_trans[j], "procesada."))
}
D_2022 <- B
rm(B, i, j)
```

E. Creación de matrices auxiliares que servirán para estandarizar (a kilogramos) las cantidades de desechos peligrosos, Año 2022. Estas variables se adjuntarán al final de la BDD estandarizada del año 2022.

```{r}
sec_canon <- c(166, 180, 224, 226, 228, 230, 231, 232, 242, 245, 266, 271, 277, 281, 284)
# Secuenciales de los desechos peligrosos, según el cánon, que se irán a validar.
nom_canon <- c("(E.38.02) - Lixiviados generados en vertederos, rellenos y celdas de seguridad",
               "(G.46.01) - Lodos de las plantas de tratamiento de aguas residuales industriales  que contienen sustancias peligrosas",
               "(Q.86.01) - Cultivos de agentes infecciosos y desechos de producción biológica, vacunas vencidas o inutilizadas, cajas
de petri, placas de frotis y todos los instrumentos usados para manipular, mezclar o inocular microorganismos.",
               "(Q.86.03) - Sangre, sus derivados e insumos usados para procedimientos de análisis y administración de los mismos.",
               "(Q.86.05) - Objetos cortopunzantes que han sido utilizados en la atención de seres humanos o animales; en la
investigación, en laboratorios y administración de fármacos.",
               "(Q.86.07) - Material e insumos que han sido utilizados para procedimientos médicos y que han estado en contacto con
fluidos corporales",
               "(Q.86.08) - Fármacos caducados o fuera de especificaciones",
               "(Q.86.09) - Desechos químicos de laboratorio, químicos caducados o fuera de especificaciones",
               "(NE-03) - Aceites minerales usados o gastados",
               "(NE-06) - Aguas residuales industriales que cuyas concentraciones de Cr (VI), As, Cd, Se, Sb, Te, Hg, Tl, Pb, cianuros,
fenoles u otras sustancias peligrosas excedan los límites máximos permitidos (Anexo 1 del Libro VI del TULSMA)",
               "(NE-27) - Envases contaminados con materiales peligrosos",
               "(NE-32) - Filtros usados de aceite mineral",
               "(NE-38) - Lodos de tanques de almacenamiento de hidrocarburos",
               "(NE-42) - Material adsorbente contaminado con hidrocarburos: waipes, paños, trapos, aserrín, barreras adsorbentes y
otros materiales sólidos adsorbentes",
               "(NE-45) - Mezclas oleosas, emulsiones de hidrocarburos- agua, desechos de taladrina")
# Nombres de los desechos del cánon a validar, según aparece en la BDD 2022 publicada.

sec_genera <- seq(10545, 10986, 21)
# Secuenciales de todas las variables de la columna (1) de generación de desechos peligrosos.

var_genera <- paste0("v", sec_genera)
# Vector de cadena que contiene los nombres de las variables del formulario que corresponden
# a la columna (1) (¿Genéro el desecho peligroso?) de la Tabla 10.III.3.

ind_genera_2022 <- sapply(var_genera, function(x) which(x == names(B_2022)))
# Vector de índices de columna donde se encuentran las variables generadoras de desechos peligrosos.

ind_nombre_2022 <- ind_genera_2022 - 1
# Vector de índices de columna donde se encuentran las variables de nombres de desechos peligrosos.
```

```
ind_unidad_2022 <- ind_genera_2022 + 2
# Vector de índices de columna donde se encuentran las variables de unidad de medida de desechos peligrosos.

ind_cantidad_2022 <- ind_genera_2022 + 3
# Vector de índices de columna donde se encuentran las variables de cantidad de desechos peligrosos.

nombres_des <- data.frame(matrix(NA_character_, nrow=dim(B_2022)[1], ncol=length(ind_genera_2022)))
# Frame que contiene los nombres de todos los desechos peligrosos generados por cada empresa.

unidad_des <- data.frame(matrix(NA_integer_, nrow=dim(B_2022)[1], ncol=length(ind_genera_2022)))
# Frame que contiene las unidades de medida de todos los desechos peligrosos generados por cada empresa.

cantidad_des <- data.frame(matrix(NA_integer_, nrow=dim(B_2022)[1], ncol=length(ind_genera_2022)))
# Frame que contiene las cantidades de todos los desechos peligrosos generados por cada empresa.

names(nombres_des) <- names(unidad_des) <- names(cantidad_des) <- var_genera

for (j in 1:22) {
  unidad_des[, j] <- sapply(1:dim(B_2022)[1], function(i) ifelse(B_2022[i, ind_genera_2022[j]] == 1, B_2022[i, ind_unidad_2022[j]],
NA_integer_))
  nombres_des[, j] <- sapply(1:dim(B_2022)[1], function(i) ifelse(B_2022[i, ind_genera_2022[j]] == 1 & unidad_des[i, j] > 0,
B_2022[i, ind_nombre_2022[j]], NA_character_))
  cantidad_des[, j] <- sapply(1:dim(B_2022)[1], function(i) ifelse(B_2022[i, ind_genera_2022[j]] == 1, B_2022[i,
ind_cantidad_2022[j]], NA_integer_))
  print(paste("Se ha procesado con éxito la columna", var_genera[j]))
}
```

F. Creación de las variables de desechos peligrosos estandarizados (en kilogramos), Año 2022. Estas variables se adjuntarán al final de la BDD estandarizada del año 2022 ('D_2022').

```{r}
Res_2022 <- data.frame(matrix(NA_real_, nrow=dim(B_2022)[1], ncol=length(ind_genera_2022)))
names(Res_2022) <- paste0("desecho_", 1:22)
# 'Res_2022' contiene todas las cantidades estandarizadas de los desechos peligrosos
# que aparecen en las columnas de nombre 'tipo_desecho_NN' de la BDD del corte actual.

for (j in 1:22) {
  Res_2022[, j] <- sapply(1:dim(B_2022)[1], function(i) ifelse(unidad_des[i, j] == 2, cantidad_des[i, j] * 1000, ifelse(unidad_des[i, j]
== 3, cantidad_des[i, j] * 3.78541, cantidad_des[i, j])))
  print(paste("Se ha procesado con éxito la columna", var_genera[j]))
}
# En este código se realiza la estandarización concreta de las cantidades de desechos peligrosos.

var_canon <- paste0("y", sec_canon)
# Vector de cadenas que contiene los nombres de variables de los desechos peligrosos estandarizados,
# tal y como aparecerían si se los extrae desde la BDD derivada de desechos peligrosos.

library(foreach)
nom_des_x_fila <- foreach(x=1:dim(B_2022)[1]) %do% as.integer(sapply(1:22, function(y) which(nombres_des[x, y] ==
nom_canon)))
nom_des_x_fila <- sapply(1:dim(B_2022)[1], function(y) var_canon[as.integer(na.omit(nom_des_x_fila[[y]]))])
# Lista que contiene los nombres de variables de desechos peligrosos estandarizados para los
# desechos que están en el cánon y que provienen del frame 'nombres_des'.

cant_des_x_fila <- foreach(x=1:dim(B_2022)[1]) %do% as.numeric(Res_2022[x, which(nombres_des[x, ] %in% nom_canon)])
# Lista que contiene las cantidades de desechos peligrosos para los desechos que están en el cánon
# y que provienen del frame 'Res_2022'.

Repo_2022 <-data.frame(matrix(NA_real_, nrow=dim(B_2022)[1], ncol=length(nom_canon)))
names(Repo_2022) <- var_canon
# Este frame es el que incluirá las columnas de desechos peligrosos estándar que después se
# usarán para realizar las comparaciones interanuales.

for (i in 1:dim(B_2022)[1]) {
  n <- length(cant_des_x_fila[[i]])
  if (n > 0) {
    for (j in 1:n) Repo_2022[i, nom_des_x_fila[[i]][j]] <- cant_des_x_fila[[i]][j]
```

```
  }
  print(paste0("Se ha copiado con éxito la fila ", i, " al frame Repo_2022"))
}

D_2022 <- cbind(D_2022, Repo_2022)

rm(ind_cantidad_2022, ind_genera_2022, ind_nombre_2022, ind_unidad_2022)
rm(i, j, n, Repo_2022, Res_2022, cantidad_des, cant_des_x_fila, nombres_des, nom_des_x_fila, unidad_des)
```

G. Creación de matrices auxiliares que servirán para estandarizar (a kilogramos) las cantidades de desechos peligrosos, Año 2023. Estas variables se adjuntarán al final de la BDD estandarizada del año 2023.

```{r}
# sec_canon <- c(166, 180, 224, 226, 228, 230, 231, 232, 242, 245, 266, 271, 277, 281, 284)
# # Secuenciales de los desechos peligrosos, según el cánon, que se irán a validar.
# nom_canon <- c("(E.38.02) - Lixiviados generados en vertederos, rellenos y celdas de seguridad",
#                "(G.46.01) - Lodos de las plantas de tratamiento de aguas residuales industriales  que contienen sustancias peligrosas",
#                "(Q.86.01) - Cultivos de agentes infecciosos y desechos de producción biológica, vacunas vencidas o inutilizadas, cajas de petri, placas de frotis y todos los instrumentos usados para manipular, mezclar o inocular microorganismos.",
#                "(Q.86.03) - Sangre, sus derivados e insumos usados para procedimientos de análisis y administración de los mismos.",
#                "(Q.86.05) - Objetos cortopunzantes que han sido utilizados en la atención de seres humanos o animales; en la investigación, en laboratorios y administración de fármacos.",
#                "(Q.86.07) - Material e insumos que han sido utilizados para procedimientos médicos y que han estado en contacto con fluidos corporales",
#                "(Q.86.08) - Fármacos caducados o fuera de especificaciones",
#                "(Q.86.09) - Desechos químicos de laboratorio, químicos caducados o fuera de especificaciones",
#                "(NE-03) - Aceites minerales usados o gastados",
#                "(NE-06) - Aguas residuales industriales que cuyas concentraciones de Cr (VI), As, Cd, Se, Sb, Te, Hg, Tl, Pb, cianuros, fenoles u otras sustancias peligrosas excedan los límites máximos permitidos (Anexo 1 del Libro VI del TULSMA)",
#                "(NE-27) - Envases contaminados con materiales peligrosos",
#                "(NE-32) - Filtros usados de aceite mineral",
#                "(NE-38) - Lodos de tanques de almacenamiento de hidrocarburos",
#                "(NE-42) - Material adsorbente contaminado con hidrocarburos: waipes, paños, trapos, aserrín, barreras adsorbentes y otros materiales sólidos adsorbentes",
#                "(NE-45) - Mezclas oleosas, emulsiones de hidrocarburos- agua, desechos de taladrina")
# # Nombres de los desechos del cánon a validar, según aparece en la BDD 2022 publicada.
#
# sec_genera <- seq(10545, 10986, 21)
# # Secuenciales de todas las variables de la columna (1) de generación de desechos peligrosos.
#
# var_genera <- paste0("v", sec_genera)
# Vector de cadena que contiene los nombres de las variables del formulario que corresponden
# a la columna (1) (¿Genéro el desecho peligroso?) de la Tabla 10.III.3.

ind_genera_2023 <- sapply(var_genera, function(x) which(x == names(B_2023)))
# Vector de índices de columna donde se encuentran las variables generadoras de desechos peligrosos.

ind_nombre_2023 <- ind_genera_2023 - 1
# Vector de índices de columna donde se encuentran las variables de nombres de desechos peligrosos.

ind_unidad_2023 <- ind_genera_2023 + 1
# Vector de índices de columna donde se encuentran las variables de unidad de medida de desechos peligrosos.

ind_cantidad_2023 <- ind_genera_2023 + 2
# Vector de índices de columna donde se encuentran las variables de cantidad de desechos peligrosos.

nombres_des <- data.frame(matrix(NA_character_, nrow=dim(B_2023)[1], ncol=length(ind_genera_2023)))
# Frame que contiene los nombres de todos los desechos peligrosos generados por cada empresa.

unidad_des <- data.frame(matrix(NA_integer_, nrow=dim(B_2023)[1], ncol=length(ind_genera_2023)))
# Frame que contiene las unidades de medida de todos los desechos peligrosos generados por cada empresa.

cantidad_des <- data.frame(matrix(NA_integer_, nrow=dim(B_2023)[1], ncol=length(ind_genera_2023)))
# Frame que contiene las cantidades de todos los desechos peligrosos generados por cada empresa.
```

```
names(nombres_des) <- names(unidad_des) <- names(cantidad_des) <- var_genera

for (j in 1:22) {
  unidad_des[, j] <- sapply(1:dim(B_2023)[1], function(i) ifelse(B_2023[i, ind_genera_2023[j]] == 1, B_2023[i, ind_unidad_2023[j]],
NA_integer_))
  nombres_des[, j] <- sapply(1:dim(B_2023)[1], function(i) ifelse(B_2023[i, ind_genera_2023[j]] == 1 & unidad_des[i, j] > 0,
B_2023[i, ind_nombre_2023[j]], NA_character_))
  cantidad_des[, j] <- sapply(1:dim(B_2023)[1], function(i) ifelse(B_2023[i, ind_genera_2023[j]] == 1, B_2023[i,
ind_cantidad_2023[j]], NA_integer_))
  print(paste("Se ha procesado con éxito la columna", var_genera[j]))
}
```

H. Creación de las variables de desechos peligrosos estandarizados (en kilogramos), Año 2023. Estas variables se adjuntarán al final de la BDD estandarizada del año 2023 ('D_2023').

```{r}
Res_2023 <- data.frame(matrix(NA_real_, nrow=dim(B_2023)[1], ncol=length(ind_genera_2023)))
names(Res_2023) <- paste0("desecho_", 1:22)
# 'Res_2022' contiene todas las cantidades estandarizadas de los desechos peligrosos
# que aparecen en las columnas de nombre 'tipo_desecho_NN' de la BDD del corte actual.

for (j in 1:22) {
  Res_2023[, j] <- sapply(1:dim(B_2023)[1], function(i) ifelse(unidad_des[i, j] == 2, cantidad_des[i, j] * 1000, ifelse(unidad_des[i, j]
== 3, cantidad_des[i, j] * 3.78541, cantidad_des[i, j])))
  print(paste("Se ha procesado con éxito la columna", var_genera[j]))
}
# En este código se realiza la estandarización concreta de las cantidades de desechos peligrosos.

var_canon <- paste0("y", sec_canon)
# Vector de cadenas que contiene los nombres de variables de los desechos peligrosos estandarizados,
# tal y como aparecerían si se los extrae desde la BDD derivada de desechos peligrosos.

if (isFALSE("package:foreach" %in% search())) library(foreach) # Carga condicional
# de la librería que ofrece la posibilidad de construir objetos iterables.

nom_des_x_fila <- foreach(x=1:dim(B_2023)[1]) %do% as.integer(sapply(1:22, function(y) which(nombres_des[x, y] ==
nom_canon)))
nom_des_x_fila <- sapply(1:dim(B_2023)[1], function(y) var_canon[as.integer(na.omit(nom_des_x_fila[[y]]))])
# Lista que contiene los nombres de variables de desechos peligrosos estandarizados para los
# desechos que están en el cánon y que provienen del frame 'nombres_des'.

cant_des_x_fila <- foreach(x=1:dim(B_2023)[1]) %do% as.numeric(Res_2023[x, which(nombres_des[x, ] %in% nom_canon)])
# Lista que contiene las cantidades de desechos peligrosos para los desechos que están en el cánon
# y que provienen del frame 'Res_2022'.

Repo_2023 <-data.frame(matrix(NA_real_, nrow=dim(B_2023)[1], ncol=length(nom_canon)))
names(Repo_2023) <- var_canon
# Este frame es el que incluirá las columnas de desechos peligrosos estándar que después se
# usarán para realizar las comparaciones interanuales.

for (i in 1:dim(B_2023)[1]) {
  n <- length(cant_des_x_fila[[i]])
  if (n > 0) {
    for (j in 1:n) Repo_2023[i, nom_des_x_fila[[i]][j]] <- cant_des_x_fila[[i]][j]
    print(paste0("Se ha copiado con éxito la fila ", i, " al frame Repo_2023"))
  }
}

D_2023 <- cbind(D_2023, Repo_2023)

rm(ind_cantidad_2023, ind_genera_2023, ind_nombre_2023, ind_unidad_2023)
rm(i, j, n, x, Repo_2023, Res_2023, cantidad_des, cant_des_x_fila, nombres_des, nom_des_x_fila, unidad_des)

var_chk <- c(var_chk, var_canon) # Union de las variables fijas con las de desechos peligrosos a ser verificadas.
```

I. Creación y llenado del frame E que contendrá las columnas 2021 y 2022 de las variables a verificar. También se incluirán columnas: "Justificación_Zonal", "Observación_Zonal" y "Revisión_PC".

```{r}
lista_vars_identif <- c("inec_identificador_empresa", "ciiu4_actividad_principal") # Recorte temporal del vector.

E <- data.frame(matrix(NA_real_, nrow=length(id), ncol=6*length(var_chk) + 5))
# 'E' es el frame de reporte a nivel nacional que luego se filtrará y exportará a Excel para cada zonal.

ind_2022 <- sapply(id, function(x) which(D_2022$inec_identificador_empresa == x))
ind_2023 <- sapply(id, function(x) which(D_2023$inec_identificador_empresa == x))
for (j in 1:length(var_chk)) {
  for (i in 1:length(id)) {
    a2022 <- E[i, 0+6*j] <- ifelse(var_chk[j] %in% names(D_2022), as.numeric(D_2022[ind_2022[i], var_chk[j]]), NA_real_)
    a2023 <- E[i, 1+6*j] <- as.numeric(D_2023[ind_2023[i], var_chk[j]])
    if (isTRUE(a2022 > 0)) {
      E[i, 2+6*j] <- ifelse(is.na(a2023) | a2023 == 0, -1, (a2023 - a2022) / a2022)
    } else {
      E[i, 2+6*j] <- NA_real_
    }
    E[i, 3+6*j] <- E[i, 4+6*j] <- E[i, 5+6*j] <- NA_character_
  }
  names(E)[0+6*j] <- "Año 2022"
  names(E)[1+6*j] <- "Año 2023"
  names(E)[2+6*j] <- var_chk[j]
  names(E)[3+6*j] <- "Justificación_Zonal"
  names(E)[4+6*j] <- "Observación_Zonal"
  names(E)[5+6*j] <- "Revisión_PC"
  print(paste0("Se ha copiado los datos 2022 y 2023 de la variable ", var_chk[j]))
}

E[sapply(E[], is.nan)] <- NA_real_
E[sapply(E[], is.infinite)] <- NA_real_
E[, 1:length(lista_vars_identif)] <- B_2023[ind_2023, lista_vars_identif]
names(E)[1:length(lista_vars_identif)] <- lista_vars_identif

rm(a2022, a2023, i, j)
```

J. Algoritmo de determinación de umbrales para todas las variables de validación (que se encuentran en el vector de nombres 'var_chk'). Se usará el frame 'E' para tomar las variables a validar.

```{r}
Calc_Umbrales <- function(cota_inf=0.7, cota_sup=0.7) {
u_sup <- sapply(E[, var_chk], function(x) {m <- quantile(x[which(x > 0)], cota_sup, na.rm=T); ifelse(m <= 0.5, 0.5, m)})
# 'u_sup' es el vector que contiene los umbrales superiores de las variables de validación en 'var_chk'.

u_inf <- sapply(E[, var_chk], function(x) {m <- quantile(x[which(x > -1 & x < 0)], cota_inf, na.rm=T); ifelse(m <= -0.5, m, -0.5)})
# 'u_inf' es el vector que contiene los umbrales inferiores de las variables de validación en 'var_chk'.

out_sup <- sapply(1:length(var_chk), function(x) which(E[, var_chk[x]] > u_sup[x]))
# 'out_sup' contiene los índices de los outliers superiores para las variables de validación en 'var_chk'.

out_inf <- sapply(1:length(var_chk), function(x) which(E[, var_chk[x]] < u_inf[x]))
# 'out_inf' contiene los índices de los outliers inferiores para las variab les de validación en 'var_chk'.

outliers <- sapply(1:length(var_chk), function(x) unlist(union(out_inf[x], out_sup[x])))
# 'outliers' contiene los índices de todos los outliers para las variables de validación en 'var_chk'.

N_sup <- sapply(1:length(var_chk), function(x) length(which(E[, var_chk[x]] > 0)))
# 'N-sup' es el recuento de casos positivos para las variables de validación en 'var_chk'.

N_inf <- sapply(1:length(var_chk), function(x) length(which(E[, var_chk[x]] < 0)))
# 'N-sup' es el recuento de casos negativos para las variables de validación en 'var_chk'.

N <- sapply(1:length(var_chk), function(x) length(which(!is.na(E[, var_chk[x]]))))
# 'N' es el recuento de todos los casos válidos (no missing) de las variables de validación en 'var_chk'.

n_sup <- sapply(1:length(var_chk), function(x) length(which(E[which(E[, var_chk[x]] > 0), var_chk[x]] > u_sup[x])))
# 'n_sup' es el recuento de casos que superan el umbral superior para las variables de validación en 'var_chk'.
```

```
n_inf <- sapply(1:length(var_chk), function(x) length(which(E[which(E[, var_chk[x]] > -1 & E[, var_chk[x]] < 0), var_chk[x]] < u_inf[x])))
# 'n_inf' es el recuento de casos que caen por debajo del umbral inferior para las variables de validación en 'var_chk'.

n <- sapply(outliers, function(x) length(x))
# 'n' es el recuento de todos los casos que caen por fuera del intervalo entre umbrales para las variables de validación en 'var_chk'.

r_sup <- n_sup / N_sup
# 'r_sup' contiene los porcentajes de casos que superan los umbrales superiores con respecto al total de casos válidos
# para las variables de validación en 'var_chk'.

r_inf <- n_inf / N_inf
# 'r_inf' contiene los porcentajes de casos que caen por debajo de los umbrales inferiores con respecto al total de casos válidos
# para las variables de validación en 'var_chk'.

r <- n / N
# 'r' contiene los porcentajes de casos que caen fuera del intervalo entre umbrales para las variables de validación en 'var_chk'.

names(u_sup) <- names(u_inf) <- names(outliers) <- var_chk
names(N_sup) <- names(N_inf) <- names(N) <- names(n_inf) <- names(n_sup) <- var_chk
names(n) <- names(r_sup) <- names(r_inf) <- names(r) <- var_chk

return(outliers)
}
```

K. Función de borrado de celdas adyacentes a una celda de coordenadas dadas (id_emp, variable).

```{r}
BorrarCeldas <- function(BDD, ind_fila, nom_variable) {
  comando1 <- paste0("ind_col <- which(names(", BDD, ") %in% '", nom_variable, "')")
  eval(parse(text = comando1))

  ind_fil <- ind_fila

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col - 2, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col - 1, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)
}
```

L. Generación de outliers para las variables de validación cuyos nombres están en el vector 'var_chk'. Después, se eliminarán de una copia del frame 'E' los casos que no ameritan ser validados.

```{r}
outliers <- Calc_Umbrales() # Llamada a generación de outliers para las variables de validación.
E2 <- E
for (j in 1:length(outliers)) {
  ind_borrar <- setdiff(1:dim(E)[1], outliers[[j]]) # Índices de fila a borrar
  sapply(1:length(ind_borrar), function(i) BorrarCeldas("E2", ind_borrar[i], names(outliers[j])))
  print(paste0("Se ha depurado con éxito la columna de reporte ", names(outliers[j])))
}
rm(j, ind_borrar)
```

M. Construcción de recuento de errores de comparación por empresa.

```{r}
err_x_emp <- apply(E2[, var_chk], 1, function(x) length(which(!is.na(x))))
err_x_var <- apply(E2[, var_chk], 2, function(x) length(which(!is.na(x))))
```

ANEXO D. Código R Markdown para la verificación de valores extremos de las variables de escala.

---
title: "SX Valores extremos 2023"
author: "Ramiro Benavides"
date: "22/11/2021"
update: "30/12/2024"
output: html_document
---

1. Carga de la BDD.

```{r}
EMP_2023 <- haven::read_sav("C:/Users/Documents/EMPRESAS 2023/BDD/Empresas_AMB_2023.sav")

B <- EMP_2023[which(EMP_2023$estado == "C" & EMP_2023$efectividad == 1), ]

# Eliminar primero las empresas no efectivas (dejar únicamente a las efectivas).
B <- B[B$efectividad==1, ]

# Eliminar a las empresas fusionadas y absorbidas.
B <- B[B$fusionada!="1.7" & B$absorbida!="1.8", ]
```

2. Asignación de crítico por zonal.

```{r}
# Asignar empresas a zonal, según el nombre del crítico.
B$Zonal_DEAGA <- NA_character_
```

3. Determinación de lista de variables numéricas de escala a verificar sus valores extremos.

```{r}
sec_var_sin_trans <- c(7002, 7005, 7006, 8098, 8099, 8100, 9001, 9002, 9052, 9058, 9062, 9066, 9070, 9074, 9078, 9082, 9086, 9090, 9094, 9098, 9105, 10000, 10001, 10035)
sec_var_trans <- c(10004, seq(10065, 10275, 21), 10317, seq(10380, 10464, 21))
lista_vars_sin_trans <- paste0("v", sec_var_sin_trans)
lista_vars_trans <- paste0("t", sec_var_trans)
lista_vars_num <- c(lista_vars_trans, lista_vars_sin_trans)
```

4. Transformación de las variables numéricas de escala que requieren hacerlo.

```{r}
# Creación de variables transformadas para las pertenecientes a la lista "lista_vars_sin_trans".
for (j in 1:length(sec_var_sin_trans)) {
  B[, paste0("t", sec_var_sin_trans[j])] <- B[, paste0("v", sec_var_sin_trans[j])]
  print(paste0("Columna v", sec_var_sin_trans[j], " procesada."))
}

**# Creación de variables transformadas para las pertenecientes a la lista "lista_vars_trans".**
B$t10004 <- ifelse(B$v10003 == 1, B$v10004 * 0.00378541, B$v10004) # B$v10003 == 1 (US gal).
for (j in 2:length(sec_var_trans)) {
  for (i in 1:dim(B)[1]) {
    B[i, paste0("t", sec_var_trans[j])] <- ifelse(B[i, paste0("v", sec_var_trans[j]-1)] == 2, B[i, paste0("v", sec_var_trans[j])] * 1000, ifelse(B[i, paste0("v", sec_var_trans[j]-1)] == 3, B[i, paste0("v", sec_var_trans[j])] * 3.78541, B[i, paste0("v", sec_var_trans[j])]))
  }
  print(paste("Columna", lista_vars_trans[j], "procesada."))
}
```

5. Conversión de 0's en missing y transformación de todas las variables en sus propios logaritmos decimales.

```{r}
l1 <- c(paste0("t", sec_var_sin_trans), paste0("t", sec_var_trans))
for (v in 1:length(l1)) {
  B[l1[v]][B[l1[v]] == 0] <- NA
  B[l1[v]] <- log10(B[l1[v]])
  print(paste("Columna", l1[v], "procesada."))
```

```
}
```

6. Función "asignar_grupos()" para segmentar por tamaño el análisis de caja.

```{r}
asignar_grupos <- function (variable) {

tryCatch(rm(caja), error=function(e){}, warning=function(w){})
comando <- paste0("caja <- robustbase::adjbox(", variable, "[is.finite(", variable, ")] ~ B$inec_tamanio[is.finite(", variable, ")], main = 'Caja ajustada [log(", variable, ")]')")
eval(parse(text = paste0(comando)), envir=.GlobalEnv)

tryCatch(rm(vec), error=function(e){}, warning=function(w){})
comando <- paste0("vec <- matrix(nrow=dim(B)[1], ncol=1)")
eval(parse(text = paste0(comando)), envir=.GlobalEnv)

comando <- paste0("grupos <- caja$group[!duplicated(caja$group)]") # Grupos de los extremos hallados (1=Mediana A; 2=Mediana B; 3=Grande).
eval(parse(text = paste0(comando)), envir=.GlobalEnv)
}
```

7. Función "agregar_vector()" para agregar las nuevas variables de rango a la BDD.

```{r}
agregar_vector <- function(variable2) {

# Agrega la nueva variable de rango a la BDD y le da un nombre con significado de rango.
comando <- paste0("B['RG", substr(variable2, 4, nchar(variable2)), "'] <- vec")
eval(parse(text = paste0(comando)), envir=.GlobalEnv)
}
```

8. Generación de las variables de reporte de control de extremos "RGxxx" en la base "B".

```{r}
require(robustbase) # Biblioteca que incluye la función adjbox()

for (v in 1:length(l1)) {
  nom_variable <- paste0("B$", l1[v])
  comando <- paste0("condicion <- (dim(B)[1] == as.numeric(summary(", nom_variable, ")[7]))")
  eval(parse(text = comando), envir=.GlobalEnv)
  if (!condicion) {
    asignar_grupos(nom_variable)
    bandera <- paste0("(", nom_variable, " %in% caja$out == TRUE) & B$inec_tamanio == h+2")
    expres_menor <- paste0(nom_variable, "[which(", bandera, ")] < caja$stats[1, h]")
    expres_mayor <- paste0(nom_variable, "[which(", bandera, ")] > caja$stats[5, h]")
    for (h in grupos) {
      if (caja$stats[1, h] < caja$stats[5, h]) {
        comando <- paste0("vec[", bandera, "][", expres_menor, "] <- 'INF'")
        eval(parse(text = comando), envir=.GlobalEnv)
        comando <- paste0("vec[", bandera, "][", expres_mayor, "] <- 'SUP'")
        eval(parse(text = comando), envir=.GlobalEnv)
      }
    }
  } else {
      comando <- paste0("vec <- matrix(nrow=dim(B)[1], ncol=1)")
      eval(parse(text = comando), envir=.GlobalEnv)
  }
  agregar_vector(nom_variable)
  print(paste0("Columna RG",substr(l1[v], 2, nchar(l1[v])), " añadida a la BDD."))
  rm(bandera, caja, comando, condicion, expres_mayor, expres_menor, grupos, h, nom_variable, v, vec)
}
```

9. Generación de la matriz de reporte de valores extremos por empresa.

```{r}
lista_vars_identif <- c("inec_identificador_empresa",  "Zonal_DEAGA", "inec_tamanio", "ciiu4_actividad_secundaria", "novedad",
                        "efectividad")
```

```
lista_vars_extremos <- names(B)[grep("^RG", names(B))]

todas_las_vars <- c(lista_vars_identif, lista_vars_extremos)

reporte <- subset(B, select = todas_las_vars)

SUP <- apply(reporte[, lista_vars_extremos], 1, function(x) {length(which(x == 'SUP'))})
INF <- apply(reporte[, lista_vars_extremos], 1, function(x) {length(which(x == 'INF'))})
EXT <- apply(reporte[, lista_vars_extremos], 1, function(x) {length(which(x == 'SUP' | x == 'INF'))})

reporte <- cbind(reporte, SUP, INF, EXT) # Agregar las nuevas columnas de recuento de supremos, ínfimos y extremos al frame de reporte.
```

10. Exportación a Excel del frame de reporte.

```{r}
writexl::write_xlsx(reporte,"C:/Users/Lenovo/Documents/EMPRESAS_2023/BDD_16-12-2023/Reporte_Extremos_2023.xlsx")
```

Elaborado por	Ramiro Benavides	Firmado electrónicamente por: RAMIRO VICENTE BENAVIDES LEON Validar únicamente con FirmaEC
Revisado por	Carlos Pilataxi	Firmado electrónicamente por: CARLOS PATRICIO PILATAXI GUAIRACAJA Validar únicamente con FirmaEC
Aprobado por	Armando Salazar	Firmado electrónicamente por: ARMANDO DAVID SALAZAR MENDEZ Validar únicamente con FirmaEC

INEC

Buenas cifras,
mejores vidas

@InecEcuador

@ecuadorencifras

@ecuadorencifras

INECEcuador