República del Ecuador

Reporte de validaciones de bases de datos

Módulo de Información Económica Ambiental de la Encuesta Estructural Empresarial (ENESEM), 2022

Abril, 2024

INEC | Buenas cifras, mejores vidas

Tabla de Contenido

Introducción

Las operaciones estadísticas del Instituto Nacional de Estadística y Censos (INEC) siguen un modelo de producción estándar que sistematiza 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. 5).

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 serán aplicables a una determinada operación estadística, dependiendo de la naturaleza de la misma.

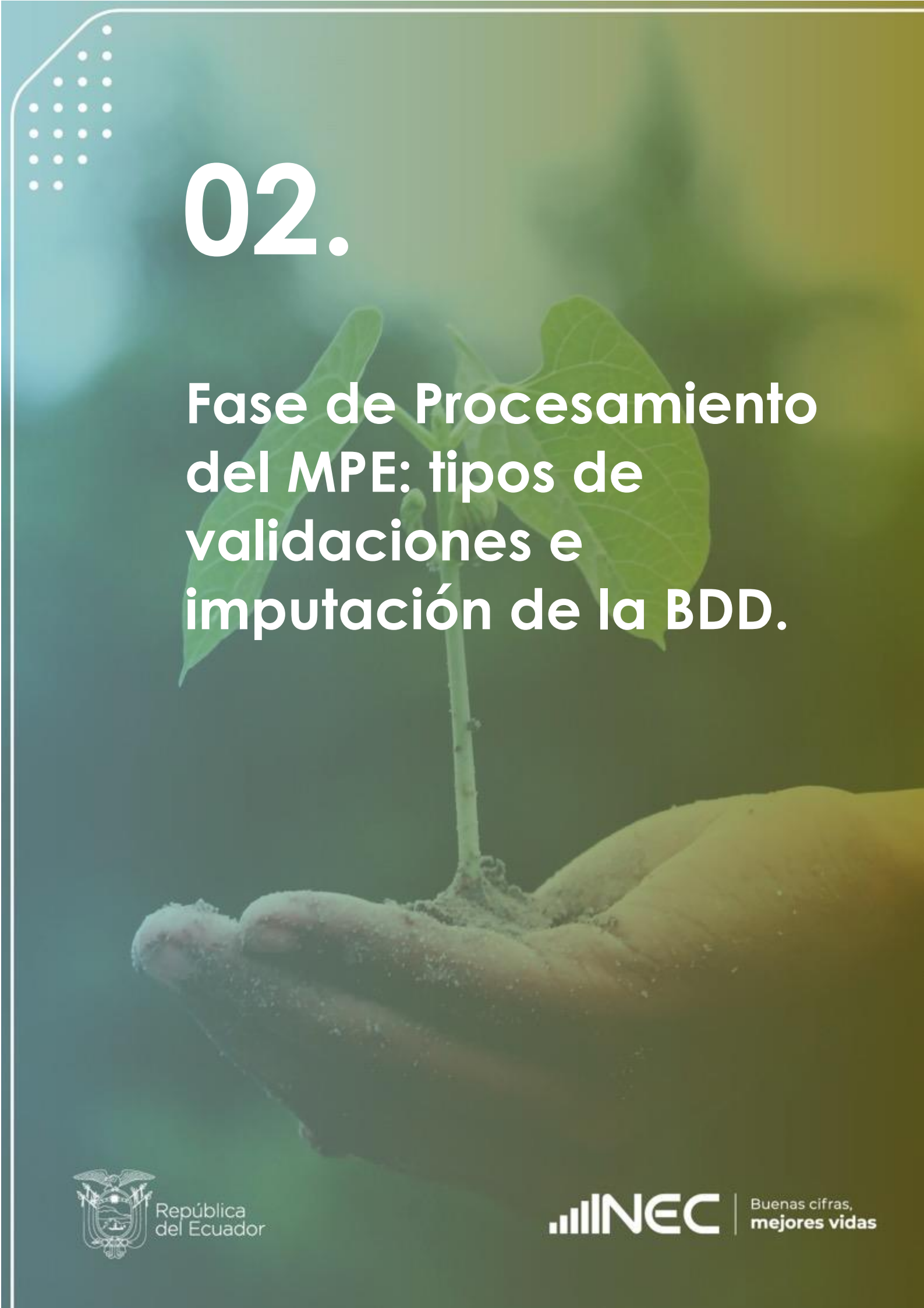
En el caso particular del Módulo de Información Ambiental Económica en Empresas (ENESEM), edición 2022, todas las fases designadas por el MPE se aplicaron a dicho módulo. En particular, en la fase de Procesamiento, existen 7 subprocesos que configuran esta fase: (1) Criticar e integrar la base de datos; (2) Clasificar y codificar; (3) Validar e imputar; (4) Derivar nuevas variables y unidades; (5) Ajustar los factores de expansión; (6) Tabular y generar indicadores; y (7) Finalizar los archivos de datos.

El presente reporte brinda información sobre las tareas y resultados obtenidos en el subproceso ***(3) Validar e imputar*** aplicados al Módulo de Información Ambiental Económica de la Encuesta Estructural Empresarial (ENESEM), edición 2022.

[1] INEC, 2016. Modelo de Producción Estadística del Ecuador. URL: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Sistema_Estadistico_Nacional/Normativas_y_Estandares/Documento_del_Modelo_de_Produccion_Estadistica.pdf; accedido el 18 de abril de 2024.

02.

Fase de Procesamiento del MPE: tipos de validaciones e imputación de la BDD.



En la fase de Procesamiento del Modelo de Producción Estadística,

> "…*se ejecutan actividades de crítica, codificación, digitalización y validación de los datos recolectados, así como la generación de los tabulados e indicadores de la operación. También se llevan a cabo métodos para proteger y salvaguardar la seguridad de los datos en cada uno de los procesos de esta fase. Para los resultados estadísticos producidos regularmente, esta fase se produce en cada iteración*" (p. 18).

Todos los procesos y actividades de esta fase:

> "…*se ejecutarán acorde a las buenas prácticas de los principios de: i) Procedimientos Estadísticos Adecuados, ii) Precisión y Confiabilidad, y iii) Coherencia y Comparabilidad, contenidos en el Código de Buenas Prácticas Estadísticas*". (p. 19).

De toda la fase de Procesamiento del MPE, quizá el proceso de mayor importancia es el de ***Validación e Imputación***. Según el Modelo de Producción Estadística:

> "*Este proceso está encaminado al tratamiento de errores en la información y de ser necesario solventar la información faltante o datos perdidos. Aquí se toman en cuenta los criterios establecidos en el plan de validación y de imputación los cuales fueron elaborados en la fase de diseño (2.6 Diseñar el procesamiento y análisis)*". (p. 20).

Con respecto a las tareas que típicamente suelen realizarse en el proceso de Validación e Imputación, el Modelo de Producción Estadística prescribe que:

> "*En la validación se examinan los datos para tratar de identificar los posibles problemas, errores y discrepancias, como los valores atípicos, la falta de respuesta y el error de codificación. Este proceso se puede ejecutar de forma iterativa o mediante la validación de los datos contra reglas predefinidas en un orden establecido. La validación se aplica a los datos, independiente de su fuente u origen, antes y después de la integración*". (p.20).

El MPE menciona que algunas o todas las tareas del proceso de Validación e Imputación se pueden efectuar en paralelo, a lo largo del desarrollo de las actividades de la fase previa de Recolección de Datos.

Con respecto a la imputación de los datos incorrectos, no verificables o faltantes,

> "*En caso de que se requiera, se pueden emplear métodos de imputación de manera rigurosa, sustentada y documentada. Si los datos se consideran incorrectos o no fiables, se pueden remplazar por nuevos valores calculados a través de métodos estadísticos robustos y probados. Existe una variedad de métodos para imputar y a menudo se utiliza un enfoque basado en reglas*". (p. 20).

En el caso del Módulo de Información Ambiental Económica de la ENESEM 2022, se han implementado no solamente uno, sino varios tipos de validaciones con los cuales se intenta maximizar el grado de 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 2022 son los siguientes:

1. ***Validación estándar (o lógica):*** Por la cual 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 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. En el Anexo B 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.

3. ***Validaciones especiales:*** Por las cuales se determina cuáles empresas no han ingresado la observación obligatoria debida al registro del código 3 ("Otros usos") para la energía eléctrica complementaria generada y consumida, así como para los combustibles y lubricantes utilizados. También abarca la revisión manual de observaciones no pertinentes a los otros usos de energía y combustibles. En el Anexo C se adjunta el código R que ejecuta este subproceso.
4. ***Validaciones de comparaciones de valores 2022 con valores 2021:*** En esta validación se suele calcular la variación interanual relativa (2022-2022) para todas las variables escalares del módulo. Luego, se procede a aplicar el logaritmo decimal a estas variaciones interanuales, con el fin de minimizar el efecto de "apreciación magnificada" que ocurre cuando existen variaciones en varios órdenes de magnitud. A los valores extremos superiores de esta

distribución logaritmada de variaciones interanuales se las marca para ser enviadas a Crítica, con el fin de comprobar si efectivamente la variación de valores de la variable bajo análisis es correcta, o si se deben corregir valores, o finalmente ratificar los altos niveles de variación interanual con una justificación válida. En el Anexo D se adjunta el código R que ejecuta este subproceso.

Esta validación solía 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.

Sin embargo, 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.

5. ***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 ratificar los valores ingresados con la correspondiente justificación. En el Anexo E se adjunta el código R que ejecuta este subproceso.

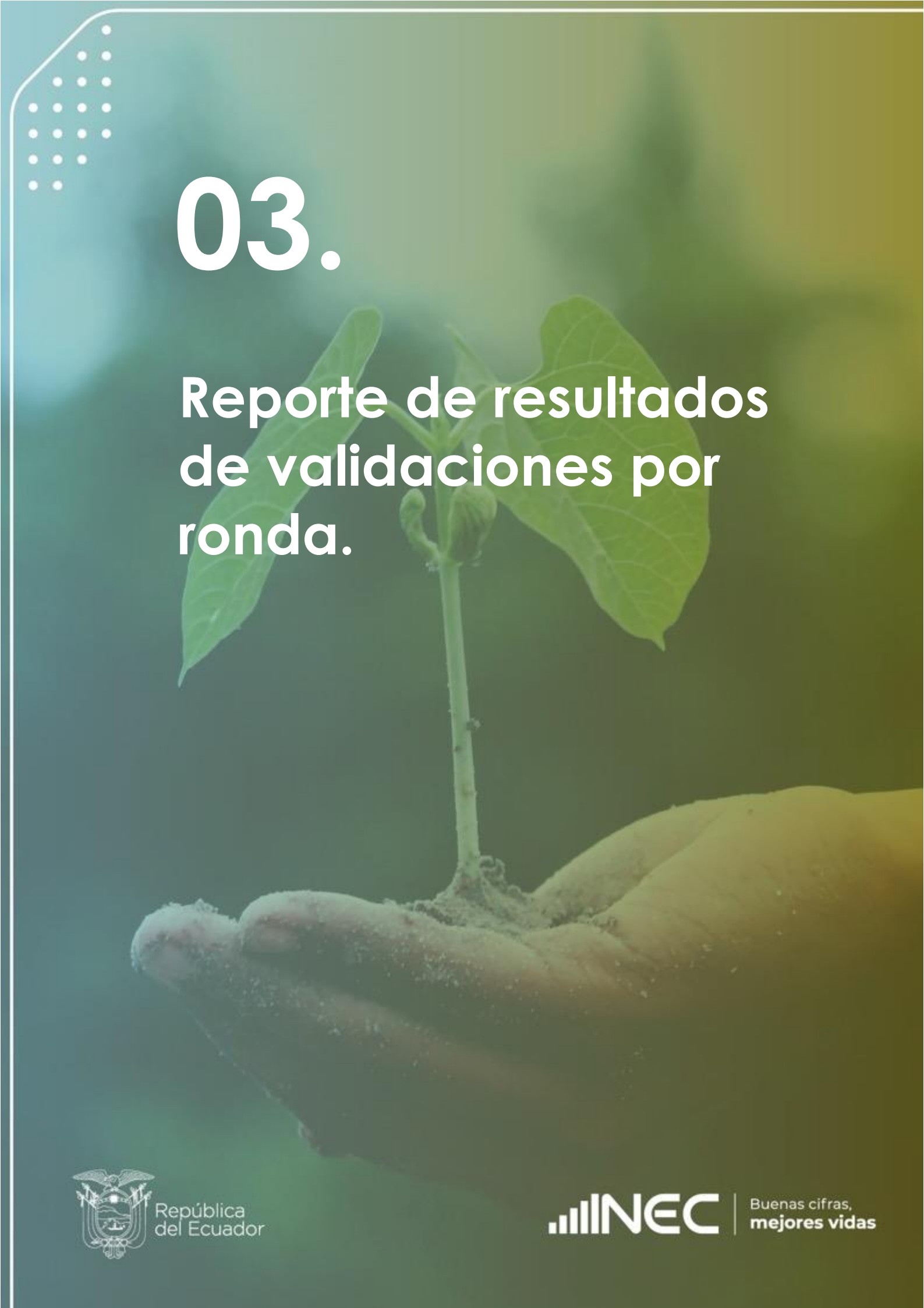
   Esta validación suele realizarse, generalmente, en la última ronda de validaciones, debido a que se dispone de la mayor muestra posible para estimar la distribución teórica de probabilidades de las variables y, por tanto, se disminuye la varianza y se minimiza la cantidad de "falsos positivos" que

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 2022, se procede a reportar los resultados encontrados para estas validaciones en los diferentes cortes de la base de datos, correspondientes a las cinco rondas de validación efectuadas.

03.

Reporte de resultados de validaciones por ronda.



República del Ecuador

INEC Buenas cifras, mejores vidas

Los resultados de las validaciones lógicas, según los cinco cortes de las bases de datos correspondientes a las rondas de validación planificadas, fueron los siguientes:

1. **Corte #1 al 31-07-2023:** De un universo de 806 empresas efectivas y criticadas, 240 empresas tuvieron al menos un error de validación lógica. Estas 240 empresas se distribuyen por zonal como: 37=Zonal Centro; 97=Zonal DICA; 68=Zonal Litoral; 38=Zonal Sur. Entre todas las zonales, se generaron 143 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control[2].
2. **Corte #2 al 14-08-2023:** De un universo de 1232 empresas efectivas y criticadas, 323 empresas[3] tuvieron al menos un error de validación lógica. Estas 323 empresas se distribuyen por zonal como: 36=Zonal Centro; 132=Zonal DICA; 102=Zonal Litoral; 53=Zonal Sur. Entre todas las zonales, se generaron 190 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
3. **Corte #3 al 28-08-2023:** De un universo de 1725 empresas efectivas y criticadas, 137 empresas tuvieron al menos un error de validación lógica. Estas 137 empresas se distribuyen por zonal como: 12=Zonal Centro; 33=Zonal DICA; 38=Zonal Litoral; 54=Zonal Sur. Entre todas las zonales, se generaron 113 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
4. **Corte #4 al 25-09-2023:** De un universo de 2579 empresas efectivas y criticadas, 104 empresas tuvieron al menos un error de validación lógica. Estas 104 empresas se distribuyen por zonal como: 7=Zonal Centro; 55=Zonal DICA; 17=Zonal Litoral; 25=Zonal Sur. Entre todas las zonales, se generaron 54 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
5. **Corte #5 al 02-12-2023:** De un universo de 3015 empresas efectivas y criticadas, 55 empresas tuvieron al menos un error de validación lógica. Estas 55 empresas se distribuyen por zonal como: 3=Zonal Centro; 19=Zonal DICA; 18=Zonal Litoral;

[2] Correspondientes a 632 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 13% y el 29% de las empresas a ser validadas.

15=Zonal Sur. Entre todas las zonales, se generaron 53 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.

6. **Corte #6 al 16-10-2023:** De un universo de 3473 empresas efectivas y criticadas, 152 empresas tuvieron al menos un error de validación lógica. Estas 152 empresas se distribuyen por zonal como: 6=Zonal Centro; 91=Zonal DICA; 21=Zonal Litoral; 34=Zonal Sur. Entre todas las zonales, se generaron 124 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
7. **Corte #7 al 06-11-2023:** De un universo de 4013 empresas efectivas y criticadas, 54 empresas tuvieron al menos un error de validación lógica. Estas 54 empresas se distribuyen por zonal como: 1=Zonal Centro; 26=Zonal DICA; 11=Zonal Litoral; 16=Zonal Sur. Entre todas las zonales, se generaron 165 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
8. **Corte #8 al 04-12-2023:** De un universo de 4307 empresas efectivas y criticadas, 3424 empresas tuvieron al menos un error de combustibles fuera de rango. En esta ronda de validaciones, no se ejecutaron validaciones lógicas; únicamente se realizaron comprobaciones de costos unitarios de combustibles, los cuales debían estar en un determinado rango de valores válidos. Estas 3424 empresas se distribuyen por zonal como: 168=Zonal Centro; 1284=Zonal DICA; 1554=Zonal Litoral; 418=Zonal Sur. Entre todas las zonales, se generaron 11 variables de control de rangos válidos de combustibles.
9. **Corte #9 al 11-12-2023:** De un universo de 4307 empresas efectivas y criticadas, 36 empresas tuvieron al menos un error de validación lógica. Estas 36 empresas se distribuyen por zonal como: 0=Zonal Centro; 12=Zonal DICA; 16=Zonal Litoral; 8=Zonal Sur. Entre todas las zonales, se generaron 36 variables de control con al menos un error a verificar, de un total teórico de 632 variables de control.
10. **Corte #10 al 18-12-2023:** De un universo de 4307 empresas efectivas y criticadas, 0 empresas tuvieron errores de validación lógica ambientales. Esta ronda de validación no se la realizó, ya que no existían empresas para aplicar validaciones lógicas. El corte de BDD realizado sirvió para realizar algunos ajustes y/o imputaciones exclusivamente a las variables del Módulo Económico.
11. **Corte #11 al 26-12-2023:** De un universo de 4307 empresas efectivas y criticadas, 0 empresas tuvieron errores de validación lógica ambientales. Esta ronda de validación no se la realizó, ya que no existían empresas para aplicar validaciones lógicas. El corte de BDD realizado sirvió para realizar algunos ajustes y/o imputaciones exclusivamente a las variables del Módulo Económico.

12. **Corte #12 al 02-01-2024:** De un universo de 4307 empresas efectivas y criticadas, 0 empresas tuvieron errores de validación lógica ambientales. Esta ronda de validación no se la realizó, ya que no existían empresas para aplicar validaciones lógicas. El corte de BDD realizado sirvió para realizar algunos ajustes y/o imputaciones exclusivamente a las variables del Módulo Económico.
13. **Corte #13 al 08-01-2024:** De un universo de 4307 empresas efectivas y criticadas, 0 empresas tuvieron errores de validación lógica ambientales. Esta ronda de validación no se la realizó, ya que no existían empresas para aplicar validaciones lógicas. El corte de BDD realizado sirvió para realizar algunos ajustes y/o imputaciones exclusivamente a las variables del Módulo Económico.
14. **Corte #14 al 22-01-2024:** De un universo de 4307 empresas efectivas y criticadas, 0 empresas tuvieron errores de validación lógica ambientales. Esta ronda de validación no se la realizó, ya que no existían empresas para aplicar validaciones lógicas. El corte de BDD realizado sirvió para realizar algunos ajustes y/o imputaciones exclusivamente a las variables del Módulo Económico.

De las 4307 empresas que conformaron la muestra ENESEM 2022 previa a la fase de levantamiento de información, permanecieron 3951 empresas en la BDD de publicación. Las 356 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 2022.

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 2022. 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 2022:

Nombre de variable	v7005	v8087	v8088	v8091	v8092	v8093	v8096	v8098	v8100
Recuento de cambios realizados	1	2	3	1	1	2	1	5	1

Nombre de variable	v9i2	v9034	v9037	v9038	v9041	v9052	v9053	v9054	v9055
Recuento de cambios realizados	5	1	2	1	1	3	1	1	1

Nombre de variable	v9056	v9ii1	v10006	v10014	v10016	v10022	v10032	v10035
Recuento de cambios realizados	1	3	3	1	1	5	1	1

TABLA 1. Registro de cambios realizados a datos de la BDD de prepublicación.
Fuente: Módulo de Información Económica Ambiental de la ENESEM 2022.

Se realizaron 49 cambios de información puntuales a las variables del Módulo de Información Económica Ambiental de la ENESEM 2022. El código R que realiza esta tarea aparece en el Anexo B del presente documento.

En el tema de las validaciones especiales, normalmente se encontraban entre 0 y N, donde N es un valor menor que 22 empresas, para enviar a revisión a las zonales en cada ronda de validación. Este tipo de validación afecta solamente a algunas variables de generación de energía complementaria y su uso principal, así como las de generación de combustibles y su uso principal. El código R que realiza esta tarea aparece en el Anexo C del presente documento.

Para las validaciones de comparaciones de valores de las variables ambientales entre los años 2021 y 2022, se controlaron 39 variables del Módulo Ambiental. Los resultados de este tipo de validaciones por comparación, según los cortes 1ro. hasta 9no. de las bases de datos correspondientes a las rondas de validación planificadas, fueron las siguientes:

1. **Corte #1 al 31-07-2023:** De un universo de 638 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 583 empresas tuvieron al menos una variable con una variación interanual mayor al

umbral establecido[4]. Estas 583 empresas se distribuyen por zonal como: 49=Zonal Centro; 303=Zonal DICA; 150=Zonal Litoral; 81=Zonal Sur. En total, se generaron 2453 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 165=Zonal Centro; 1359=Zonal DICA; 625=Zonal Litoral; 304=Zonal Sur.

2. **Corte #2 al 14-08-2023:** De un universo de 972 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 821 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 821 empresas se distribuyen por zonal como: 54=Zonal Centro; 394=Zonal DICA; 244=Zonal Litoral; 129=Zonal Sur. En total, se generaron 3023 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 155=Zonal Centro; 1557=Zonal DICA; 894=Zonal Litoral; 417=Zonal Sur.
3. **Corte #3 al 28-08-2023:** De un universo de 1307 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 1238 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 1238 empresas se distribuyen por zonal como: 81=Zonal Centro; 546=Zonal DICA; 404=Zonal Litoral; 207=Zonal Sur. En total, se generaron 5603 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 286=Zonal Centro; 2641=Zonal DICA; 1842=Zonal Litoral; 834=Zonal Sur.
4. **Corte #4 al 25-09-2023:** Para esta ronda de validaciones no se realizaron control de validaciones de comparaciones ambientales, ya que las Coordinaciones

[4] Este umbral se había fijado en 30% para las variables de escala del Módulo Ambiental de la ENESEM 2020, después de hacer un análisis descriptivo de las distribuciones de las variaciones interanuales 2019-2020 de todas sus variables escalares. Sin embargo, para el levantamiento de la ENESEM 2021 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 2020-2021. 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 2020-2021 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 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 2020-2021, 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 2022.

Zonales mencionaron que no habían concluido con las tareas de validaciones de la tercera ronda de validaciones.

5. **Corte #5 al 02-10-2023:** De un universo de 2239 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 648 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 648 empresas se distribuyen por zonal como: 18=Zonal Centro; 362=Zonal DICA; 221=Zonal Litoral; 47=Zonal Sur. En total, se generaron 1900 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 50=Zonal Centro; 1038=Zonal DICA; 671=Zonal Litoral; 141=Zonal Sur.
6. **Corte #6 al 16-10-2023:** De un universo de 2571 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 690 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 690 empresas se distribuyen por zonal como: 17=Zonal Centro; 282=Zonal DICA; 330=Zonal Litoral; 61=Zonal Sur. En total, se generaron 2213 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 26=Zonal Centro; 952=Zonal DICA; 1075=Zonal Litoral; 160=Zonal Sur.
7. **Corte #7 al 06-11-2023:** De un universo de 2897 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 1504 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 1504 empresas se distribuyen por zonal como: 33=Zonal Centro; 651=Zonal DICA; 706=Zonal Litoral; 114=Zonal Sur. En total, se generaron 3497 novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 46=Zonal Centro; 1545=Zonal DICA; 1666=Zonal Litoral; 240=Zonal Sur.
8. **Corte #8 al 04-12-2023:** Para esta ronda de validaciones no se realizaron control de validaciones de comparaciones ambientales, ya que las Coordinaciones Zonales se dedicaron a la tarea de validaciones de rangos de combustibles y lubricantes.
9. **Corte #9 al 11-12-2023:** De un universo de 1512 empresas efectivas y criticadas que resultan de la intersección entre las BDD 2022 y 2021 (publicada), 124 empresas tuvieron al menos una variable con una variación interanual mayor al umbral establecido. Estas 124 empresas se distribuyen por zonal como: 0=Zonal Centro; 12=Zonal DICA; 111=Zonal Litoral; 1=Zonal Sur. En total, se generaron 340

novedades a verificar para las 39 variables de control establecidas, distribuidas por zonal como: 0=Zonal Centro; 37=Zonal DICA; 302=Zonal Litoral; 1=Zonal Sur.

Finalmente, con respecto a la verificación de valores extremos efectuada sobre el 10mo. corte de BDD al 18 de diciembre de 2022, se obtuvo los siguientes resultados:

- Se tomó como universo a las 4307 empresas[5] validadas y criticadas hasta la 10ma. ronda de validación. El universo de variables a controlar abarca a 42 variables escalares[6] como ingresos, gastos, cantidad de energía, cantidad de residuos generada, etc. Se detectaron 547 empresas con al menos una de las 42 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 24 empresas de 176 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 12 supremos, 14 ínfimos y 26 extremos en total. Se detectaron valores extremos en 17 de las 42 variables escalares a validar.
    - Zonal DICA: Hubo 245 empresas de 1543 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 83 supremos, 164 ínfimos y 247 extremos en total. Se detectaron valores extremos en 30 de las 42 variables escalares a validar.
    - Zonal Litoral: Hubo 224 empresas de 1760 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 72 supremos, 156 ínfimos y 228 extremos en total. Se detectaron valores extremos en 27 de las 42 variables escalares a validar.

[5] No obstante, se aplicó el algoritmo de valores extremos a las 3950 empresas de publicación de la BDD del Módulo Ambiental de la ENESEM 2022.

[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 681 variables originales sobre el Impacto Ambiental Agregado 2020. Esta reducción de la carga de empresas a validar por las zonales, para este tipo de validación, se constituye en otra innovación técnica para la ENESEM 2022.

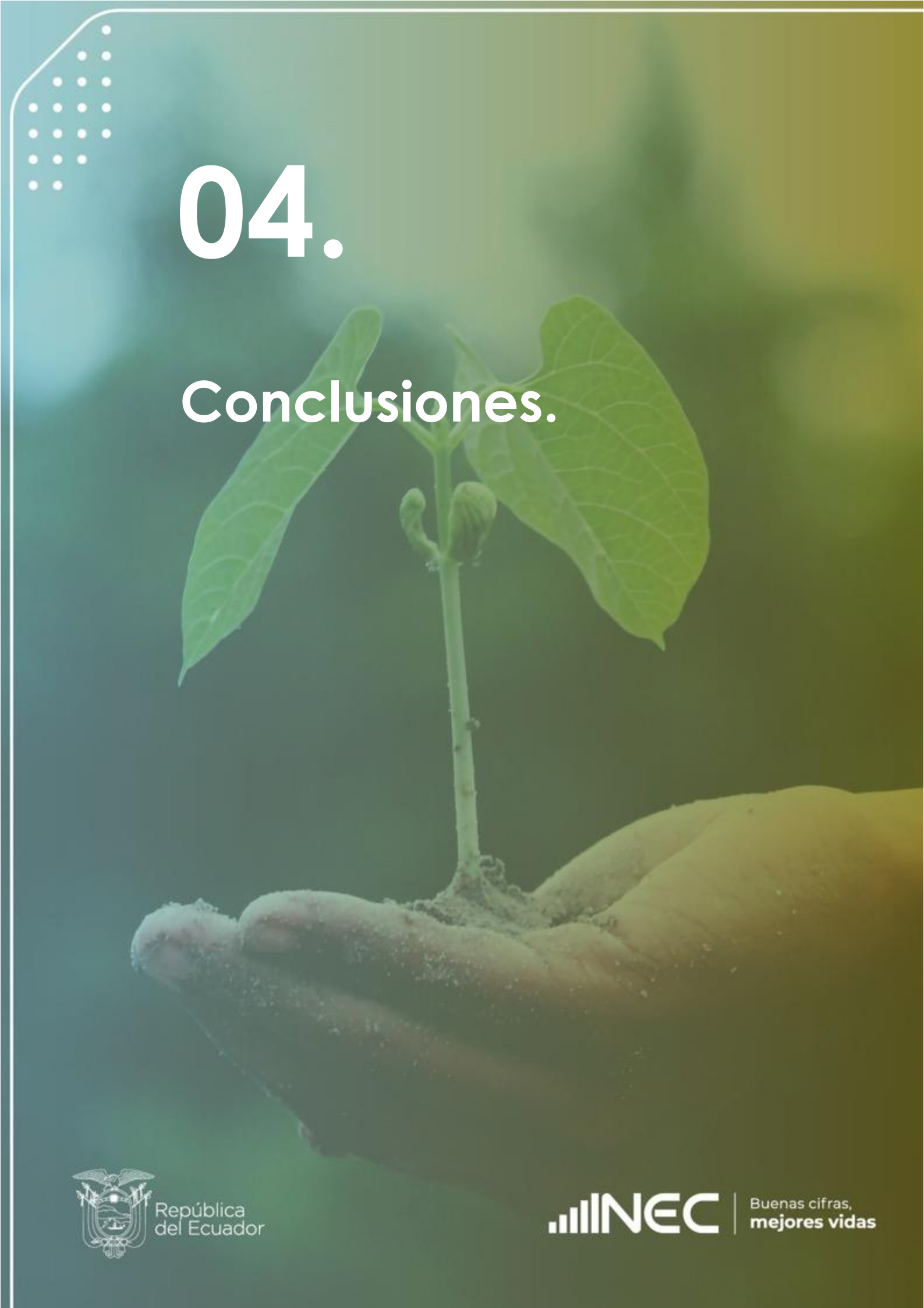
[7] Esto implica que, por ejemplo, para la variable ***v7001*** no existe únicamente una distribución de los valores logaritmados de esa variable, sino tres: uno para c/u de los tamaños de empresa (Mediana A, Mediana B y Grande). Si una empresa sobrepasa los umbrales por tamaño, determinados por el algoritmo 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 Sur: Hubo 54 empresas de 471 publicables de esta zonal con al menos un valor extremo a ser revisado. Se generaron 25 supremos, 30 ínfimos y 55 extremos en total. Se detectaron valores extremos en 19 de las 42 variables escalares a validar.

Cabe mencionar que todas estas 547 empresas de las 4307 con potencial de ser publicables fueron justificadas en su totalidad sobre sus valores extremos. Aproximadamente el 64% 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, la fase de Validación e Imputación del Modelo de Producción Estadística es crucial para el aseguramiento de la calidad de la información recolectada en campo para una operación estadística.

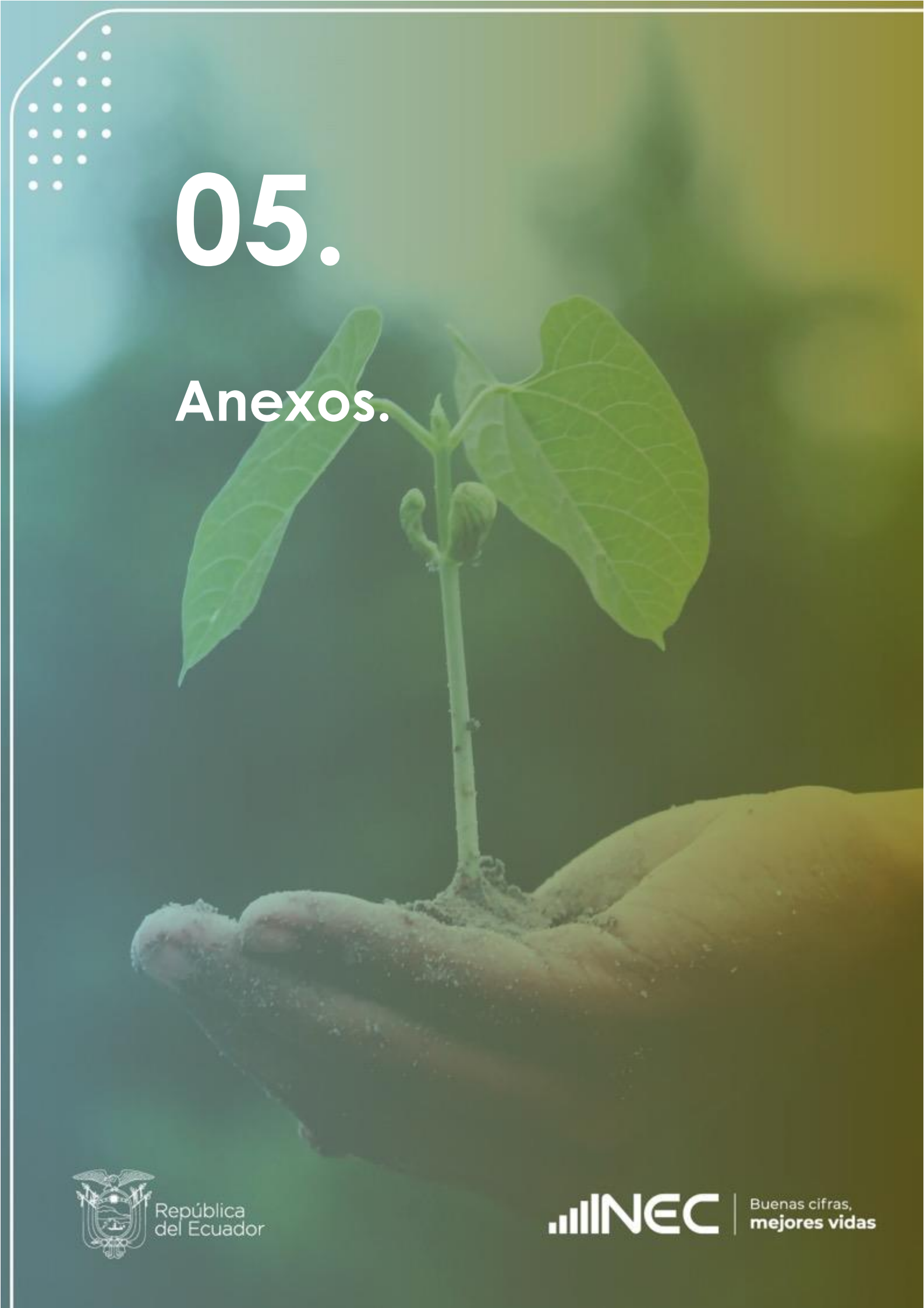
En el caso de la fase de Validación e Imputación del Módulo de Información Ambiental Económica de la ENESEM 2022, se la ha implementado desde varias perspectivas, cada una de las cuales minimiza la cantidad de errores de levantamiento, garantizando así la mejor calidad de la información disponible para los usuarios de esta operación estadística.

La implantación continua de los distintos tipos de validación que se utilizan para validar el Módulo de Información Ambiental Económica de la ENESEM 2022 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 4.1.2, 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.

05.

Anexos.



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

```
---
title: "Validacion Logica ENESEM 2022"
author: "Ing. Ramiro Benavides Leon"
date: "28/08/2020"
update: "27/03/2023"
output: html_document
---
```

A. Carga de la base de datos en alguno de los formatos CSV o Excel.

```{r}
winDialog("ok", "Para ejecutar la validación del Módulo Ambiental de la ENESEM 2022, se podrá hacerlo únicamente si se carga la BDD en formatos CSV o Excel. A continuación se le dará la opción de escoger la BDD a validar en uno de estos dos formatos...")
# formato <- svDialogs::dlgInput("Presione 1 -> Formato CSV \nPresione 2 -> Formato Excel ", Sys.info()["user"])$res
Nom_BDD <- file.choose() # Se carga el fullpathname del archivo que contiene la BDD a validar.
if (isTRUE(grepl(".csv", Nom_BDD))) {
  require(readr)
  EMP_2022 <- read_delim(Nom_BDD, delim = ";", escape_double = FALSE, col_types = cols(inec_identificador_empresa = col_character()), locale = locale(date_names = "es", encoding = "WINDOWS-1252", asciify = TRUE), trim_ws = TRUE) # EMP_2022 es el frame espejo de la BDD SPSS a validar.
  detach(package:readr, unload = T)
} else if (isTRUE(grepl(".xls", Nom_BDD))) {
  EMP_2022 <- readxl::read_excel(Nom_BDD) # EMP_2022 es el frame espejo de la BDD SPSS a validar.
} else {
  winDialog("ok", "No se ha escogido una BDD del Módulo Ambiental ENESEM 2022 en formato CSV o Excel. Favor volver a ejecutar el bloque de código de carga de la BDD.")
}

# Eliminar las empresas no criticadas y las no efectivas (dejar únicamente a las efectivas y criticadas).
# El frame B es el que contendrá las empresas que efectivamente se validarán.
B <- EMP_2022[which(EMP_2022$estado == "C" & EMP_2022$efectividad == 1), ]

# Eliminar a las empresas fusionadas o absorbidas.
B <- B[-which(!((is.na(B$fusionada) | B$fusionada == "") & (is.na(B$absorbida) | B$absorbida == ""))), ]

# Eliminación de objetos temporales.
rm(EMP_2022, Nom_BDD)
```

```{r}
# Asignar empresas a zonal, según el nombre del crítico. Por el momento, el código escribe
# la cadena constante "INEC". Cuando se entregue el archivo con la lista de empresas a validar,
# durante cada ronda de validación, se escribirá aquí el código de selección de las empresas
# a validar en el frame B desde el archivo que contiene la lista de empresas a validar.
B$Zonal_DEAGA <- "INEC"
```

```{r}
# Se crean las columnas B$UNO y B$CIEN, principalmente para validar porcentajes.
B$UNO <- 1
B$CIEN <- 100
```

B. Creación de filtro "Efectivas" para filtrar los casos con errores.

Este bloque de código estará vacío mientras no haya discrepancia entre las empresas marcadas como "Efectivas" en la respectiva variable de cobertura, y las empresas que se entreguen en la lista de empresas a validar en un corte determinado. En caso de discrepancia, se escribirá aquí un código que gestione esta discrepancia y determine exactamente a las empresas que deberán entrar al proceso de validación.

```{r}
```

C. Crear el dataframe con las variables de identificación, el cual irá agregando después las variables de validación. Este frame se enviará a las zonales para su validación.

```{r}
lista_vars_identif <- c("secuencial_sistema", "inec_identificador_empresa", "Zonal_DEAGA", "Estado", "ciiu4_actividad_principal", "observaciones", "observaciones_investigador", "observaciones_critica")
```

```
# El frame G es el frame de trabajo en donde se guardarán las variables de control que se vayan
# generando durante el proceso de validación estándar, esto es, acorde a la Malla de Validación.
# El frame G se inicializa con las variables identificadoras definidas en el vector lista_vars_identif.
G <- subset(B, select = lista_vars_identif)
G_orig <- G # G_orig es un respaldo del frame G antes de la ejecución de las validaciones.
```

D. Funciones utilitarias.

```{r}
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)
}

AddRule <- function(label, regla, filtro="") {
  comando1 <- paste0("K <- dim(B)[1]")
  comando2 <- paste0("K <- length(which(", filtro, "))")
  comando <- ifelse(filtro == "", comando1, comando2)
  eval(parse(text = comando), envir = .GlobalEnv)
  M <- get0("K")
  r <- get0("G")
  comando1 <- paste0("r$", label, " <- ifelse(!(", regla, "), 1, NA)")
  comando2 <- paste0("r$", label, " <- ifelse(!(", regla, ") & (", filtro, "), 1, NA)")
  comando <- ifelse(filtro == "", comando1, comando2)
  eval(parse(text = comando))
  if (!(all(is.na(r[dim(r)[2]])))) {
      assign("G", r, envir = .GlobalEnv)
      N <- sum(r[dim(r)[2]], na.rm = T)
      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"))
  }
  comando <- paste0("rm(K)")
  eval(parse(text = comando), envir = .GlobalEnv)
}

AddValidSumas <- function(label, colbase, col1, ..., filtro="") {
  p <- CalcDiffCol(colbase, col1, ...)
  if (!(is.null(p))) {
    comando1 <- paste0("K <- dim(B)[1]")
    comando2 <- paste0("K <- length(which(", filtro, "))")
    comando <- ifelse(filtro == "", comando1, comando2)
    eval(parse(text = comando), envir = .GlobalEnv)
    M <- get0("K")
    r <- get0("G")
    comando1 <- paste0("r$", label, " <- ifelse(abs(p) > 0, 1, NA)")
    comando2 <- paste0("r$", label, " <- ifelse(", filtro, ", ifelse(abs(p) > 0, 1, NA), NA)")
    comando <- ifelse(filtro == "", comando1, comando2)
    eval(parse(text = comando))
    if (!(all(is.na(r[dim(r)[2]])))) {
      assign("G", r, envir = .GlobalEnv)
      N <- sum(r[dim(r)[2]], na.rm = T)
      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"))
    }
    comando <- paste0("rm(K)")
    eval(parse(text = comando), envir = .GlobalEnv)
  } else {
    print(paste0("NO se agregó la regla ", label, " al frame de validación"))
  }
}
```

E. Validación de sumas: Capítulo 7

```{r}
AddValidSumas("sum517", B$v7002, B$v7003, B$v7004)
```

E. Validación de sumas: Capítulo 8

```{r}
AddValidSumas("sum527", B$v8098, B$v8087, B$v8093)
AddValidSumas("sum528", B$v8099, B$v8089, B$v8095)
AddValidSumas("sum529", B$v8100, B$v8091, B$v8097)
```

E. Validación de sumas: Capítulo 9

```{r}
AddValidSumas("sum530", B$v9052, B$v9005, B$v9013, B$v9021, B$v9029, B$v9037, B$v9045, filtro="B$v9i2==1")
AddValidSumas("sum531", B$v9053, B$v9006, B$v9014, B$v9022, B$v9030, B$v9038, B$v9046, filtro="B$v9i2==1")
AddValidSumas("sum532", B$v9054, B$v9007, B$v9015, B$v9023, B$v9031, B$v9039, B$v9047, filtro="B$v9i2==1")
AddValidSumas("sum533", B$v9055, B$v9009, B$v9017, B$v9025, B$v9033, B$v9041, B$v9049, filtro="B$v9i2==1")
AddValidSumas("sum534", B$v9056, B$v9010, B$v9018, B$v9026, B$v9034, B$v9042, B$v9050, filtro="B$v9i2==1")
AddValidSumas("sum535", 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")
```

E. Validación de sumas: Capítulo 10. Aguas de captación y residuales.

```{r}
AddValidSumas("sum536", B$v10030, B$v10012, B$v10020, B$v10028, filtro="B$v10i3==1")
AddValidSumas("sum537", B$v10031, B$v10013, B$v10021, B$v10029, filtro="B$v10i3==1")
```

E. Validación de sumas: Capítulo 10.1 Residuos No Peligrosos.

```{r}
AddValidSumas("sum555", B$CIEN, B$v10079, B$v10080, B$v10081, filtro="B$v10073 >= 0")
AddValidSumas("sum556", B$CIEN, B$v10100, B$v10101, B$v10102, filtro="B$v10094 >= 0")
AddValidSumas("sum557", B$CIEN, B$v10121, B$v10122, B$v10123, filtro="B$v10115 >= 0")
AddValidSumas("sum558", B$CIEN, B$v10142, B$v10143, B$v10144, filtro="B$v10136 >= 0")
AddValidSumas("sum559", B$CIEN, B$v10163, B$v10164, B$v10165, filtro="B$v10157 >= 0")
AddValidSumas("sum560", B$CIEN, B$v10184, B$v10185, B$v10186, filtro="B$v10178 >= 0")
AddValidSumas("sum561", B$CIEN, B$v10205, B$v10206, B$v10207, filtro="B$v10199 >= 0")
AddValidSumas("sum562", B$CIEN, B$v10226, B$v10227, B$v10228, filtro="B$v10220 >= 0")
AddValidSumas("sum563", B$CIEN, B$v10247, B$v10248, B$v10249, filtro="B$v10241 >= 0")
AddValidSumas("sum564", B$CIEN, B$v10268, B$v10269, B$v10270, filtro="B$v10262 >= 0")
AddValidSumas("sum565", B$CIEN, B$v10289, B$v10290, B$v10291, filtro="B$v10283 >= 0")
AddValidSumas("sum567", B$CIEN, B$v10331, B$v10332, B$v10333, filtro="B$v10325 >= 0")
```

E. Validación de sumas: Capítulo 10.2 Desechos especiales.

```{r}
AddValidSumas("sum578", B$CIEN, B$v10394, B$v10395, B$v10396, filtro="B$v10388 >= 0")
AddValidSumas("sum579", B$CIEN, B$v10415, B$v10416, B$v10417, filtro="B$v10409 >= 0")
AddValidSumas("sum580", B$CIEN, B$v10436, B$v10437, B$v10438, filtro="B$v10430 >= 0")
AddValidSumas("sum581", B$CIEN, B$v10457, B$v10458, B$v10459, filtro="B$v10451 >= 0")
AddValidSumas("sum582", B$CIEN, B$v10478, B$v10479, B$v10480, filtro="B$v10472 >= 0")
AddValidSumas("sum585", B$CIEN, B$v10541, B$v10542, B$v10543, filtro="B$v10535 >= 0")
```

E. Validación de sumas: Capítulo 10.3 Desechos peligrosos.

```{r}
AddValidSumas("sum608", B$CIEN, B$v10562, B$v10563, B$v10564, filtro="B$v10556 >= 0")
AddValidSumas("sum609", B$CIEN, B$v10583, B$v10584, B$v10585, filtro="B$v10577 >= 0")
AddValidSumas("sum610", B$CIEN, B$v10604, B$v10605, B$v10606, filtro="B$v10598 >= 0")
AddValidSumas("sum611", B$CIEN, B$v10625, B$v10626, B$v10627, filtro="B$v10619 >= 0")
AddValidSumas("sum612", B$CIEN, B$v10646, B$v10647, B$v10648, filtro="B$v10640 >= 0")
AddValidSumas("sum613", B$CIEN, B$v10667, B$v10668, B$v10669, filtro="B$v10661 >= 0")
AddValidSumas("sum614", B$CIEN, B$v10688, B$v10689, B$v10690, filtro="B$v10682 >= 0")
AddValidSumas("sum615", B$CIEN, B$v10709, B$v10710, B$v10711, filtro="B$v10703 >= 0")
AddValidSumas("sum616", B$CIEN, B$v10730, B$v10731, B$v10732, filtro="B$v10724 >= 0")
AddValidSumas("sum617", B$CIEN, B$v10751, B$v10752, B$v10753, filtro="B$v10745 >= 0")
```

```
AddValidSumas("sum618", B$CIEN, B$v10772, B$v10773, B$v10774, filtro="B$v10766 >= 0")
AddValidSumas("sum619", B$CIEN, B$v10793, B$v10794, B$v10795, filtro="B$v10787 >= 0")
AddValidSumas("sum620", B$CIEN, B$v10814, B$v10815, B$v10816, filtro="B$v10808 >= 0")
AddValidSumas("sum621", B$CIEN, B$v10835, B$v10836, B$v10837, filtro="B$v10829 >= 0")
AddValidSumas("sum622", B$CIEN, B$v10856, B$v10857, B$v10858, filtro="B$v10850 >= 0")
AddValidSumas("sum623", B$CIEN, B$v10877, B$v10878, B$v10879, filtro="B$v10871 >= 0")
AddValidSumas("sum624", B$CIEN, B$v10898, B$v10899, B$v10900, filtro="B$v10892 >= 0")
AddValidSumas("sum625", B$CIEN, B$v10919, B$v10920, B$v10921, filtro="B$v10913 >= 0")
AddValidSumas("sum626", B$CIEN, B$v10940, B$v10941, B$v10942, filtro="B$v10934 >= 0")
AddValidSumas("sum627", B$CIEN, B$v10961, B$v10962, B$v10963, filtro="B$v10955 >= 0")
AddValidSumas("sum628", B$CIEN, B$v10982, B$v10983, B$v10984, filtro="B$v10976 >= 0")
AddValidSumas("sum629", B$CIEN, B$v11003, B$v11004, B$v11005, filtro="B$v10997 >= 0")
```

F. Validaciones lógicas: Capítulo 7.

```{r}
#----------------
# 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.

AddRule("val_7_001", "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.

AddRule("val_7_002", "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 o igual que VAR_7001. En caso contrario, mostrar mensaje de error.

AddRule("val_7_003", "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 error.

AddRule("val_7_004", "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.

AddRule("val_7_005", "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.

AddRule("val_7_006", "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.
```

```
AddValidSumas("val_7_007", 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 (425*12* Valor(Pregunta 7003)) y no puede ser mayor que la variable 5180.

AddRule("val_7_008", "B$v7005 >= 425*12*B$v7003 & B$v7005 <= B$v5180", "B$v7003 > 0")

#----------------
# REGLA val_7_009
#----------------
# Si el valor es menor al sueldo básico (425*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(G)[1])

cond_1[c1] <- TRUE
cond_2[c2] <- TRUE
cond_3[c3] <- TRUE

condicion <- (cond_1 | cond_2 | cond_3)

AddRule("val_7_009", "!(B$v7005 < 425*12*B$v7003) | condicion", "B$v7003 > 0")

rm(cadena_OK, cond_1, cond_2, cond_3, c1, c2, c3, 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)
AddRule("val_7_010", "z <= B$v5180", "B$v7002 > 0")
rm(z)
```

G. Validaciones lógicas: Capítulo 8.

```{r}
#----------------
# REGLA val_8_001
#----------------
# Obligatorio recuperar y desplegar información bajo la condición: v8001 = v4146 + v4147 + v4148 + v4196 + v4197.

AddValidSumas("val_8_001", B$v8001, B$v4146, B$v4147, B$v4148, B$v4196, B$v4197)
```

```{r}
#----------------------------
# 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("val_8_00", 1+(i-8084)/2)
```

```
  variable <- paste0("B$v", i)
  regla <- paste0(variable, " %in% 1:2")
  AddRule(etiqueta, regla)
}
rm(etiqueta,i,regla,seq_i,variable)
```

```{r}
#----------------------------
# 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 <- ifelse(i <= 8089, paste0("val_8_00", 7+(i-8085)/2), paste0("val_8_0", 7+(i-8085)/2))
  variable <- paste0("B$v", i)
  previa <- paste0("B$v", i-1)
  regla <- paste0(variable, " > 0")
  filtrado <- paste0(previa, "==1")
  AddRule(etiqueta, regla, filtrado)
}
rm(etiqueta,filtrado,i,previa,regla,seq_i,variable)
```

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

AddRule("val_8_014", "B$v8098 <= B$v2006", "B$v8098 > 0 & B$v2006 > 0")
```

```{r}
#----------------
# REGLA val_8_015
#----------------
# El total de la columna 4 (línea 328, VAR_8099) no puede ser superior a la suma: v4146 + v4147 + v4148 + v4196 + v4197.

z <- apply(cbind(B$v4146, B$v4147, B$v4148, B$v4196, B$v4197), 1, sum, na.rm = T)
AddRule("val_8_015", "B$v8099 <= z", "B$v8099 > 0 & z > 0")
rm(z)
```

H. Validaciones lógicas: Capítulo 9, Sección I.

```{r}
#----------------
# REGLA val_9_001
#----------------
# En la pregunta 1, llenar obligatoriamente los campos de cantidad (2) y valor (3).

filtro <- "(is.na(B$v9001) | B$v9001 == 0) | (is.na(B$v9002) | B$v9002 == 0)"
AddRule("val_9_001", "nchar(trimws(as.character(B$v9003))) > 2", filtro)
rm(filtro)

#----------------
# REGLA val_9_002
#----------------
#  En la columna (3) se aplica la siguiente condición: VAR_9002 <= VAR_1173.

AddRule("val_9_002", "B$v9002 <= B$v1173", "B$v9002 > 0 & B$v1173 > 0")

#----------------
# REGLA val_9_003
#----------------
```

```
# El Valor USD(3) dividido para la Cantidad(2) debe estar dentro del rango de 0.05 a 0.44.

AddRule("val_9_003", "B$v9002 >= 0.05*B$v9001 & B$v9002 <= 0.44*B$v9001", "B$v9001 > 0")

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

AddRule("val_9_004", "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 9057.

AddRule("val_9_005", "nchar(trimws(as.character(B$v9057))) > 2", "B$v9044==1")

#----------------
# REGLA val_9_006
#----------------
# La energía total producida (Var_9052) debe ser igual a la suma  de la energía consumida (Var_9054) + la energía vendida (Var_9055).

AddValidSumas("val_9_006", B$v9052, B$v9054, B$v9055, filtro="B$v9i2==1")

#----------------
# REGLA val_9_007
#----------------
# En la columna 2 debe existir al menos un "Sí" (VAR_9004, VAR_9012, VAR_9020, VAR_9028, VAR_9036 o VAR_9044).

AddRule("val_9_007", "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 al menos una de las variables: v9004, v9012, v9020, v9028, v9036, v9044 es igual a 1, entonces la Pregunta 2 debe responder que "SI".

AddRule("val_9_008", "B$v9i2==1", "B$v9004==1 | B$v9012==1 | B$v9020==1 | B$v9028==1 | B$v9036==1 | B$v9044==1")
```

```{r, warning=FALSE}
#-----------------------------
# 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, min, na.rm = T)
seq_i <- seq(9004, 9044, 8)
for (i in seq_i) {
  etiqueta <- ifelse(i == 9004, paste0("val_9_00", 8+(i-8996)/8), paste0("val_9_0", 8+(i-8996)/8))
  variable <- paste0("B$v", i)
  secuencia <- sapply(1:6, function(j) paste0("v", i + j))
  All_NA_0 <- apply(B[, secuencia], 1, function(x) {all(is.na(x) | x == 0)})
  regla <- paste0("!(", variable, "==2) | All_NA_0==TRUE")
  AddRule(etiqueta, regla, "z==1 & B$v9i2==1")
}
rm(All_NA_0, etiqueta, i, j, regla, seq_i, secuencia, variable, z)
```

```{r}
#-----------------------------
# 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("val_9_0", 14+(i-9002)/8)
  variable <- paste0("B$v", i)
  anterior <- paste0("B$v", i-1)
  regla <- paste0(variable, " > 0")
  condicion <- paste0(anterior, " > 0")
  AddRule(etiqueta, regla, condicion)
}
rm(condicion,etiqueta,i,regla,seq_i,anterior,variable)
```

```{r}
#-----------------------------
# 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("val_9_0", 20+(i-8997)/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("AddValidSumas('", etiqueta, "', ", variable, ", ", sumando_1,
                    ", ", sumando_2, ", filtro='", filtrado, "')")
  eval(parse(text = comando), envir = .GlobalEnv)
}
rm(comando,etiqueta,filtrado,i,seq_i,sumando_1,sumando_2,variable)
```

```{r}
#-----------------------------
# REGLAS val_9_027...val_9_032
#-----------------------------
# Si existe valor en la columna 8 este debe ser >= que el valor de la columna 4.

seq_i <- seq(9010, 9050, 8)
for (i in seq_i) {
  etiqueta <- paste0("val_9_0", 26+(i-9002)/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, ")")
  AddRule(etiqueta, regla, filtrado)
}
rm(anterior,etiqueta,filtrado,i,seq_i,regla1,regla2,regla,siguiente,variable)
```

```{r}
#-----------------------------
# 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 valores en las columnas (5) y (6).

seq_i <- seq(9008, 9048, 8)
for (i in seq_i) {
  etiqueta <- paste0("val_9_0", 32+(i-9000)/8)
  variable <- paste0("B$v", i)
  anterior <- paste0("B$v", i-1)
  regla <- paste0(variable, " %in% 1:3 & ", anterior, " > 0")
  condicion <- paste0("B$v", i+1, " < ", "B$v", i-3)
```

```
  AddRule(etiqueta, regla, condicion)
}
rm(anterior,etiqueta,condicion,i,seq_i,regla,variable)
```

```{r}
#-----------------------------
# 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".

AddRule("val_9_039", "B$v9006 >= 0.05*B$v9005 & B$v9006 <= B$v9005", "B$v9005 > 0")
AddRule("val_9_040", "B$v9014 >= 0.05*B$v9013 & B$v9014 <= B$v9013", "B$v9013 > 0")
AddRule("val_9_041", "B$v9022 >= 0.05*B$v9021 & B$v9022 <= B$v9021", "B$v9021 > 0")
AddRule("val_9_042", "B$v9030 >= 0.05*B$v9029 & B$v9030 <= B$v9029", "B$v9029 > 0")
AddRule("val_9_043", "B$v9038 >= 0.05*B$v9037 & B$v9038 <= B$v9037", "B$v9037 > 0")
AddRule("val_9_044", "B$v9046 >= 0.05*B$v9045 & B$v9046 <= B$v9045", "B$v9045 > 0")
```

```{r}
#-----------------------------
# 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.

AddRule("val_9_045", "B$v9009 <= B$v9005", "B$v9005 > 0")
AddRule("val_9_046", "B$v9017 <= B$v9013", "B$v9013 > 0")
AddRule("val_9_047", "B$v9025 <= B$v9021", "B$v9021 > 0")
AddRule("val_9_048", "B$v9033 <= B$v9029", "B$v9029 > 0")
AddRule("val_9_049", "B$v9041 <= B$v9037", "B$v9037 > 0")
AddRule("val_9_050", "B$v9049 <= B$v9045", "B$v9045 > 0")
```

```{r}
#----------------
# REGLA val_9_051
#----------------
# Si V9010, V9018, V9026, V9034 > 0 entonces en la línea 327, v8093 > 0.

AddRule("val_9_051", "B$v8093 > 0", "B$v9010 > 0 | B$v9018 > 0 | B$v9026 > 0 | B$v9034 > 0")
```

```{r}
#-----------------------------
# REGLAS val_9_052...val_9_057
#-----------------------------
# Si se registra en la columna (6) como opción de respuesta 3="Otros Usos", de forma obligatoria en la columna Observación (9) debe existir información.

AddRule("val_9_052", "nchar(trimws(as.character(B$v9011))) > 2", "B$v9008==3")
AddRule("val_9_053", "nchar(trimws(as.character(B$v9019))) > 2", "B$v9016==3")
AddRule("val_9_054", "nchar(trimws(as.character(B$v9027))) > 2", "B$v9024==3")
AddRule("val_9_055", "nchar(trimws(as.character(B$v9035))) > 2", "B$v9032==3")
AddRule("val_9_056", "nchar(trimws(as.character(B$v9043))) > 2", "B$v9040==3")
AddRule("val_9_057", "nchar(trimws(as.character(B$v9051))) > 2", "B$v9048==3")
```

I. Validaciones lógicas: Capítulo 9, Sección II.

```{r}
#-----------------------------
# REGLAS val_9_058...val_9_068
#-----------------------------
# En la pregunta 1, FILA 339, solamente para el casillero (9100) puede darse la opción 5="Mantenimiento".

#for (i in c(seq(9060, 9096, 4), 9103)) {
for (i in seq(9060, 9096, 4)) {
```

```
  etiqueta <- ifelse(i < 9103, paste0("val_9_0", 57+(i-9056)/4), "val_9_068")
  regla <- paste0("B$v", i, " != 5")
  filtrado <- paste0("B$v", i-2, " > 0 | B$v", i-1, " > 0")
  AddRule(etiqueta, regla, filtrado)
}
rm(i, etiqueta, regla, filtrado)
```

```{r}
#-----------------------------
# REGLAS val_9_069...val_9_080
#-----------------------------
# En la Pregunta 1, la columna Uso Principal (4) es de llenado obligatorio, siempre y cuando exista información en las columnas Cantidad (2) y Valor (3); así como no debe permitirse ingresar información en la columna Uso Principal (4) si no existe información en las columnas Cantidad (2) y Valor (3).

for (i in c(seq(9060, 9100, 4), 9103)) {
  etiqueta <- ifelse(i < 9103, paste0("val_9_0", 68+(i-9056)/4), "val_9_080")
  regla1 <- paste0("B$v", i, " %in% c(1:4,6)")
  regla2 <- paste0("B$v", i-2, " > 0 & B$v", i-1, " > 0")
  regla <- !xor(eval(parse(text = regla1)), eval(parse(text = regla2)))
  filtrado <- "B$v9ii1 == 1"
  AddRule(etiqueta, regla, filtrado)
}
rm(i, etiqueta, regla1, regla2, regla, filtrado)
```

```{r}
#----------------
# 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)
# AddRule("val_9_081", "B$v9105 <= z", "B$v9ii1==1")
# rm(z)
```

```{r}
#-----------------------------
# REGLAS val_9_082...val_9_093
#-----------------------------
# Si se registra en la columna (4) como opción de respuesta 6="Otros Usos", de forma obligatoria en la columna Observación (5) debe existir información.

for (i in c(seq(9060, 9100, 4), 9103)) {
  etiqueta <- ifelse(i < 9103, paste0("val_9_0", 81+(i-9056)/4), "val_9_093")
  regla <- paste0("nchar(trimws(as.character(B$v", i + 1, "))) > 2")
  filtrado <- paste0("B$v", i, "==6")
  AddRule(etiqueta, regla, filtrado)
}
rm(i, etiqueta, regla, filtrado)
```

J. Validaciones lógicas: Capítulo 10, Sección I.

```{r}
#-------------------
# 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)"
# AddRule("val_10_1_001", "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.

AddRule("val_10_1_002", "B$v10001 < B$v1173", "B$v10001 > 0")

#------------------
# REGLA val_10_1_003
#------------------
# El Valor USD(3) dividido para la Cantidad(2) debe estar dentro del rango de 0.40 a 1.30.

# AddRule("val_10_1_003", "B$v10001 >= 0.4*B$v10000 & B$v10001 <= 1.3*B$v10000", "B$v10000 > 0")
# AddRule("val_10_1_003", "B$v10001 >= 0.4*B$v10000 & B$v10001 <= 1.3*B$v10000", "B$v10000 > 0")
AddRule("val_10_1_003", "B$v10001 >= 0.1*B$v10000 & B$v10001 <= 2.0*B$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).

AddRule("val_10_1_004", "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.

AddRule("val_10_1_005", "(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.

AddRule("val_10_1_006", "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).

AddRule("val_10_1_007", "B$v10006==1 | B$v10014==1 | B$v10022==1", "B$v10i3==1")
```

```{r}
#----------------------------------
# REGLAS val_10_1_008...val_10_1_010
#----------------------------------
# En el caso de que en la columna (1) conteste "SI", en la columna (2) puede contestar cualquiera de las dos opciones.

AddRule("val_10_1_008", "B$v10007 %in% 1:2", "B$v10006==1")
AddRule("val_10_1_009", "B$v10015 %in% 1:2", "B$v10014==1")
AddRule("val_10_1_010", "B$v10023 %in% 1:2", "B$v10022==1")
```

```{r}
#----------------------------------
# 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).

AddRule("val_10_1_011", "B$v10009 > 0 & B$v10010 > 0 & B$v10011 > 0", "B$v10008==1")
```

```
AddRule("val_10_1_012", "B$v10017 > 0 & B$v10018 > 0 & B$v10019 > 0", "B$v10016==1")
AddRule("val_10_1_013", "B$v10025 > 0 & B$v10026 > 0 & B$v10027 > 0", "B$v10024==1")
```

```{r, warning=FALSE}
#-----------------------------------
# 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$v10022), 1, min, na.rm = T)
for (i in seq(10006, 10022, 8)) {
  etiqueta <- paste0("val_10_1_0", 13+(i-9998)/8)
  variable <- paste0("B$v", i)
  secuencia <- sapply(1:7, 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")
  AddRule(etiqueta, regla, "z!=1 & B$v10i3==1")
}
rm(All_NA_0, etiqueta, i, j, regla, secuencia, variable, z)
```

```{r, warning=FALSE}
#-----------------------------------
# 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$v10024), 1, min, na.rm = T)
for (i in seq(10008, 10024, 8)) {
  etiqueta <- paste0("val_10_1_0", 16+(i-10000)/8)
  variable <- paste0("B$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")
  AddRule(etiqueta, regla, "z!=1 & (B$v10006 == 1 | B$v10014 == 1 | B$v10022 == 1)")
}
rm(All_NA_0, etiqueta, i, j, regla, secuencia, variable, z)
```

```{r}
#-----------------------------------
# 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).

 AddRule("val_10_1_020", "(100*B$v10012) %% 100 >= 0", "B$v10008==1 & B$v10012 > 0")
 AddRule("val_10_1_021", "(100*B$v10020) %% 100 >= 0", "B$v10016==1 & B$v10020 > 0")
 AddRule("val_10_1_022", "(100*B$v10028) %% 100 >= 0", "B$v10024==1 & B$v10028 > 0")
```

```{r}
#-----------------------------------
# REGLAS val_10_1_023...val_10_1_025
#-----------------------------------
# Las variables 10010, 10018, 10026 deben ser menores o iguales que 24.

AddRule("val_10_1_023", "B$v10010 %in% 1:24", "B$v10008==1")
AddRule("val_10_1_024", "B$v10018 %in% 1:24", "B$v10016==1")
AddRule("val_10_1_025", "B$v10026 %in% 1:24", "B$v10024==1")
```

```{r}
#-----------------------------------
# REGLAS val_10_1_026...val_10_1_028
#-----------------------------------
```

```
# Las variables 10011, 10019, 10027 deben ser menores o iguales que 31.

AddRule("val_10_1_026", "B$v10011 %in% 1:31", "B$v10008==1")
AddRule("val_10_1_027", "B$v10019 %in% 1:31", "B$v10016==1")
AddRule("val_10_1_028", "B$v10027 %in% 1:31", "B$v10024==1")
```

```{r}
#-----------------------------------
# REGLAS val_10_1_029...val_10_1_031
#-----------------------------------
# La información registrada en las variables 10009, 10017 y 10025 debe existir respuesta de formato numerico de escala (F10.2).

AddRule("val_10_1_029", "(100*B$v10009) %% 100 >= 0", "B$v10008==1 & B$v10009 > 0")
AddRule("val_10_1_030", "(100*B$v10017) %% 100 >= 0", "B$v10016==1 & B$v10017 > 0")
AddRule("val_10_1_031", "(100*B$v10025) %% 100 >= 0", "B$v10024==1 & B$v10025 > 0")
```

```{r}
#------------------
# REGLA val_10_1_032
#------------------
# En el casillero v10012 se despliega el cálculo de la fórmula: v10012 = 12 * v10009 * v10010 * v10011.

G$DOCE <- 12
z <- apply(cbind(G$DOCE, B$v10009, B$v10010, B$v10011), 1, prod, na.rm = F)
z <- round(z, 2)
AddRule("val_10_1_032", "round(B$v10012, 2) == z", "B$v10008==1")
rm(z)
```

```{r}
#------------------
# REGLA val_10_1_033
#------------------
# En el casillero v10020 se despliega el cálculo de la fórmula: v10020 = 12 * v10017 * v10018 * v10019.

z <- apply(cbind(G$DOCE, B$v10017, B$v10018, B$v10019), 1, prod, na.rm = F)
z <- round(z, 2)
AddRule("val_10_1_033", "round(B$v10020, 2) == z", "B$v10016==1")
rm(z)
```

```{r}
#------------------
# REGLA val_10_1_034
#------------------
# En el casillero v10020 se despliega el cálculo de la fórmula: v10028 = 12 * v10025 * v10026 * v10027.

z <- apply(cbind(G$DOCE, B$v10025, B$v10026, B$v10027), 1, prod, na.rm = F)
z <- round(z, 2)
AddRule("val_10_1_034", "round(B$v10028, 2) == z", "B$v10024==1")
rm(z)
```

```{r}
#------------------
# 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)
AddRule("val_10_1_035", "z <= B$v1173")
rm(z)
```

K. Validaciones lógicas: Capítulo 10, Sección II.

```{r}
#------------------
# REGLA val_10_2_001
#------------------
# Obligatorio. En el caso de contestar "Sí", ir a la pregunta 1.1; en el caso de contestar "No", pasar a la pregunta 2.

AddRule("val_10_2_001", "B$v10ii1 %in% 1:2")

#------------------
# REGLA val_10_2_002
#------------------
# Obligatorio si respondió "Sí" en la Pregunta 1. Esta variable debe admitir como valor mínimo 0.01. Formato número (F10.2).

AddRule("val_10_2_002", "B$v10ii11 >= 0.01", "B$v10ii1==1")

#------------------
# REGLA val_10_2_003
#------------------
# Obligatorio. Debe existir una sola respuesta "Sí" ó "No". Si el informante contesta "No", pasar directamente a III. Otros Residuos y/o desechos.

AddRule("val_10_2_003", "B$v10ii2 %in% 1:2")

#------------------
# REGLA val_10_2_004
#------------------
# Obligatorio. Debe existir una sola respuesta "Sí" ó "No". Si el informante contesta "No", pasar directamente a la pregunta 5.

AddRule("val_10_2_004", "B$v10ii3 %in% 1:2", "B$v10ii2==1")
```

```{r}
#------------------
# REGLA val_10_2_005
#------------------
# Debe existir respuesta de formato numérico de escala (F10.2) en las columnas (1) y (4).

oper <- function(x) {((100*x) %% 100) >= 0}
regla <- "oper(B$v10032) & oper(B$v10035)"
AddRule("val_10_2_005", regla, "B$v10ii3==1")
rm(oper, regla)
```

```{r}
#------------------
# REGLA val_10_2_006
#------------------
# En el casillero v10035 se despliega el cálculo de la fórmula: v10035 = 12 * v10032 * v10033 * v10034.

z <- apply(cbind(G$DOCE, B$v10032, B$v10033, B$v10034), 1, prod, na.rm = F)
z <- round(z, 2)
AddRule("val_10_2_006", "round(B$v10035, 2) == z", "B$v10032 > 0")
rm(z)
G$DOCE <- NULL
```

```{r}
#------------------
# REGLA val_10_2_007
#------------------
# El valor de la v10035 debe ser inferior o igual que la suma de las variables del Cap. 10, Sección I (Agua): v10000 (Preg. 1) + v10004 (Preg. 3.1) + v10030 (Preg. 3).

Agua_Tank <- ifelse(B$v10003 == 2, B$v10004, ifelse(B$v10003 == 1, B$v10004/264.17205, NA))
Agua_consumo <- apply(cbind(B$v10000, Agua_Tank, B$v10030), 1, sum, na.rm = T)
AddRule("val_10_2_007", "B$v10035 <= Agua_consumo", "B$v10ii3==1")
```

```
rm(Agua_consumo, Agua_Tank)
```

```{r}
#------------------
# REGLA val_10_2_008
#------------------
# Si en la pregunta 3 contesto "SI", obligatoriamente debe existir información en las columnas (1), (2) y (3).

AddRule("val_10_2_008", "B$v10032 > 0 & B$v10033 > 0 & B$v10034 > 0", "B$v10ii3==1")
```

```{r}
#------------------
# REGLA val_10_2_009
#------------------
# Las variables: v10033 <= 24 y v$10034 <= 31.

AddRule("val_10_2_009", "B$v10033 %in% 1:24 & B$v10034 %in% 1:31", "B$v10ii3==1")
```

```{r}
#----------------------------------
# REGLAS val_10_2_010...val_10_2_014
#----------------------------------
# Obligatorio. En el caso de contestar "Sí", continuar con la Pregunta 5.1; en el caso de contestar "No", pasar a la Sección Otros Residuos y/o Desechos.

z <- apply(cbind(B$v10ii5111, B$v10ii5112, B$v10ii5113, B$v10ii5114), 1, min)
AddRule("val_10_2_010", "is.na(z) & B$v10ii5==2", "B$v10ii2==1 & B$v10ii5==2")
AddRule("val_10_2_011", "z == 1", "B$v10ii5==1")
rm(z)
```

```{r}
#------------------
# REGLA val_10_2_015
#------------------
# Si el informante responde "SI" en al menos una de las variables 1.1, 1.2, 1.3, 1.4, la variable 2"Ninguno" no deberá estar marcada.

z <- apply(cbind(B$v10ii5111, B$v10ii5112, B$v10ii5113, B$v10ii5114), 1, min)
AddRule("val_10_2_015", "B$v10ii5==1", "z == 1")
rm(z)
```

```{r}
#------------------
# REGLA val_10_2_016
#------------------
# Si está marcada la variable 2"Ninguno" con "SI", entonces las variables 1.1,1.2,1.3,1.4, no deberán estar marcadas.

z <- apply(cbind(B$v10ii5111, B$v10ii5112, B$v10ii5113, B$v10ii5114), 1, min)
AddRule("val_10_2_016", "is.na(z) & B$v10ii5==2", "B$v10ii2==1 & B$v10ii5==2")
rm(z)
```

```{r}
#------------------
# REGLA val_10_2_017
#------------------
# Si en el capítulo 10, sección II,  pregunta 5, presenta algún tipo de tratamiento a sus aguas residuales, entonces la Línea 326 del Capítulo 8, celdas 8089 y 8091, deberán tener valores mayores que 0.

AddRule("val_10_2_017", "B$v8089 > 0 | B$v8091 > 0", "B$v10ii5==1")
```

```{r}
#----------------------
# REGLA val_10_2_018
#----------------------
# Si en la Pregunta 5 se contestó a 2.- Ninguno = "SI", el valor de la Pregunta 6 debe ser 0%.

# AddRule("val_10_2_018", "B$v10ii6==0", "B$v10ii5==2")
```

```{r}
#----------------------
# REGLA val_10_2_019
#----------------------
# En la Pregunta 6 debe existir una respuesta numérica entre 0% y 100%.

AddRule("val_10_2_019", "B$v10ii6 %in% 0:100", "B$v10ii2==1 & B$v10ii5==1")
```

```{r}
#----------------------
# REGLA val_10_2_020
#----------------------
# Si la Pregunta 6 vale 0%, pasar a la Sección de "Otros Residuos y/o Desechos".

AddRule("val_10_2_020", "B$v10ii5==2", "B$v10ii6==0")
```

L. Validaciones lógicas: Capítulo 10, Sección III (Residuos y Desechos).

```{r}
# Bloque de código que crea una lista con componentes que definen objetos que sirven para las iteraciones de las reglas de validación de residuos / desechos. También se crean algunas variables globales auxiliares útiles en la ejecución de las iteraciones. Los objetos y/o variables creados 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).

lista_R <- list() # Esta lista vacía incluirá en una componente la secuencia de variables a
# validar, en otra componente los prefijos y en otra los sufijos de las variables de control
# que se deben crear en el código de cada regla de validación.

lista_R$seq <- c("seq(10062, 10230, 21)", "seq(10251, 10272, 21)", "10314", "seq(10377, 10461, 21)", "seq(10524, 10986, 21)")
lista_R$sufix <- c("(i-10041)/21", "(i-10041)/21", "12", "(i-10104)/21", "(i-10146)/21")
lista_R$prefix <- c("val_10_3_00", "val_10_3_0", "val_10_3_0", "val_10_3_0", "val_10_3_0")
grupo_k <- 1:5 # Esta variable segmenta a las variables de la columna (1) de residuos / desechos.
seq_cad <- sapply(1:5, function(x) lista_R$seq[[x]]) # Cadenas que contienen las secuencias de
                                        # las variables de la columna (1).
seq_var <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv)))
# Este vector de cadenas contiene los sufijos numéricos de las variables de la columna (1)
# de residuos / desechos.
seq_var <- paste0("v", seq_var) # Vector de cadenas con las variables de la columna (1)
                        # de residuos / desechos.
```

```{r}
#-----------------------------------
# REGLAS val_10_3_001...val_10_3_040
#-----------------------------------
# La información de la columna (1) es obligatoria (responder "Sí" ó "No").

for (k in grupo_k) { # grupo_k es una variable global definida en el bloque de definiciones
                # anterior.
  comando <- paste0("seq_k <- ", lista_R$seq[[k]])
  # "comando" es una cadena que construye el comando de la subsecuencia de sufijos numéricos
  # de las variables de la columna pivote (col. 1) que se validarán con valores 1 ó 2.
  eval(parse(text = comando), envir = .GlobalEnv)
  # Ejecución de la expresión en lenguaje R contenida en la cadena "comando".
  for (i in seq_k) {
    comando <- paste0("etiqueta <- paste0('", lista_R$prefix[[k]], "', ", lista_R$sufix[[k]], ")")
    # "comando" es una cadena que construye la etiqueta de la variable de control correspondiente
```

```
    # al k-ésimo prefijo y k-ésimo sufijo de la variable de control a generar.
    eval(parse(text = comando), envir = .GlobalEnv)
    # Ejecución de la expresión en lenguaje R contenida en la cadena "comando".
    variable <- paste0("B$v", i) # Cadena con la etiqueta de la Variable de la columna (1)
                          # que se validará en la presente iteración.
    regla <- paste0(variable, " %in% 1:2") # Cadena con la regla de validación concreta.
    AddRule(etiqueta, regla) # Agregación de todas y c/u de las reglas que tengan al menos
                          # un caso a validar. No se usa filtro porque cada regla de
                          # validación se aplica a todas las empresas sin excepción.
  }
}
rm(comando, i, k, etiqueta, seq_k, variable, regla) # Eliminación de las variables creadas en el
                                          # bloque de código actual.
```

```{r}
#-----------------------------------
# REGLAS val_10_3_041...val_10_3_080
#-----------------------------------
# Si responde "No" en la columna (1) bloquear la linea 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.

for (k in grupo_k) { # grupo_k es una variable global definida en el bloque de definiciones
                  # anterior.
  comando <- paste0("seq_k <- ", lista_R$seq[[k]])
  # "comando" es una cadena que construye el comando de la subsecuencia de sufijos numéricos de las variables de la columna
pivote (col. 1) que se validarán con valores 1 ó 2.
  eval(parse(text = comando), envir = .GlobalEnv)
  # Ejecución de la expresión en lenguaje R contenida en la cadena "comando".

  for (i in seq_k) {
    comando <- paste0("etiqueta <- paste0('", lista_R$prefix[[2]], "', ", lista_R$sufix[[k]], " + 40)")
    # "comando" es una cadena que construye la etiqueta de la variable de control correspondiente
    # al k-ésimo prefijo y k-ésimo sufijo de la variable de control a generar.

    eval(parse(text = comando), envir = .GlobalEnv)
    # Ejecución de la expresión en lenguaje R contenida en la cadena "comando".

    variable <- paste0("B$v", i) # Cadena con la etiqueta de la Variable de la columna (1)
                          # que se validará en la presente iteración.

    sucesivas <- sapply(c(2,3,5,7,9,11,17,18,19), function(j) paste0("v", i+j))

    All_NA_0 <- apply(B[, sucesivas], 1, function(x) {all(is.na(x) | x == 0)})

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

    regla <- paste0("(!(", regla1, ") | All_NA_0 == TRUE)")

    # regla <- paste0(variable, " == 2 & All_NA_0==TRUE") # Cadena con la regla de validación concreta.
    AddRule(etiqueta, regla, filtro) # Agregación de todas y c/u de las reglas que tengan al menos
                          # un caso a validar. No se usa filtro porque cada regla de
                          # validación se aplica a todas las empresas sin excepción.
  }
}
rm(All_NA_0, activ_econ, comando, i, k, etiqueta, filtro, seq_k, variable, regla, regla1, sucesivas) # Eliminación de las variables
creadas en el bloque de código actual.
```

```{r, warning=FALSE}
```

```
#-------------------------------------
# REGLAS val_10_3_081...val_10_3_120
#-------------------------------------
# Si la columna 2.2 (Cantidad) tiene valores > 0, entonces sum(col 3.1 + col 3.2 + col 3.3 + col 4.1) = col 2.2.

seq_vnum <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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).

library(foreach) # Carga de la librería que ofrece la posibilidad de construir objetos iterables.

z_suma <- foreach(i=1:length(seq_vnum)) %do% apply(B[, c(seq_v3_1[i], seq_v3_2[i], seq_v3_3[i], seq_v4_1[i])], 1, sum, na.rm = T)
# 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 3.1, 3.2, 3.3 y 4.1 para el residuo de la componente que corresponda en la lista "z_suma".

detach("package:foreach", unload = TRUE) # Descarga de memoria de la librería "foreach".

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de la ENESEM).
  t_2_2 <- data.frame(z_2_2[, i]) # Vector que contiene los valores de la variable de la columna
                                  # pivote (col 2.2) para el residuo "i".
  t_suma <- data.frame(z_suma[[i]]) # Vector que contiene los valores de la suma de las variables
                                    # de las columnas 3.1, 3.2, 3.3 y 4.1 para el residuo "i".
  t_resta <- t_2_2 - t_suma # Vector que contiene la resta de los vectores "t_2_2" y "t_suma".
                            # Este vector sirve para saber si, finalmente, "t_2_2==t_suma",
                            # siempre que todas las componentes de "t_resta" sean cero o "missing".
  All_NA_0 <- apply(t_resta, 1, function(x) {all(is.na(x) | x == 0)})
  # All_NA_0 es el vector lógico que marca con TRUE a las componentes de t_resta son nulas o "missing".

  AddRule(paste0("val_10_3_0", 80+i), "All_NA_0==TRUE", paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(All_NA_0, i, seq_vnum, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1, z_2_2, z_condic, z_suma)
rm(t_2_2, t_suma, t_resta)
```

```{r}
#-------------------------------------
# REGLAS val_10_3_121...val_10_3_132
#-------------------------------------
# En la columna (2.1) de la Tabla 1, debe existir las opciones de unidades de medida siguientes: 1 Kilogramo; 2 Tonelada.

seq_vnum <- unlist(sapply(seq_cad[1:3], function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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 a 12, que es el recuento de tipos de residuos de la primera tabla de residuos del formulario (Tabla III.1 RESIDUOS NO PELIGROSOS).

seq_v2_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables pivote (col 2.1).
seq_v2_2 <- paste0("v", seq_vnum + 1) # Vector cadena con la secuencia de variables pivote (col 2.1).

z_regla <- data.frame(apply(B[, seq_v2_1], 2, function(x) x %in% 1:2))
```

```
# "z_regla" es el vector lógico que marca con TRUE a las variables de la columna 2.1 de la Tabla 1 que tienen únicamente valores con código 1 (kilogramo) o 2 (tonelada).

z_condic <- data.frame(apply(B[, seq_v2_2], 2, function(x) x > 0))
# "z_condic" es el vector lógico que marca con TRUE a las variables de la columna 2.2 de la Tabla 1 que tienen valores positivos. Esto es, se conoce la cantidad de residuo generado.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 12 (número de residuos de la Tabla III.1).
  AddRule(paste0("val_10_3_", 120+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v2_1, seq_v2_2, z_regla, z_condic)
```

```{r}
#-----------------------------------
# REGLAS val_10_3_133...val_10_3_138
#-----------------------------------
# En la columna (2.1) de la Tabla 2, debe existir las opciones de unidades de medida siguientes: 1 "kilogramo", 2 "tonelada", en casi todas las líneas, excepto en la línea 377, casillero 10400, donde se permitirá registrar: 1 Kilogramo; 2 Tonelada; 3 Galón.

seq_vnum <- unlist(sapply(c(seq_cad[4], "10524"), function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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 a 6, que es el recuento de tipos de residuos de la segunda tabla de desechos del formulario (Tabla III.2 DESECHOS ESPECIALES).

seq_v2_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables pivote (col 2.1).
seq_v2_2 <- paste0("v", seq_vnum + 1) # Vector cadena con la secuencia de variables pivote (col 2.1).

z_regla <- data.frame(apply(B[, setdiff(seq_v2_1, "v10400")], 2, function(x) x %in% 1:2))
# "z_regla" es el vector lógico que marca con TRUE a las variables de la columna 2.1 de la Tabla 2 que tienen únicamente valores con código 1 (kilogramo) o 2 (tonelada), salvo la variable "v10400"
# (Línea 377), la cual puede admitir valores con códigos 1 (kg), 2 (tonelada) o 3 (US gal).
z_v10400 <- data.frame(apply(B[, c("v10400", "v10379")], 2, function(x) x %in% 1:3))
z_v10400 <- data.frame(z_v10400[, 1])
names(z_v10400) <- "v10400"

z_regla <- cbind(z_regla[, 1], z_v10400, z_regla[, 2:5])

z_condic <- data.frame(apply(B[, seq_v2_2], 2, function(x) x > 0))
# "z_condic" es el vector lógico que marca con TRUE a las variables de la columna 2.2 de la Tabla 2 que tienen valores positivos. Esto es, se conoce la cantidad de residuo generado.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 6 (número de residuos de la Tabla III.2).
  AddRule(paste0("val_10_3_", 132+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v2_1, seq_v2_2, z_regla, z_condic, z_v10400)
```

```{r}
#-----------------------------------
# REGLAS val_10_3_139...val_10_3_160
#-----------------------------------
# En la columna (2.1) de la Tabla 3, debe existir las opciones de unidades de medida siguientes: 1 Kilogramo; 2 Tonelada; 3 Galón.

seq_vnum <- unlist(sapply("seq(10545, 10986, 21)", function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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 a 22, que es el recuento de tipos de residuos de la tercera tabla de desechos del formulario (Tabla III.3 DESECHOS PELIGROSOS).

seq_v2_1 <- paste0("v", seq_vnum) # Vector cadena con la secuencia de variables pivote (col 2.1).
```

```
seq_v2_2 <- paste0("v", seq_vnum + 1) # Vector cadena con la secuencia de variables pivote (col 2.1).

z_regla <- data.frame(apply(B[, seq_v2_1], 2, function(x) x %in% 1:3))
# "z_regla" es el vector lógico que marca con TRUE a las variables de la columna 2.1 de la Tabla 3
# que tienen únicamente valores con código 1 (kilogramo), 2 (tonelada) o  3 (US Galon).

z_condic <- data.frame(apply(B[, seq_v2_2], 2, function(x) x > 0))
# "z_condic" es el vector lógico que marca con TRUE a las variables de la columna 2.2 de la Tabla 3
# que tienen valores positivos. Esto es, se conoce la cantidad de residuo generado.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 22 (número de residuos de la Tabla III.3).
  AddRule(paste0("val_10_3_", 138+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o
  # fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v2_1, seq_v2_2, z_regla, z_condic)
```

```{r, warning=FALSE}
#-----------------------------------
# REGLAS val_10_3_161...val_10_3_200
#-----------------------------------
# Si en la columna (1) tiene como respuesta "Sí", en las columnas 3.1, 3.2, 3.3 y 4.1 debe ingresarse el valor 0 si no se conoce ni la unidad (columna 2.1) ni la cantidad del residuo (columna 2.2) y si se ha realizado efectivamente algún tipo de gestión.

seq_vnum <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv)))
# 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_condic 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_condic contiene en sus columnas los filtros de igualdad a 1 para las variables de la columna pivote (col 1).
z_2_2 <- data.frame(apply(B[, seq_v2_2], 2, is.na))
# El frame z_condic contiene en sus columnas los filtros de igualdad a 1 para las variables de la columna pivote (col 1).
z_condic <- 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).

library(foreach) # Carga de la librería que ofrece la posibilidad de construir objetos iterables.

z_suma <- foreach(i=1:length(seq_vnum)) %do% apply(B[, c(seq_v3_1[i], seq_v3_2[i], seq_v3_3[i], seq_v4_1[i])], 1, sum)
z_suma <- data.frame(z_suma)
# 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 3.1, 3.2, 3.3 y 4.1 para el residuo de la componente que corresponda en la lista "z_suma".

detach("package:foreach", unload = TRUE) # Descarga de memoria de la librería "foreach".

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  AddRule(paste0("val_10_3_", 160+i), paste0("z_suma[", i, "]==0"), paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v1, seq_v2_1, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1, z_1, z_2_1, z_2_2, z_condic, z_suma)
```

```
```

```{r, warning=FALSE}
#-------------------------------------
# REGLAS val_10_3_201...val_10_3_240
#-------------------------------------
# Si se conoce la unidad y la cantidad del residuo, se colocará la cantidad mayor que cero correspondiente a la gestión interna o externa del residuo (columnas 3.1, 3.2, 3.3 y 4.1).

# 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 <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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_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.

z_2_1 <- data.frame(apply(B[, seq_v2_1], 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 2.1).
z_2_2 <- 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 2.2.
z_condic <- data.frame(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_3_1 <- data.frame(apply(B[, seq_v3_1], 2, function(x) is.na(x) | x > 0))
# El frame z_3_1 contiene en sus columnas los filtros de no nulidad para las variables de la columna 3.1.
z_3_2 <- data.frame(apply(B[, seq_v3_2], 2, function(x) is.na(x) | x > 0))
# El frame z_3_2 contiene en sus columnas los filtros de no nulidad para las variables de la columna 3.2.
z_3_3 <- data.frame(apply(B[, seq_v3_3], 2, function(x) is.na(x) | x > 0))
# El frame z_3_3 contiene en sus columnas los filtros de no nulidad para las variables de la columna 3.3.
z_4_1 <- data.frame(apply(B[, seq_v4_1], 2, function(x) is.na(x) | x > 0))
# El frame z_4_1 contiene en sus columnas los filtros de no nulidad para las variables de la columna 4.1.

library(foreach) # Carga de la librería que ofrece la posibilidad de construir objetos iterables.

z_suma <- foreach(i=1:length(seq_vnum)) %do% apply(B[, c(seq_v3_1[i], seq_v3_2[i], seq_v3_3[i], seq_v4_1[i])], 1, sum, na.rm=T)
# 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 3.1, 3.2, 3.3 y 4.1 para el residuo de la componente que corresponda en la lista "z_suma".

detach("package:foreach", unload = TRUE) # Descarga de memoria de la librería "foreach".

z_regla <- data.frame(z_3_1 & z_3_2 & z_3_3 & z_4_1 & (data.frame(z_suma) > 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.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  AddRule(paste0("val_10_3_", 200+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_condic[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v2_1, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1, z_2_1, z_2_2, z_condic)
rm(z_3_1, z_3_2, z_3_3, z_4_1, z_suma, z_regla)
```

```{r, warning=FALSE}
#-------------------------------------
# REGLAS val_10_3_241...val_10_3_280
#-------------------------------------
```

```
# En las columnas (8.1), (8.2) y (8.3) debe existir un porcentaje válido. La suma de estas 3 columnas (8.1), (8.2) y (8.3) debe ser igual al 100%, caso contrario mostrar mensaje de error.

# 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 <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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.

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.

library(foreach) # Carga de la librería que ofrece la posibilidad de construir objetos iterables.

z_suma <- foreach(i=1:length(seq_vnum)) %do% apply(B[, c(seq_v8_1[i], seq_v8_2[i], seq_v8_3[i])], 1, sum, na.rm=T)
# 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".

detach("package:foreach", unload = TRUE) # Descarga de memoria de la librería "foreach".

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.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  AddRule(paste0("val_10_3_", 240+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_4_1[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, 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(z_suma, z_regla)
```

```{r}
#-------------------------------------
# REGLAS val_10_3_281...val_10_3_320
#-------------------------------------
# Si en la columna (4.1) no existe información (missing), debe bloquearse las columnas (8.1), (8.2), (8.3) y (9).

seq_vnum <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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) 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_9)
# 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.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  regla <- paste0("!(z_4_1[", i, "]==TRUE) | z_regla[", i, "]==TRUE")
  # AddRule(paste0("val_10_3_", 280+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_4_1[", i, "]==TRUE"))
  AddRule(paste0("val_10_3_", 280+i), regla)
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos /
desechos.
}

rm(i, 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)
rm(z_9, regla, z_regla)
```

```{r}
#-------------------------------------
# REGLAS val_10_3_321...val_10_3_360
#-------------------------------------
# Si existen valores mayores que cero en la columna (8.2), entonces debe existir valores mayores que cero en las columnas
(2.1), (2.2) y (4.1).

seq_vnum <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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_1 <- paste0("v", seq_vnum - 16) # Vector cadena con la secuencia de variables columna 2.1.
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.

# 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_2 contiene en sus columnas los filtros de positividad para las variables de la columna 2.2.
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 para las variables de la columna 4.1.
z_8_2 <- data.frame(apply(B[, seq_v8_2], 2, function(x) as.numeric(x) > 0))
# El frame z_8_2 contiene en sus columnas los filtros de positividad para las variables de la columna 8.2.

# z_regla <- data.frame(z_2_1 & z_2_2 & z_4_1)
z_regla <- data.frame(z_2_2 & z_4_1)
# 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.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  AddRule(paste0("val_10_3_", 320+i), paste0("z_regla[", i, "]==TRUE"), paste0("z_8_2[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos /
desechos.
}

rm(i, 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, z_regla)
```

```{r}
#-------------------------------------
# REGLAS val_10_3_361...val_10_3_400
#-------------------------------------
# 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 a 0%.

seq_vnum <- unlist(sapply(seq_cad, function(x) eval(parse(text = x), envir = .GlobalEnv))) + 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) x != ""))
# El frame z_9 contiene en sus columnas los filtros de no vaciedad para las variables de la columna 9.

for (i in 1:length(seq_vnum)) { # length(seq_vnum) vale 40 (número de residuos de las Tablas III.1,2,3).
  AddRule(paste0("val_10_3_", 360+i), paste0("z_9[", i, "]==TRUE"), paste0("z_8_3[", i, "]==TRUE"))
  # Llamada a la función de agregación de variables de control de validación para cada residuo o fila de las tablas de residuos / desechos.
}

rm(i, seq_vnum, seq_v8_3, seq_v9, z_8_3, z_9)
```

```{r}
#-------------------------------------
# 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.

AddRule("val_10_3_401", "B$v10076_1 > 0", "B$v10iii1_1 == 1")
AddRule("val_10_3_402", "B$v10076_2 > 0", "B$v10iii2_1 == 1")
AddRule("val_10_3_403", "B$v10076_3 > 0", "B$v10iii3_1 == 1")
```

```{r}
#-------------------------------------
# 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.

AddRule("val_10_3_404", "B$v10076_1 == 0 | is.na(B$v10076_1)", "B$v10iii1_1 == 2")
AddRule("val_10_3_405", "B$v10076_2 == 0 | is.na(B$v10076_2)", "B$v10iii2_1 == 2")
AddRule("val_10_3_406", "B$v10076_3 == 0 | is.na(B$v10076_3)", "B$v10iii3_1 == 2")
```

```{r}
#-------------------------------------
# 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.

AddRule("val_10_3_407", "B$v10078_1 > 0", "B$v10iii1_2 == 1")
AddRule("val_10_3_408", "B$v10078_2 > 0", "B$v10iii2_2 == 1")
AddRule("val_10_3_409", "B$v10078_3 > 0", "B$v10iii3_2 == 1")
```

```{r}
#-------------------------------------
# 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.

AddRule("val_10_3_410", "B$v10078_1 == 0 | is.na(B$v10078_1)", "B$v10iii1_2 == 2")
AddRule("val_10_3_411", "B$v10078_2 == 0 | is.na(B$v10078_2)", "B$v10iii2_2 == 2")
AddRule("val_10_3_412", "B$v10078_3 == 0 | is.na(B$v10078_3)", "B$v10iii3_2 == 2")
```

```{r}
#--------------------
# 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.

AddRule("val_10_3_413", "B$v8091 > 0", "B$v10076_1 > 0 | B$v10076_2 > 0 | B$v10076_3 > 0")
```

```{r}
#--------------------
# 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$v10076_3), 1, sum, na.rm = T)
AddRule("val_10_3_414", "z <= B$v8091", "B$v8091 > 0")
rm(z)
```

```{r}
#--------------------
# 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.

AddRule("val_10_3_415", "!(B$v10iii4 == 2) | (is.na(B$v10iii41) & is.na(B$v10iii42))")
```

```{r}
#--------------------
# 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.

AddRule("val_10_3_416", "!(B$v10iii41 == 2) | is.na(B$v10iii42)", "B$v10iii4 == 1")
```

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

AddRule("val_10_3_417", "B$v10iii42 > 0", "B$v10iii41 == 1")
```

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

AddRule("val_10_3_418", "B$v10iii42 < B$v3181", "B$v10iii41 == 1")
```

```{r}
#--------------------
# ELIMINACIÓN DEL WORKSPACE DE LAS VARIABLES GLOBALES TEMPORALES.
#--------------------
# El valor de la variable de la Pregunta 4.2 debe ser menor que la v3181.

rm(activ_econ, All_NA_0, comando, etiqueta, filtro, k, lista_R, grupo_k, regla, regla1, seq_cad, seq_k, sucesivas, variable)
rm(seq_v1, seq_v2_1, seq_v2_2, seq_v3_1, seq_v3_2, seq_v3_3, seq_v4_1, seq_v8_1, seq_v8_2, seq_var)
rm(z_1, z_2_1, z_2_2, z_3_1, z_3_2, z_condic)
```

```{r}
# Cálculo del total de errores por empresa.
# G$DOCE <- NULL
vars_err_suma <- names(G)[!(names(G) %in% lista_vars_identif)]
Err_x_Col <- apply(G[, vars_err_suma], 2, sum, na.rm = T) # Errores por variable de validación.
G$SumERR <- apply(G[, vars_err_suma], 1, sum, na.rm = T) # Errores por empresa
```

```{r}
# Transformación a Excel del dataframe G que contiene las validaciones.

writexl::write_xlsx(G, "C:/Users/rbenavides/Documents/EMPRESAS 2022/9no Corte - 11-12-2023/VAL
ESTANDAR/VAL_EST_MA_11_12_2023.xlsx")
```

#--- FIN DE LA SINTAXIS DE VALIDACIÓN LÓGICA DEL MÓDULO AMBIENTAL DE LA ENESEM 2022 ---

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

*** SINTAXIS SPSS DE CONTROL DE INTEGRIDAD DE LA BDD ENESEM 2022 (Excepto Residuos y/o desechos).**
*** Fecha de creación: 08/01/2024.**
*** Fecha de última edición: 26/01/2024.**

*** A. Calcular filtro para empresas efectivas. Determinación de esas empresas efectivas.**

```
FREQUENCIES efectividad absorbida v8086.
COMPUTE Efectivas_AMB = (efectividad = 1 & Missing(absorbida) & v8086 > 0).
FREQUENCIES Efectivas_AMB.

FILTER BY Efectivas_AMB.

**********
*** CAPÍTULO 8.
**********

* 1. Verificar que se cumpla: v8001= v4146 + v4147 + v4148 + v4196 + v4197.
COMPUTE #SUMA_v8001 = sum.1(v4146, v4147, v4148, v4196, v4197).
COMPUTE Dif_v8001 = #SUMA_v8001 - v8001.
FREQUENCIES Dif_v8001.
DELETE VARIABLES Dif_v8001.

* 2. Verificar que v8086 sea 1 y que v8087 sea mayor que cero. Al mismo tiempo, verificar que v8086 sea 2 y que v8087 sea
missing.
COMPUTE y8086 = (v8086 = 1 & v8087 > 0) | (v8086 = 2 & Missing(v8087)).
FREQUENCIES y8086.
DELETE VARIABLES y8086.

* 3. Verificar que v8088 sea 1 y que v8089 sea mayor que cero. Al mismo tiempo, verificar que v8088 sea 2 y que v8089 sea
missing.
COMPUTE y8088 = (v8088 = 1 & v8089 > 0) | (v8088 = 2 & Missing(v8089)).
FREQUENCIES y8088.
DELETE VARIABLES y8088.

* 4. Verificar que v8090 sea 1 y que v8091 sea mayor que cero. Al mismo tiempo, verificar que v8090 sea 2 y que v8091 sea
missing.
COMPUTE y8090 = (v8090 = 1 & v8091 > 0) | (v8090 = 2 & Missing(v8091)).
FREQUENCIES y8090.
DELETE VARIABLES y8090.

* 5. Verificar que v8092 sea 1 y que v8093 sea mayor que cero. Al mismo tiempo, verificar que v8092 sea 2 y que v8093 sea
missing.
COMPUTE y8092 = (v8092 = 1 & v8093 > 0) | (v8092 = 2 & Missing(v8093)).
FREQUENCIES y8092.
IF (y8092 = 0) v8093 = $SYSMIS.
EXECUTE.
DELETE VARIABLES y8092.

* 6. Verificar que v8094 sea 1 y que v8095 sea mayor que cero. Al mismo tiempo, verificar que v8094 sea 2 y que v8095 sea
missing.
COMPUTE y8094 = (v8094 = 1 & v8095 > 0) | (v8094 = 2 & Missing(v8095)).
FREQUENCIES y8094.
IF (y8094 = 0) v8095 = $SYSMIS.
EXECUTE.
DELETE VARIABLES y8094.

* 7. Verificar que v8096 sea 1 y que v8097 sea mayor que cero. Al mismo tiempo, verificar que v8096 sea 2 y que v8097 sea
missing.
COMPUTE y8096 = (v8096 = 1 & v8097 > 0) | (v8096 = 2 & Missing(v8097)).
FREQUENCIES y8096.
DELETE VARIABLES y8096.
```

```
* 8. Verificar que todas las empresas con "No" en las variables binarias del Cap. 8 tengan por valor "Missing" en v8098, v8099, v8100.
COMPUTE NO_Cap8 = SUM(v8086, v8088, v8090, v8092, v8094, v8096).
FREQUENCIES NO_Cap8.
COMPUTE Flag = (NO_Cap8 = 12 & SUM(v8098, v8099, v8100) = 0).
FREQUENCIES Flag.
DO IF (Flag = 1).
   COMPUTE v8098 = $SYSMIS.
   COMPUTE v8099 = $SYSMIS.
   COMPUTE v8100 = $SYSMIS.
END IF.
EXECUTE.
DELETE VARIABLES Flag NO_Cap8.

* 9. Verificar que si v8086 = 2 y v8092 = 2 entonces v8087 = v8093 = v8098 = $SYSMIS.
DO IF (v8086 = 2 & v8092 = 2).
   COMPUTE v8087 = $SYSMIS.
   COMPUTE v8093 = $SYSMIS.
   COMPUTE v8098 = $SYSMIS.
END IF.
EXECUTE.

* 10. Verificar que si v8088 = 2 y v8094 = 2 entonces v8089 = v8095 = v8099 = $SYSMIS.
DO IF (v8088 = 2 & v8094 = 2).
   COMPUTE v8089 = $SYSMIS.
   COMPUTE v8095 = $SYSMIS.
   COMPUTE v8099 = $SYSMIS.
END IF.
EXECUTE.

* 11. Verificar que si v8090 = 2 y v8096 = 2 entonces v8091 = v8097 = v8100 = $SYSMIS.
DO IF (v8090 = 2 & v8096 = 2).
   COMPUTE v8091 = $SYSMIS.
   COMPUTE v8097 = $SYSMIS.
   COMPUTE v8100 = $SYSMIS.
END IF.
EXECUTE.

****************************************
*** CAPÍTULO 9, SECCIÓN 1.
****************************************

* 12. Verificar que no hayan vacíos en la v9001 y v9002 sin justificación en v9003.
COMPUTE y9001 = (Missing(v9001) | Missing(v9002)) & (CHAR.LENGTH(v9003) = 0).
FREQUENCIES y9001.
DELETE VARIABLES y9001.

* 13. Verificar que todas las empresas que generan energía alternativa lo hagan también en la Tabla 9.i.3.
FREQUENCIES v9i2.
COMPUTE y9i2 = MIN(v9004, v9012, v9020, v9028, v9036, v9044).
FREQUENCIES y9i2.
CTABLES
  /VLABELS VARIABLES=v9i2 y9i2 DISPLAY=LABEL
  /TABLE v9i2 [COUNT F40.0] BY y9i2
  /CATEGORIES VARIABLES=v9i2 ORDER=A KEY=VALUE EMPTY=INCLUDE TOTAL=YES POSITION=AFTER
  /CATEGORIES VARIABLES=y9i2 ORDER=A KEY=VALUE EMPTY=EXCLUDE TOTAL=YES POSITION=AFTER
  /CRITERIA CILEVEL=95.
DELETE VARIABLES y9i2.

* 14. Corrección de empresa que Sí genera energía solar, pero no tiene ninguna información válida.
COMPUTE y9004 = (v9004 = 1 & v9005 = 0 & v9006 = 0 & v9007 = 0).
FREQUENCIES y9004.
DO IF (y9004 = 1).
   COMPUTE v9i2 = 2.
   COMPUTE v9004 = $SYSMIS.
   COMPUTE v9005 = $SYSMIS.
   COMPUTE v9006 = $SYSMIS.
```

```
    COMPUTE v9007 = $SYSMIS.
    COMPUTE v9008 = $SYSMIS.
    COMPUTE v9011 = "".
    COMPUTE v9012 = $SYSMIS.
    COMPUTE v9020 = $SYSMIS.
    COMPUTE v9028 = $SYSMIS.
    COMPUTE v9036 = $SYSMIS.
    COMPUTE v9044 = $SYSMIS.
    COMPUTE v9052 = $SYSMIS.
    COMPUTE v9053 = $SYSMIS.
    COMPUTE v9054 = $SYSMIS.
    COMPUTE v9055 = $SYSMIS.
    COMPUTE v9056 = $SYSMIS.
END IF.
EXECUTE.
DELETE VARIABLES y9004.

* 15. Imputación natural de valores missing a ceros en valores de producción, consumo y uso principal de energía solar.
DO IF (v9004 = 1).
    IF (v9006 = 0) v9006 = $SYSMIS.
    IF (v9007 = 0 | Missing(v9007)) v9008 = $SYSMIS.
    IF (v9007 = 0) v9007 = $SYSMIS.
END IF.
EXECUTE.

* 16. Imputación natural de valores missing a ceros en valores de producción, consumo y uso principal de energía solar.
DO IF (v9004 = 1 & v9009 = 0 & v9010 = 0).
    COMPUTE v9009 = $SYSMIS.
    COMPUTE v9010 = $SYSMIS.
END IF.
EXECUTE.

* 17. Imputación natural de valores missing a ceros en valores de producción, consumo y uso principal de energía eólica.
COMPUTE y9015 = (v9012 = 1 & v9015 = 0).
FREQUENCIES y9015.
DO IF (y9015 = 1).
    COMPUTE v9015 = $SYSMIS.
    COMPUTE v9016 = $SYSMIS.
END IF.
EXECUTE.
DELETE VARIABLES y9015.

* 18. Imputación natural de los ceros por missing en las variables de cantidad e ingresos por venta de energía de biomasa.
DO IF (v9020 = 1).
    IF (v9025 = 0) v9025 = $SYSMIS.
    IF (v9026 = 0) v9026 = $SYSMIS.
END IF.
EXECUTE.

* 19. Imputación natural de los ceros por missing en las variables de cantidad e ingresos por venta de energía hidráulica.
DO IF (v9028 = 1).
    IF (v9031 = 0) v9031 = $SYSMIS.
    IF (v9031 = 0 | Missing(v9031)) v9032 = $SYSMIS.
    IF (v9033 = 0) v9033 = $SYSMIS.
    IF (v9034 = 0) v9034 = $SYSMIS.
END IF.
EXECUTE.

* 20. Se imputa los valores faltantes de generación de energía termoeléctrica, tomando como costo promedio de generación
3.18¢USD / kWh, a partir de datos tomados del "Análisis y Determinación del Costo del Servicio Público de Energía Eléctrica.
Período Enero - Diciembre 2022. Informe N°. DRETSE-2021-044", página 19, de la ARCERNNR. URL de la publicación:
* https://www.controlrecursosyenergia.gob.ec/wp-
content/uploads/downloads/2022/02/2._informe_analisis_y_determinacion_costos_spee_2022.pdf.
DO IF (v9036 = 1).
    IF (v9037 = 0 & v9038 ~= 0) v9037 = v9038 / 0.0318.
    IF (v9037 ~= 0 & v9038 = 0) v9038 = v9037 * 0.0318.
    DO IF (v9037 = 0 & v9038 = 0 & v9039 = 0).
```

```
      COMPUTE v9036 = 2.
      COMPUTE v9037 = $SYSMIS.
      COMPUTE v9038 = $SYSMIS.
      COMPUTE v9039 = $SYSMIS.
      COMPUTE v9040 = $SYSMIS.
      COMPUTE v9041 = $SYSMIS.
      COMPUTE v9042 = $SYSMIS.
      COMPUTE v9043 = "".
   END IF.
   DO IF (v9039 = 0).
      COMPUTE v9039 = $SYSMIS.
      COMPUTE v9040 = $SYSMIS.
      COMPUTE v9041 = $SYSMIS.
      COMPUTE v9042 = $SYSMIS.
      COMPUTE v9043 = "".
   END IF.
   DO IF (v9041 = 0 | v9042 = 0).
      COMPUTE v9041 = $SYSMIS.
      COMPUTE v9042 = $SYSMIS.
   END IF.
END IF.
EXECUTE.

* 21. Imputación natural de los ceros por missing en las variables de cantidad e ingresos por venta de energía de otra fuente alternativa.
DO IF (v9044 = 1).
   DO IF (v9049 = 0 & v9050 = 0).
      COMPUTE v9049 = $SYSMIS.
      COMPUTE v9050 = $SYSMIS.
   END IF.
   IF (v9047 = 0) v9047 = $SYSMIS.
END IF.
EXECUTE.

* 22. Recálculo de la variable filtro v9i2.
COMPUTE NUEVO_v9i2 = MIN(v9004, v9012, v9020, v9028, v9036, v9044).
FREQUENCIES NUEVO_v9i2 v9i2.

DO IF (v9i2 = 1 & NUEVO_v9i2 = 2).
   COMPUTE v9i2 = 2.
   COMPUTE v9004 = $SYSMIS.
   COMPUTE v9012 = $SYSMIS.
   COMPUTE v9020 = $SYSMIS.
   COMPUTE v9028 = $SYSMIS.
   COMPUTE v9036 = $SYSMIS.
   COMPUTE v9044 = $SYSMIS.
   COMPUTE v9052 = $SYSMIS.
   COMPUTE v9053 = $SYSMIS.
   COMPUTE v9054 = $SYSMIS.
   COMPUTE v9055 = $SYSMIS.
   COMPUTE v9056 = $SYSMIS.
END IF.
EXECUTE.

DELETE VARIABLES NUEVO_v9i2.

* 23. Conversión en missing de los valores nulos de las variables totalizadoras de energía.
COMPUTE Fil_Energ = sum.1(v9052, v9053, v9054, v9055, v9056).
FREQUENCIES Fil_Energ.

DO IF (v9i2 = 2 & Fil_Energ = 0).
   COMPUTE v9052 = $SYSMIS.
   COMPUTE v9053 = $SYSMIS.
   COMPUTE v9054 = $SYSMIS.
   COMPUTE v9055 = $SYSMIS.
   COMPUTE v9056 = $SYSMIS.
END IF.
```

```
DO IF (v9i2 = 1).
   IF (v9052 = 0) v9052 = $SYSMIS.
   IF (v9053 = 0) v9053 = $SYSMIS.
   IF (v9054 = 0) v9054 = $SYSMIS.
   IF (v9055 = 0) v9055 = $SYSMIS.
   IF (v9056 = 0) v9056 = $SYSMIS.
END IF.
EXECUTE.

DELETE VARIABLES Fil_Energ.

* 24. Hacer missing los ceros de la v9001 y v9002.
IF (v9001 = 0) v9001 = $SYSMIS.
IF (v9002 = 0) v9002 = $SYSMIS.
EXECUTE.

IF (v9001 > 0 & v9002 > 0) v9003 = "".
EXECUTE.

****************************************
*** CAPÍTULO 9, SECCIÓN 2.
****************************************

* 25. Corrección de la v9ii1 para empresas que consumieron combustibles y/o lubricantes, pero que no aparecen con v9ii1 = 1.
COMPUTE Tiene_Comb = (v9105 > 0).
FREQUENCIES Tiene_Comb v9ii1.

COMPUTE Flag = (Tiene_Comb = 1 & v9ii1 = 2).
FREQUENCIES Flag.

IF (inec_identificador_empresa = '46833489097') v9ii1 = 1.
IF (inec_identificador_empresa = '46967439070') v9ii1 = 1.
FREQUENCIES v9ii1.

IF (v9060 = 0) v9060 = $SYSMIS.
IF (v9064 = 0) v9064 = $SYSMIS.
IF (v9068 = 0) v9068 = $SYSMIS.
IF (v9072 = 0) v9072 = $SYSMIS.
IF (v9076 = 0) v9076 = $SYSMIS.
IF (v9080 = 0) v9080 = $SYSMIS.
IF (v9084 = 0) v9084 = $SYSMIS.
IF (v9088 = 0) v9088 = $SYSMIS.
IF (v9092 = 0) v9092 = $SYSMIS.
IF (v9096 = 0) v9096 = $SYSMIS.
IF (v9100 = 0) v9100 = $SYSMIS.
IF (v9103 = 0) v9103 = $SYSMIS.
EXECUTE.

DO IF (v9ii1 = 1 & v9105 = 0).
   COMPUTE v9ii1 = 2.
   COMPUTE v9105 = $SYSMIS.
END IF.
EXECUTE.

DELETE VARIABLES Flag Tiene_Comb.

****************************************
*** CAPÍTULO 10, SECCIÓN 1.
****************************************

* 26. Hacer missing los ceros de la v10000 y v10001.
IF (v10000 = 0) v10000 = $SYSMIS.
IF (v10001 = 0) v10001 = $SYSMIS.
EXECUTE.

IF (v10000 > 0 & v10001 > 0) v10002 = "".
EXECUTE.
```

```
* 27. Verificación de coherencia entre la variable v10i2 y las variables de consumo de agua de tanquero.
DO IF (v10i2 = 1 & v10003 = 2 & v10004 = 0 & v10005 = 0).
   COMPUTE v10003 = $SYSMIS.
   COMPUTE v10004 = $SYSMIS.
   COMPUTE v10005 = $SYSMIS.
END IF.
EXECUTE.

* 28. Verificación de coherencia entre la variable v10i3 y las variables v10006, v10014 y v10022 de captación de fuentes naturales de agua.
COMPUTE Capta_Agua = SUM.1(v10006, v10014, v10022).
FREQUENCIES Capta_Agua v10i3.

DO IF (v10i3 = 1 & v10006 = 0 & v10007 = 0 & v10008 = 0).
   COMPUTE v10006 = 2.
   COMPUTE v10007 = $SYSMIS.
   COMPUTE v10008 = $SYSMIS.
   COMPUTE v10012 = $SYSMIS.
END IF.
EXECUTE.

IF (v10013 <= 1) v10013 = $SYSMIS.
EXECUTE.

DO IF (v10i3 = 1 & v10014 = 0 & v10015 = 0 & v10016 = 0).
   COMPUTE v10014 = 2.
   COMPUTE v10015 = $SYSMIS.
   COMPUTE v10016 = $SYSMIS.
   COMPUTE v10020 = $SYSMIS.
END IF.
EXECUTE.

IF (v10i3 = 1 & v10014 = 1 & v10015 = 1 & v10016 = 0) v10016 = 1.
EXECUTE.

IF (v10021 <= 1) v10021 = $SYSMIS.
EXECUTE.

DO IF (v10i3 = 1 & v10022 = 0 & v10023 = 0 & v10024 = 0).
   COMPUTE v10022 = 2.
   COMPUTE v10023 = $SYSMIS.
   COMPUTE v10024 = $SYSMIS.
   COMPUTE v10028 = $SYSMIS.
END IF.
EXECUTE.

IF (v10029 <= 1) v10029 = $SYSMIS.
EXECUTE.

IF (v10028 = 3720) v10025 = 1.
EXECUTE.

DELETE VARIABLES Capta_Agua.

COMPUTE Capta_Agua = MIN(v10006, v10014, v10022).
FREQUENCIES Capta_Agua v10i3.
FREQUENCIES v10006 v10014 v10022.

IF (v10030 = 0) v10030 = $SYSMIS.
IF (v10031 <= 1) v10031 = $SYSMIS.
EXECUTE.

DELETE VARIABLES Capta_Agua.

*****************************************
*** CAPÍTULO 10, SECCIÓN 2.
```

```
*****************************************

* 29. Encerar la variable v10ii6 cuando v10ii2 = 1, v10ii3 = 2, v10ii5 = 2 y v10ii6 >= 0. Esto es, cuando las empresas generan
aguas residuales, pero no les dan ningún tipo de tratamiento.
IF (v10ii2 = 1 & v10ii3 = 2 & v10ii5 = 2 & v10ii6 >= 0) v10ii6 = $SYSMIS.
EXECUTE.

*****************************************
*** CAPÍTULO 10, SECCIÓN 3.
*****************************************

* 30. Recuperación de valores de la variable v10iii1_3 por consulta a las Coordinaciones Zonales.
FREQUENCIES v10iii1_3.

* Zonal Sur.
IF (inec_identificador_empresa = '13602010010') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13648597074') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13650063079') v10iii1_3 = 2.
IF (inec_identificador_empresa = '14629094033') v10iii1_3 = 2.
IF (inec_identificador_empresa = '14777668014') v10iii1_3 = 2.
IF (inec_identificador_empresa = '14834744075') v10iii1_3 = 2.
IF (inec_identificador_empresa = '47280000091') v10iii1_3 = 1.
FREQUENCIES v10iii1_3.

* Zonal Litoral.
IF (inec_identificador_empresa = '13706139090') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13706828090') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13708437098') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13710186094') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13825679175') v10iii1_3 = 1.
IF (inec_identificador_empresa = '14595074097') v10iii1_3 = 2.
IF (inec_identificador_empresa = '43314669178') v10iii1_3 = 2.
IF (inec_identificador_empresa = '43462753099') v10iii1_3 = 2.
IF (inec_identificador_empresa = '47051683095') v10iii1_3 = 2.
IF (inec_identificador_empresa = '47058469099') v10iii1_3 = 2.
FREQUENCIES v10iii1_3.

* Zonal DICA.
IF (inec_identificador_empresa = '13706539098') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13824630172') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13824656171') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13828497175') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13828695171') v10iii1_3 = 2.
IF (inec_identificador_empresa = '13828939178') v10iii1_3 = 1.
IF (inec_identificador_empresa = '13829071171') v10iii1_3 = 2.
IF (inec_identificador_empresa = '14604743177') v10iii1_3 = $SYSMIS.
IF (inec_identificador_empresa = '14689285172') v10iii1_3 = 2.
IF (inec_identificador_empresa = '44481927177') v10iii1_3 = 2.
IF (inec_identificador_empresa = '46828150171') v10iii1_3 = 2.
IF (inec_identificador_empresa = '46941105173') v10iii1_3 = 2.
IF (inec_identificador_empresa = '47578526176') v10iii1_3 = 2.
FREQUENCIES v10iii1_3.

* Zonal Centro.
IF (inec_identificador_empresa = '13828238170') v10iii1_3 = 1.
FREQUENCIES v10iii1_3.


*****************************************************************************************************
*** CAMBIOS A REALIZAR POR DETECCIÓN, POSTERIOR A LA FASE DE TABULADOS,
*** DE VALORES QUE SESGAN LOS RESULTADOS A PUBLICAR. TODOS LOS DATOS
*** ACTUALIZADOS FUERON REINVESTIGADOS POR LOS RESPONSABLES ZONALES.
*****************************************************************************************************

* #1. Primera empresa.
DO IF (inec_identificador_empresa = '13707376095').
   COMPUTE v8093 = 13906794.
```

```
   COMPUTE v8098 = 13906794.
END IF.

* #2. Segunda empresa.
DO IF (inec_identificador_empresa = '13850415239').
   COMPUTE v8087 = 5739225.
   COMPUTE v8098 = 5739225.
END IF.

* #3. Tercera empresa.
DO IF (inec_identificador_empresa = '14738926171').
   COMPUTE v8093 = 6157051.
   COMPUTE v8098 = 39040903.
   COMPUTE v9034 = 6157051.
   COMPUTE v9056 = 6157051.
END IF.

* #4. Cuarta empresa.
DO IF (inec_identificador_empresa = '14650448171').
   COMPUTE v10032 = 9.15.
   COMPUTE v10035 = 36892.8.
END IF.

* #5. Quinta empresa.
DO IF (inec_identificador_empresa = '13763885130').
   COMPUTE v9i2 = 1.
   COMPUTE v9036 = 1.
   COMPUTE v9037 = 4800.
   COMPUTE v9038 = 2965.
   COMPUTE v9039 = 4800.
   COMPUTE v9052 = 4800.
   COMPUTE v9053 = 2965.
   COMPUTE v9054 = 4800.
END IF.

* #6. Sexta empresa.
DO IF (inec_identificador_empresa = '47280000091').
   COMPUTE v8088 = 2.
   COMPUTE v8089 = $SYSMIS.
   COMPUTE v8099 = $SYSMIS.
END IF.

* #7. Séptima empresa.
DO IF (inec_identificador_empresa = '13825551179').
   COMPUTE v8088 = 2.
   COMPUTE v8089 = $SYSMIS.
   COMPUTE v8099 = $SYSMIS.
END IF.

* #8. Octava empresa.
DO IF (inec_identificador_empresa = '14716231144').
   COMPUTE v7005 = 71956.
   COMPUTE v8090 = 1.
   COMPUTE v8091 = 71956.
   COMPUTE v8092 = 2.
   COMPUTE v8093 = $SYSMIS.
   COMPUTE v8096 = 2.
   COMPUTE v8097 = $SYSMIS.
   COMPUTE v8098 = 162645.
   COMPUTE v8100 = 71956.
END IF.

* #9. Novena empresa.
DO IF (inec_identificador_empresa = '13707900096').
   COMPUTE v8087 = 54607236.
   COMPUTE v8098 = 54607236.
END IF.
```

```
* #10. Décima empresa.
DO IF (inec_identificador_empresa = '14633469173').
   COMPUTE v8088 = 2.
   COMPUTE v8089 = $SYSMIS.
   COMPUTE v8099 = $SYSMIS.
END IF.

* #11. Corrección del valor de venta de energía de generador termoeléctrico.
IF (inec_identificador_empresa = '13602431015') = v9042 / 1.467.
# El valor $0.729/kWh se obtuvo del tercer cuartil del cociente: v9042 / v9041.

DO IF (inec_identificador_empresa = '13602431015').
   COMPUTE v9052 = SUM.1(v9052, v9037).
   COMPUTE v9055 = SUM.1(v9055, v9041).
END IF.

* #12. Corrección del valor de producción de energía de bienes y servicios ambientales.
DO IF (inec_identificador_empresa = '13602431015').
   COMPUTE v9052 = SUM.1(v9052, v9037).
   COMPUTE v9055 = SUM.1(v9055, v9041).
END IF.

EXECUTE.

*****************************************************************
*** FIN DEL CÓDIGO DE CONTROL DE INTEGRIDAD
*** MÓDULO DE INFORMACIÓN ECONÓMICA AMBIENTAL
*** ENESEM 2022.
*****************************************************************
```

```
# Código R Script de integridad de REsiduos y Desechos
# BDD ENESEM 2022
# Realizado por: Ramiro Benavides L.
# Fecha de creación: 15/01/2024

# 0. Creación de secuencia numérica de sufijos de variables de generación de residuos y/o desechos.
attach(B)
sgenera <- c(seq(10062, 10272, 21), 10314, seq(10377, 10461, 21), 10524, seq(10545, 10986, 21))
vgenera <- paste0("v", sgenera)
cond <- sapply(sgenera, function(x) paste0("which(", paste0("v", sgenera), " == 1 & ", paste0("v", sgenera+3), " == 0)"))
cond <- cond[1:40]

# Reordenar toda la BDD por identificador de empresa en orden ascendente.
detach(B)
B <- B[order(B$inec_identificador_empresa), ]

# 1. Creación de banderas de cuáles empresas generan residuos sin cantidades.
attach(B)
casos_gen_no_cant <- sapply(1:40, function(x) eval(parse(text = cond[x]), envir = .GlobalEnv))

# 2. Tabulación de las variables de unidad de medida de residuos y/o desechos.
vumed <- paste0("v", sgenera+2)
tabla_u_med <- sapply(vumed, function(x) eval(parse(text = paste0("table(", x, ")")), envir = .GlobalEnv))

# 3. Transformación de valores <> 0 de Col. 2.1 y ceros de Col. 2.2 en missing.
vcant <- paste0("v", sgenera+3)
detach(B)
for (i in 1:40) {
  B[casos_gen_no_cant[[i]], vumed[i]] <- NA_integer_
  B[casos_gen_no_cant[[i]], vcant[i]] <- NA_real_
}
attach(B)
tabla_u_med <- sapply(vumed, function(x) eval(parse(text = paste0("table(", x, ")")), envir = .GlobalEnv))
tabla_cant <- sapply(vcant, function(x) eval(parse(text = paste0("table(", x, ")")), envir = .GlobalEnv))
cant_min <- sapply(1:40, function(x) min(tabla_cant[[x]]))
```

```
# 4. Determinar si las empresas con "Sí" en v10iii1_3 coincide con las que generan al menos un desecho peligroso o especial.
vmatrizgen <- vgenera[-(1:12)]
empmatrizgen <- union(which(B[, vmatrizgen[1]] == 1), which(B[, vmatrizgen[2]] == 1))
empmatriznogen <- intersect(which(B[, vmatrizgen[1]] == 2), which(B[, vmatrizgen[2]] == 2))
for (i in 2:27) {
  empmatrizgen <- union(empmatrizgen, union(which(B[, vmatrizgen[i]] == 1), which(B[, vmatrizgen[i+1]] == 1)))
  empmatriznogen <- intersect(empmatriznogen, intersect(which(B[, vmatrizgen[i]] == 2), which(B[, vmatrizgen[i+1]] == 2)))
}
table(v10iii1_3, useNA = "ifany")
letra_CIIU <- substring(B$ciiu4_actividad_principal, 1, 1)

# 5. Determinación de empresas con "Sí" en Col. 1, pero en Col 2.2 tienen missing.
cond <- sapply(sgenera, function(x) paste0("which(", paste0("v", sgenera), " == 1 & ", paste0("v", sgenera+2), " > 0 & is.na(", paste0("v", sgenera+3), "))"))
cond <- cond[1:40]
empgenmiss <- sapply(1:40, function(x) eval(parse(text = cond[x]), envir = .GlobalEnv))

# 6. Transformación de valores <> 0 de Col. 2.1 y ceros de Col. 2.2 en missing.
detach(B)
for (i in 1:40) {
  B[empgenmiss[[i]], vumed[i]] <- NA_integer_
}
attach(B)
tabla_u_med <- sapply(vumed, function(x) eval(parse(text = paste0("table(", x, ")")), envir = .GlobalEnv))
tabla_cant <- sapply(vcant, function(x) eval(parse(text = paste0("table(", x, ")")), envir = .GlobalEnv))

# 7. Casos particulares
B$v10iii1_2[which(is.na(B$v10iii1_2) & B$v10078_1 == 40)] <- 1
B$v10379[which(B$v10377 == 1 & B$v10379 == 2 & is.na(B$v10380))] <- NA_integer_
B$v10iii2_2[which(B$v10iii2_2 == 2 & B$v10078_2 > 0)] <- 1

B$v10085[which(B$v10083 == 1 & B$v10085 == 1 & is.na(v10086))] <- NA_integer_
B$v10127[which(B$v10125 == 1 & B$v10127 == 1 & is.na(v10128))] <- NA_integer_

B$v10631[which(B$v10629 == 1 & B$v10631 == 1 & is.na(v10632))] <- NA_integer_
B$v10652[which(B$v10650 == 1 & B$v10652 == 1 & is.na(v10653))] <- NA_integer_
B$v10715[which(B$v10713 == 1 & B$v10715 == 1 & is.na(v10716))] <- NA_integer_
B$v10736[which(B$v10734 == 1 & B$v10736 == 1 & is.na(v10737))] <- NA_integer_
B$v10925[which(B$v10923 == 1 & B$v10925 == 3 & is.na(v10926))] <- NA_integer_


# 8. Creación de la BDD Módulo Ambiental ENESEM 2022 con las variables íntegras para Desechos y/o Residuos.
haven::write_sav(B, "C:/Users/rbenavides/Documents/EMPRESAS 2022/Proceso de Integridad/BDD_Integra_RESIDUOS.sav",
compress =TRUE)

#-----------------------------------------------------------------------------------------------------------------------
# FIN DE LA SINTAXIS DE CONTROL DE INTEGRIDAD DE LA BDD AMBIENTAL ENESEM 2022.
#-----------------------------------------------------------------------------------------------------------------------
```

ANEXO C. Código R Markdown para las validaciones especiales.

```
---
title: "Validaciones especiales ENESEM 2022"
author: "Ramiro Benavides"
date: "07/10/2021"
update: "24/08/2023"
output: html_document
---
```

1. Carga de la base de datos ENESEM 2022 en versión Excel

```
```{r}
# BDD_10mo <- readxl::read_excel("~/EMPRESAS 2022/10mo Corte - 18-10-2023/BDD/BDD_10mo_18_10_2023.xlsx")
# NOTA: Se recomienda trabajar cargando la versión SPSS del archivo anterior, para evitar caracteres extraños.
# Intersección de identificadores de la BDD global con la base efectiva y con críticos.
# id <- intersect(B_2022$inec_identificador_empresa, BDD_10mo$inec_identificador_empresa)
# ind <- which(BDD_10mo$inec_identificador_empresa %in% id)
```
```

2. Generación de la BDD de trabajo.

```
```{r}
# B_2022_orig <- B_2022
# observacion <- NA_character_
# BDD <- cbind(B_2021_orig, observacion)
# # BDD <- BDD[which(BDD$nombre_critico != "" & BDD$estado == "C"), ]
# BDD <- BDD[BDD$fusionada!="1.7" & BDD$absorbida!="1.8", ]
# BDD <- BDD[BDD$efectividad == 1, ]
#BDD <- BDD[which(BDD$inec_identificador_empresa %in% id), ]
ind <- 1:dim(BDD)[1]

lista_vars_identif <- c("secuencial_sistema", "inec_identificador_empresa",  "Zonal_DEAGA",       "Estado",
                        "ciiu4_actividad_principal")
# "Origen" es la BDD que indica el origen histórico de cada empresa.
# Origen <- readxl::read_excel("~/EMPRESAS 2022/310mo Corte - 18-12-2023/BDD/Empresas_val3.xlsx")
# Zonal_DEAGA <- Origen$Zonal_D[Origen$inec_identificador_empresa %in% BDD$inec_identificador_empresa]
# Zonal_DEAGA[Zonal_DEAGA == "AC CAMPO"] <- "DICA"
# Zonal_DEAGA[Zonal_DEAGA == "CENTRO"] <- "Centro"
# Zonal_DEAGA[Zonal_DEAGA == "LITORAL"] <- "Litoral"
# Zonal_DEAGA[Zonal_DEAGA == "SUR"] <- "Sur"
# BDD <- cbind(BDD, Zonal_DEAGA)
```
```

3. Generación de las diferentes hojas de Energía del libro "Reporte Otros Usos Energía y Combustibles_BDD_30_10_2022.xlsx".

```
```{r}
# Usar la lista de variables "lista_vars_identif" de la corrida de comparaciones.
E_Solar <- BDD[ind, c(lista_vars_identif, "v9004", "v9008", "v9011", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Solar <- E_Solar[which(E_Solar$v9004 == 1 & E_Solar$v9008 == 3), ]

E_Eolica <- BDD[ind, c(lista_vars_identif, "v9012", "v9016", "v9019", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Eolica <- E_Eolica[which(E_Eolica$v9012 == 1 & E_Eolica$v9016 == 3), ]

E_Biomasa <- BDD[ind, c(lista_vars_identif, "v9020", "v9024", "v9027", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Biomasa <- E_Biomasa[which(E_Biomasa$v9020 == 1 & E_Biomasa$v9024 == 3), ]

E_Hidraulica <- BDD[ind, c(lista_vars_identif, "v9028", "v9032", "v9035", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Hidraulica <- E_Hidraulica[which(E_Hidraulica$v9028 == 1 & E_Hidraulica$v9032 == 3), ]

E_Generador <- BDD[ind, c(lista_vars_identif, "v9036", "v9040", "v9043", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Generador <- E_Generador[which(E_Generador$v9036 == 1 & E_Generador$v9040 == 3), ]
```
```

```
E_Otro <- BDD[ind, c(lista_vars_identif, "v9057", "v9044", "v9048", "v9051", "observaciones", "observaciones_investigador",
"observaciones_critica")]
E_Otro <- E_Otro[which(E_Otro$v9044 == 1 & E_Otro$v9048 == 3), ]
```

4. Generación de las diferentes hojas de Combustibles del libro "Reporte Otros Usos Energía y Combustibles_BDD_30_10_2022.xlsx".

```{r}
# Usar la lista de variables "lista_vars_identif" de la corrida de comparaciones.
G_Super <- BDD[ind, c(lista_vars_identif, "v9060", "v9061", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Super <- G_Super[which(G_Super$v9060 == 6), ]

G_Extra <- BDD[ind, c(lista_vars_identif, "v9064", "v9065", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Extra <- G_Extra[which(G_Extra$v9064 == 6), ]

G_Jet <- BDD[ind, c(lista_vars_identif, "v9068", "v9069", "observaciones", "observaciones_investigador", "observaciones_critica")]
G_Jet <- G_Jet[which(G_Jet$v9068 == 6), ]

G_Diesel <- BDD[ind, c(lista_vars_identif, "v9072", "v9073", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Diesel <- G_Diesel[which(G_Diesel$v9072 == 6), ]

G_GLP <- BDD[ind, c(lista_vars_identif, "v9076", "v9077", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_GLP <- G_GLP[which(G_GLP$v9076 == 6), ]

G_Gas <- BDD[ind, c(lista_vars_identif, "v9080", "v9081", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Gas <- G_Gas[which(G_Gas$v9080 == 6), ]

G_Fuel <- BDD[ind, c(lista_vars_identif, "v9084", "v9085", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Fuel <- G_Fuel[which(G_Fuel$v9084 == 6), ]

G_Crudo <- BDD[ind, c(lista_vars_identif, "v9088", "v9089", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Crudo <- G_Crudo[which(G_Crudo$v9088 == 6), ]

G_Carbon <- BDD[ind, c(lista_vars_identif, "v9092", "v9093", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Carbon <- G_Carbon[which(G_Carbon$v9092 == 6), ]

G_Ecopais <- BDD[ind, c(lista_vars_identif, "v9096", "v9097", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Ecopais <- G_Ecopais[which(G_Ecopais$v9096 == 6), ]

G_Aceites <- BDD[ind, c(lista_vars_identif, "v9100", "v9101", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Aceites <- G_Aceites[which(G_Aceites$v9100 == 6), ]

G_Otros <- BDD[ind, c(lista_vars_identif, "otro_combustible", "v9103", "v9104", "observaciones", "observaciones_investigador",
"observaciones_critica")]
G_Otros <- G_Otros[which(G_Otros$v9103 == 6), ]
```

5. Reporte del libro "Reporte Otros Usos Energía y Combustibles_BDD_30_10_2022.xlsx".

```{r}
writexl::write_xlsx(list("Otros usos - E. Solar" = E_Solar,
                         "Otros usos - E. Eólica" = E_Eolica,
                         "Otros usos - E. Biomasa" = E_Biomasa,
                         "Otros usos - E. Hidráulica" = E_Hidraulica,
                         "Otros usos - G. Termoeléctrico" = E_Generador,
                         "Otros usos - Otro tipo de energía" = E_Otro,
                         "Otros usos - Gasolina Súper" = G_Super,
                         "Otros usos - Gasolina Extra" = G_Extra,
```

```
                "Otros usos - Jet Fuel" = G_Jet,
                "Otros usos - Diesel" = G_Diesel,
                "Otros usos - GLP" = G_GLP,
                "Otros usos - Gas natural" = G_Gas,
                "Otros usos - Residuo Fuel Oil" = G_Fuel,
                "Otros usos - Crudo residual" = G_Crudo,
                "Otros usos - Carbón" = G_Carbon,
                "Otros usos - Gasolina Ecopaís" = G_Ecopais,
                "Otros usos - Aceites" = G_Aceites,
                "Otros usos - Otros combustibles" = G_Otros),
              "~/EMPRESAS 2022/10mo Corte- 18-12-2023/VAL ESPECIALES/Reporte Otros Usos Energía y
Combustibles_BDD_18_12_2023.xlsx")
```

6. Generación de las diferentes hojas de reporte del libro "Reporte Otros Capítulos 7 y 9_BDD_30_10_2022.xlsx".

```{r}
# Usar la lista de variables "lista_vars_identif" de la corrida de comparaciones.
# Reporte_7A <- BDD[ind, c(lista_vars_identif, "v7024", "v7025", "observacion")]
# Reporte_7A <- Reporte_7A[which(Reporte_7A$v7024 > 0), ]

# Reporte_7B <- BDD[ind, c(lista_vars_identif, "v75", "v751", "v7511", "observacion")]
# Reporte_7B <- Reporte_7B[which(Reporte_7B$v75 == 1 & Reporte_7B$v751 == 4), ]

Reporte_9A <- BDD[ind, c(lista_vars_identif, "v9044", "v9057", "v9051", "observaciones", "observaciones_investigador",
"observaciones_critica")]
Reporte_9A <- Reporte_9A[which(Reporte_9A$v9044 == 1), ]
```

7. Reporte del libro "Reporte Otros Capítulos 7 y 9_BDD_30_10_2022.xlsx".

```{r}
# writexl::write_xlsx(list("Reporte 7A" = Reporte_7A,
                #     "Reporte 7B" = Reporte_7B,
                #     "Reporte 9A" = Reporte_9A),
                # "~/EMPRESAS 2022/10mo Corte - 18-12-2023/Validaciones especiales/Reporte Otros Capítulos 7 y
9_BDD_18_12_2023.xlsx")

writexl::write_xlsx(list("Reporte 9A" = Reporte_9A), "~/EMPRESAS 2022/10mo Corte- 18-12-2023/VAL ESPECIALES/Reporte
Otros Capítulos 7 y 9_BDD_18_12_2023.xlsx")
```

ANEXO D. Código R Markdown para las comparaciones de valores 2022-2021.

---
title: "SX Comparación BDD 2022-2021"
author: "Ramiro Benavides"
date: "10/08/2023"
update: "14/08/2023"
output: html_document

A. Carga de las BDD 2021 y 2020 del Módulo Ambiental ENESEM.

```{r}
# B_2022 es el frame que carga la BDD a validar del último corte criticado.
B_2022 <- haven::read_sav("~/EMPRESAS 2022/3er Corte - 03-10-2022/BDD/Empresas_03_10_2022.sav")

# B_2021 es el frame de la base de publicación 2021.
B_2021 <- readxl::read_excel("~/EMPRESAS 2021/Herramientas/BDD_Original_Validada_e_Integra_4052e_AMB.xlsx")

# "Origen" es la BDD que tiene la información de zonal de crítica por empresa.
# Origen <- readxl::read_excel("~/EMPRESAS 2021/3er Corte - 03-10-2022/BDD/Empresas_val3.xlsx")

# B_2021_BKUP es el respaldo del frame "B_2021".
B_2022_BK <- B_2022
B_2021_BK <- B_2021

# El frame C es la juntura de los frames 2021 y 2022. Sirve para determinar la intersección de identificadores
# entre las empresas 2022 criticadas y las empresas 2021 publicadas.
C <- merge(B_2022, B_2021, by.x="inec_identificador_empresa", by.y="inec_identificador_empresa")
id <- C$inec_identificador_empresa
```

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(7002, 7003, 7004, 7005, 7006, 8001, 8098, 8099, 8100, 9001, 9002, 9052, 9053, 9054, 9055, 9056,
                       9058, 9062, 9066, 9070, 9074, 9078, 9082, 9086, 9090, 9094, 9098, 9105, 10000, 10001, 10030)
var_sin_trans <- paste0("v", sec_var_sin_trans)
var_sin_trans <- c(var_sin_trans, "v10ii11", "v10ii5", "v10ii6")

sec_var_trans <- c(10004, seq(10065, 10275, 21), 10317, seq(10380, 10464, 21), 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 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 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_2022 <- B
```

D. Creación de variables transformadas a unidades estándar para el frame del año 2021.

```{r}
B <- B_2021
sec_var_trans <- sec_var_trans[-(2:13)]
var_trans <- paste0("v", sec_var_trans)
# 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_2021 <- B
sec_var_trans <- c(10004, seq(10065, 10275, 21), 10317, seq(10380, 10464, 21), 10527)
var_trans <- paste0("v", sec_var_trans)
rm(B, i, j)
```

E. 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}
E <- data.frame(matrix(NA_real_, nrow=length(id), ncol=6*length(var_chk) + 5))
ind_2021 <- sapply(id, function(x) which(D_2021$inec_identificador_empresa == x))
ind_2022 <- sapply(id, function(x) which(D_2022$inec_identificador_empresa == x))
for (j in 1:length(var_chk)) {
  for (i in 1:length(id)) {
    a2021 <- E[i, 0+6*j] <- ifelse(var_chk[j] %in% names(D_2021), as.numeric(D_2021[ind_2021[i], var_chk[j]]), NA_real_)
    a2022 <- E[i, 1+6*j] <- as.numeric(D_2022[ind_2022[i], var_chk[j]])
    if (isTRUE(a2021 > 0)) {
      E[i, 2+6*j] <- ifelse(is.na(a2022) | a2022 == 0, -1, (a2022 - a2021) / a2021)
    } 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 2021"
  names(E)[1+6*j] <- "Año 2022"
  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 2021 y 2022 de la variable ", var_chk[j]))
}

E[sapply(E[], is.nan)] <- NA_real_
E[sapply(E[], is.infinite)] <- NA_real_
E[, 1:5] <- B_2022[ind_2022, lista_vars_identif]
names(E)[1:5] <- lista_vars_identif

rm(a2021, a2022, i, j)
reporte_BKUP <- reporte <- E
```

F. Rutina para cargar un Excel con justificaciones de zonales y dejar únicamente los códigos 1 y 2.

```{r}
Z <- SUR_7 # En el frame "Z" se carga el Excel de la respuesta de la zonal que se quiera cargar.
id <- Z$inec_identificador_empresa
nom_vars <- names(Z)
ind_just <- grep("^Just", names(Z))
Z <- Z[, ind_just]
names(Z) <- nom_vars[ind_just - 1]
Z <- cbind(id, Z)
names(Z)[1] <- "inec_identificador_empresa"
S7ma <- Z
# M1[] <- lapply(M, function(x) replace(x, is.infinite(x), NA))
```

```
rm(ind_just, nom_vars, Z)
```

G. Función de borrado de celdas adyacentes a una celda de coordenadas dadas (id_emp, variable).

```{r}
BorrarCeldas <- function(BDD, id_emp, variable) {
  comando1 <- paste0("ind_col <- which(names(", BDD, ") %in% '", variable, "')")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0("ind_fil <- which(", BDD, "$inec_identificador_empresa %in% '", id_emp, "')")
  eval(parse(text = comando1), envir = .GlobalEnv)

  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)

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col + 1, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col + 2, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0(BDD, "[", ind_fil, ", ", ind_col + 3, "] <- NA")
  eval(parse(text = comando1), envir = .GlobalEnv)

  comando1 <- paste0("rm(ind_fil, ind_col)")
  eval(parse(text = comando1), envir = .GlobalEnv)
}
```

H. Eliminación de casos (borrado de celdas) para las validaciones con justificación en los archivos: C2da,...,S2da y C3ra,...,S3ra., etc.

```{r}
# bases <- c("C2da", "C3ra", "C4ta", "C5ta", "C7ma", "C8va", "D2da", "D3ra", "D4ta", "D5ta", "D6ta", "D7ma", "D8va", "L2da",
"L3ra", "L4ta", "L5ta", "L6ta", "L7ma", "S2da", "S3ra", "S4ta", "S5ta", "S6ta", "S7ma", "S8va")
bases <- c("C2da", "C3ra", "C5ta", "C6ta", "C7ma", "D2da", "D3ra", "D5ta", "D6ta", "D7ma", "L2da", "L3ra", "L5ta", "L6ta", "L7ma",
"S3ra", "S5ta", "S7ma")
for (b in bases) {
  comando <- paste0("id_b <- intersect(", b, "$inec_identificador_empresa, reporte$inec_identificador_empresa)")
  eval(parse(text = comando), envir = .GlobalEnv)

  for (fil in 1:length(id_b)) {
    comando <- paste0("var_list_justif <- as.character(apply(", b, "[", fil, ", ], 1, function(x) {names(x[which(!is.na(x))])}))")
    eval(parse(text = comando), envir = .GlobalEnv)

    var_list_justif <- var_list_justif[-1]

    var_list_reales <- var_list_justif[which(var_list_justif %in% names(reporte))]

    for (var in var_list_reales) {
      BorrarCeldas("reporte", id_b[fil], var)
    }
    print(paste0("Barridas todas las variables de ", b, "[", fil, ", ]"))
  }
  print(paste0("Completada toda la base ", b))
}
rm(b, bases, id_b, fil, var_list_justif, var_list_reales, var)
rm(comando)
```

I. Determinación de umbrales superiores de variación interanual por variable de control.

```{r, warning=FALSE}
```

```
for (v in var_chk) {
  # comando <- paste0(sub("v", "q", v), " <- quantile(E$", v, ", seq(0,1,0.01), na.rm=T)")
  comando <- paste0(sub("v", "q", v), " <- quantile(reporte$", v, ", seq(0,1,0.01), na.rm=T)")
  eval(parse(text = comando), envir = .GlobalEnv)
}

lista_q <- sub("v", "q", var_chk)
# Q es el frame con los centiles de todas las 53 variables a chequear. Sirve para determinar
# visualmente los umbrales de variación máximos permitidos para esas variables.
Q <- data.frame(matrix(NA_real_, nrow = 101, ncol=0))
for (v in lista_q) {
  comando <- paste0("Q <- cbind(Q, ", v, ")")
  eval(parse(text = comando), envir = .GlobalEnv)
}
rm(list = lista_q)
rm(comando, lista_q, v)
```

J. Publicación en Excel del frame de reporte "info3", con el frame "informe2".

```{r}
# El vector "umbrales" se obtiene después de analizar, columna por columna, al frame Q generado en el bloque anterior.
umbral_inf <- c(-0.9,  -0.9,         -0.9,         -0.99, -0.99,              -0.999,      -0.64, -0.83,              -0.99, -0.92,              -0.92, -
0.98, -0.98, -0.98, -0.95, -0.995, -0.99, -0.999,           -0.5, -0.99, -0.99,      -0.9,         -0.9,         -0.9,         -0.9,         -0.999,
                -0.999,      -0.99, -0.999, -0.999, 0, -0.99, -0.4, -0.5, -0.5, 0, 0, 0, 0, 0, 0, 0,   0, 0, 0, 0, 0, -0.999, -0.99, -0.999,
                -0.99, -0.99, -0.99)

umbral_sup <- c(1.6, 1.8, 1.4, 1.61, 7.55, 10.42, 3.1, 0.92, 2.1, 2.2, 2.1, 2.2, 2, 3, 0.17, 1.17, 3, 3.5, 1.5, 3.9, 3.7, 1, 2.4, 1, 1, 1.9,
2.4, 2.4, 3.3, 3, 0, 3, 0.9, 1.5, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2.6, 2, 2, 2, 2, 2)

names(umbral_inf) <- names(umbral_inf) <- var_chk
Err_x_EMP <- apply(reporte[1:dim(reporte)[1], var_chk], 1, function(x) sum(as.numeric(x < min(-0.35, umbral_inf) | x > max(0.35,
umbral_sup)), na.rm=T))
reporte$Err_x_EMP <- Err_x_EMP
```

K. Publicación en Excel del frame de reporte "info3", con el frame "informe2".

```{r}
writexl::write_xlsx(reporte, "C:/Users/rbenavides/Documents/EMPRESAS 2022/7mo Corte- 06-11-2023/VAL
COMPARACIONES/VAL_COMP_7mo_Corte_06_11_2023.xlsx")
```

ANEXO E. Código R Markdown para la verificación de valores extremos de las variables de escala.

```
---
title: "SX Valores extremos 2022"
author: "Ramiro Benavides"
date: "22/11/2021"
update: "30/12/2024"
output: html_document
---
```

1. Carga de la BDD.

```
```{r}
EMP_2022 <- haven::read_sav("C:/Users/Ramiro/Documents/EMPRESAS 2022/10mo Corte - 18-12
-2023/BDD/Empresasfusionado.sav")

B <- EMP_2022[which(EMP_2022$estado == "C" & EMP_2022$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 2022/10mo Corte - 18-12-2023/Reporte_Extremos_2022_12_18.xlsx")
```

Elaborado por	Ramiro Benavides	Firmado electrónicamente por: RAMIRO VICENTE BENAVIDES LEON
Revisado por	Carlos Pilataxi	Firmado electrónicamente por: CARLOS PATRICIO PILATAXI GUAIRACAJA
Aprobado por	Armando Salazar	Firmado electrónicamente por: ARMANDO DAVID SALAZAR MENDEZ


INEC
Buenas cifras,
mejores vidas
www.ecuadorencifras.gob.ec
@ecuadorencifras
@ecuadorencifras
@InecEcuador
INECEcuador