Práctica 2 - Proceso de dosificación y control de peso medio en envasado.

Juan Riera

17/9/2021

## Introducción

Uo de los aspectos en los que y los métodos estadísticos pueden ayudarnos es en la determinación de los **pesos nominales** de un producto envasado.

Cuando compramos un producto, podemos hacerlo de dos formas:

* a **peso variable** : cuando adquirimos una unidad o parte de ella y el peso no ha sido declarado por el fabricante. En este caso, es el comerciante quien debe pesar la cantidad entregada para poder determinar el precio de venta. Es el caso de los *lineales de corte* de queso o de charcutería, en los que pedimos al charcutero la cantidad que deseamos y él o ella pesan la cantidad realmente servida, marcándola con una etiqueta para que en la caja nos digan el precio.
* a **peso fijo**: cuando la unidad completa y cerrada tiene su peso establecido y etiquetado por el fabricante. Es el caso de las compras en el *lineal de libre servicio*, en donde no hay una dispensación de la cantidad, sino que los envases están cerrados y etiquetados. El peso declarado por el fabricante se conoce como **peso nominal**; lo que el envase contiene realmente se conoce como **peso real** o **contenido efectivo**.

Las etiquetas de peso fijo permiten a la distribución comercial identificar el dódigo de producto y su peso nominal correspondiente mediante el [código de barras EAN 13](https://es.wikipedia.org/wiki/European_Article_Number) que el fabricante imprime en el envase. Este sistema es mucho más cómodo para el comercio porque no tiene que destinar personas a pesar y empaquetar el producto, y la lectura en la línea de cajas es más rápida; todo ello contribuye a mejorar la productividad y rentabilidad del comercio.

En el caso de los productos de peso fijo, un aspecto muy importante a controlar es el rigor del fabricante al determinar el peso, para evitar los fraudes al consumidor. El responsable de supervisar y controlar que los fabricantes etiquetan correctamente el peso de sus productos es el Ministerio de Consumo, a través de una ley que se conoce como [Ley de contenido efectivo](https://www.boe.es/buscar/pdf/2008/BOE-A-2008-17629-consolidado.pdf). Los inspectores del Ministerio realizan inspecciones al azar revisando tanto los productos en el punto de venta como las plantas de envasado, para asegurarse de que el fabricante cumple con la legislación.

Hay dos modalidades para que un fabricante pueda declarar el peso de sus productos:

* El fabricante puede ajustarse a la ley de contenido efectivo: en este caso, el fabricante debe realizar controles estadísticos de peso de cada lote envasado, para demostrar que cumple con lo especificado en la ley. Al hacerlo así, la ley le permite indicar en su etiquetado que está realizando este control estadístico, mediante la impresión de una letra **e** en una zona de su etiqueta. Esta letra **e** es conocida como [signo de estimación](https://es.m.wikipedia.org/wiki/Signo_de_estimaci%C3%B3n).
* Si el fabricante no desea hacer un control estadístico de peso, debe garantizar que todos y cada uno de sus envases cumplen con el contenido nominal declarado. Ee este caso no puede mostrar la letra **e** en su etiquetado.

Como sabemos muy bien, cuando hacemos una serie de medidas de peso, estas medidas tienen una distribución de frecuencias que cuando están en número mayor de se puede aproximar a una [distribución normal](https://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal) o [campana de Gauss](https://es.wikipedia.org/wiki/Funci%C3%B3n_gaussiana). En nuestro dcumento inicial, hemos visto que una distribución normal puede definirse por sus parámetros media y desviación típica , y que según la desviación típica, podemos esperar una determinada cantidad de valores por encima o por debajo de la media.

Esto quiere decir que en un producto envasado, en el cual el fabricante declara un **peso nominal**, el peso real de un envase concreto puede no ser exactamente el peso nominal etiquetado, sino otro que será función de la dispersión de peso del proceso de envasado.

Cuando un fabricante elige utilizar un método de control estadístico del peso envasado, de acuerdo con la ley de contenido efectivo, la propia ley establece los valores máximos de dispersión que el fabricante debe tener; lo veremos a continuación.

Es evidente que, puesto que existe dispersión alrededor del peso medio, el cumplimiento de la ley obliga en la mayor parte de los casos a dosificar un peso ligeramente superior al peso nominal. La diferencia entre el peso real y el nominal es lo que conocemos como **sobrepeso**.

En resumen,

* Cuando se realiza el control estadístico de peso, y se imprime la **e** en la etiqueta, estamos diciendo al consumidor que es posible que alguno de los envases *tenga un peso inferior al nominal*, pero el número de envases que podemos llamar *deficientes* se ajusta a lo especificado por la ley, y garantizamos que el *peso medio del lote* es igual o superior al peso nominal.
* Cuando no se realiza un control estadístico del peso envasado, no podemos imprimir la letra **e**; en este caso, como hemos visto, *todos* los envases deben tener un peso igual o superior al peso nominal.

## La ley de contenido efectivo.

Tal como indicábamos más arriba, los productos envasados están sujetos a una normativa legal específica en cuanto a su contenido, que es la [Ley de contenido efectivo](https://www.boe.es/buscar/pdf/2008/BOE-A-2008-17629-consolidado.pdf). Vamos a ver sus aspectos más importantes a continuación.

En su capítulo 3, la ley proporciona las definiciones siguientes:

1. **Envase**: Unidad formada por el envase propiamente dicho y su contenido, de forma que la cantidad de producto que contiene no pueda variarse sin que el envase propiamente dicho sufra una apertura o modificación perceptible.
2. **Cantidad nominal** (masa nominal o volumen nominal): Es la masa o volumen de producto marcado en el etiquetado del envase; es decir, la cantidad de producto que se estima debe contener el envase.
3. **Contenido efectivo**: Es la cantidad (masa o volumen) de producto que contiene realmente el envase. Cuando se exprese en unidades de volumen, se entenderá referido a la temperatura de 20 ºC, con exclusión de los productos congelados y ultracongelados.
4. El **error por defecto** en un envase es la diferencia en menos del contenido efectivo respecto a la cantidad nominal.
5. El **error máximo por defecto tolerado** en un envase es la cantidad máxima que puede diferir en menos de la cantidad nominal.
6. El **contenido mínimo tolerado** en un envase es el obtenido restando a la cantidad nominal del envase el error máximo por defecto tolerado.
7. **Envase deficiente** es aquel cuyo contenido efectivo es inferior al contenido mínimo tolerado.
8. **Lote**: conjunto de envases de iguales cantidades nominales, modelo y fabricación, llenados en el mismo lugar y que son objeto del control.

Como es lógico, la ley no indica en ningún momento que el hecho de que un envase contenga *más* peso que el declarado en la **cantidad nominal** sea un defecto, sólo se considera *envase deficiente* el que contiene **menos** peso de una cantidad determinada, por debajo de la cantidad nominal.

El exceso de peso no es un problema legal, sino económico: estamos *regalando* producto por encima del peso nominal que no vamos a cobrar, pero que tiene un coste de producción para la empresa; en cambio, entregar menos peso de lo permitido en la ley constituye un **fraude**

Por esta razón, la ley se ocupa de definir con precisión la *falta de peso*, y define lo que es un **envase deficiente** como aquellos envases que tienen un contenido inferior al **contenido mínimo tolerado**. Más adelante especifica cuál es el **error máximo por defecto tolerado** (haremos referencia a esta cantidad como **EMDT**), que depende del peso o volumen de la unidad envasada y del peso nominal, y que quiere decir *la máxima falta de peso que puede admitirse en un envase sin incurrir en un fraude*.

La siguiente tabla nos indica cuál es el **EMDT** para envases de diferentes pesos o volúmenes nominales:

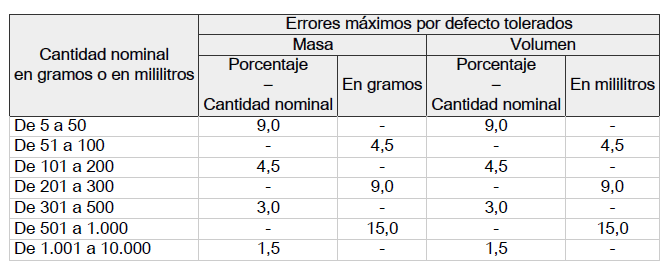


tabla con los EMDT

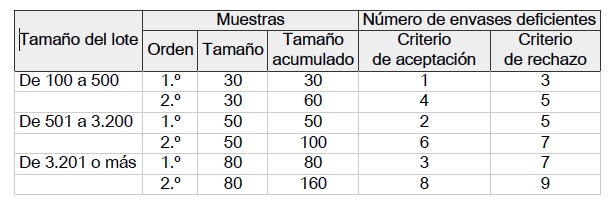
Nuestros envases de mermelada tienen un peso nominal, declarado por el fabricante, de 250 gramos. Esto quiere decir que el **EMDT** es de 9 gramos.

Según esto, podemos calcular que la norma considera **envases deficientes** (ver punto (g) anterior) los que contengan *menos* de , que es el *contenido mínimo tolerado* que se indica en el punto (f).

En el capítulo 7, la norma completa las restricciones:

* El peso medio del lote no debe ser inferior a la cantidad nominal
* El número de envases deficientes en el lote debe cumplir las exigencias del decreto (veremos a continuación cuáles son)
* Ningún envase del lote debe tener un contenido inferior al resultado de restar al peso nominal el doble del **EMDT**

Finalmente, la norma define el tamaño de muestra necesario para hacer el control según el tamaño del lote, y los criterios de aceptación o rechazo.



Criterios de rechazo

Para facilitar el control, la norma establece un muestreo en dos etapas: en una primera etapa tomamos una muestra que es la mitad de la muestra máxima, y verificamos el resultado. Si los criterios de aceptación no se cumplen, tomamos una segunda muestra y acumulamos los valores de ambas para determinar la aceptación o rechazo.

En nuestro caso, consideramos **lote** la producción realizada en un día, que es de unidades. Para este tamaño de lote, la norma establece que cada etapa de muestreo debe ser de unidades. Nuestro **EMDT** es gramos, luego:

* envase deficiente es aquél que tiene un peso *inferior* a
* no pueden aparecer envases con peso *inferior* a

El procedimiento que seguimos es el siguiente:

* **Rechazamos** el lote si en la primera muestra nos aparecen:
  + o más envases con un peso inferior a , **o**
  + o más envases con un peso inferior a (envases *deficientes*)
* **Aceptamos** el lote si en la primera muestra de unidades nos aparecen:
  + ningún envase con un peso inferior al , **y**
  + o menos envases con un peso inferior a (envases *deficientes*)

Si en la primera muestra nos aparecen más de (criterio de aceptación) y menos de envases deficientes, tomamos una segunda muestra de unidades, y acumulamos los datos con la primera.

* **Rechazamos** el lote si en el total ( unidades) nos aparecen:
  + o más envases con un peso inferior a , **o**
  + o más envases con un peso inferior a (envases *deficientes*)
* **Aceptamos** el lote si en en el toal de las dos muestras ( unidades) nos aparecen:
  + ningún envase con un peso inferior a , **y**
  + o menos envases con un peso inferior a (envases *deficientes*)

Es importante insistir en que **rechazamos** el lote si se cumple *cualquiera de las dos condiciones* de rechazo; y que para **aceptar** el lote, deben darse **las dos** condiciones de aceptación.

## Presentación del caso

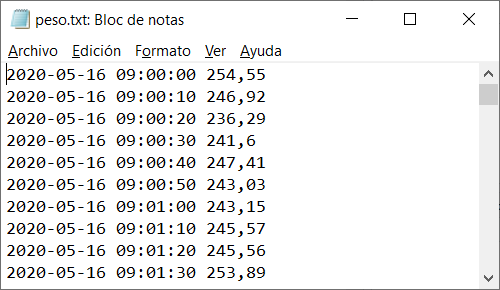
Un fabricante de conservas de la región ha instalado una nueva línea de envasado, para producir una mermelada de manzana natural en formato de 250 gramos. Quiere servir a la distribución comercial declarando peso fijo. Ha solicitado al centro de FP Juan de Villanueva, de Pola de Siero, un estudiante o una estudiante en prácticas para ayudar en el proceso de puesta en marcha, y quiere que esta persona lo ayude a conseguir una dosificación óptima. A partir de los ensayos que ha hecho, el fabricante cree que es capaz de dosificar exactamente 250 gramos, y quiere confirmar que el proceso es estable; ha guardado algunos datos de los primeros envasados en una hoja de cálculo. También quiere saber si el etiquetado debe llevar la “**e**” o no, porque no entiende muy bien su significado y las implicaciones para su empresa.

## Preparación de los datos: importando un nuevo formato de fichero de texto

Como siempre, el primer paso es leer los datos que nos han proporcionado y cargarlos en un *dataframe* de . Hasta ahora hemos visto cómo leer datos de hojas Excel mediante la función read\_excel()de la librería readxl (que forma parte de lo que se conoce como [universo tidyverse](https://www.tidyverse.org/)) y también cómo leer archivos en formato mediante la función read.csv2(), que forma parte de las funciones básicas de .

Estos formatos son perfectos para nosotros, porque la primera línea tiene los nombres de las variables, y los valores vienen delimitados, y en el caso de Excel, directamente nos proporcionan las correcciones de formato, fecha, etc.

No siempre es así, los sistemas de lectura automática, en muchas ocasiones entregan *ficheros planos* de texto, probablemente sin nombres de variables, y sólo como listas de valores. Supongamos que el fichero de datos de peso que nos han entregado tiene este formato de texto plano; en el *bloc de notas* de Windows se podría ver así:



peso.txt visualizado en bloc de notas de windows

Esto no representa ningún inconveniente para nosotros, sólo nos da algo más de trabajo para configurar correctamente nuestro *dataframe*. Utilizamos una nueva función para leer este fichero, que lee tablas de texto:

df <- read.table("peso.txt")  
head(df)

## V1 V2 V3  
## 1 2020-05-16 09:00:00 254,55  
## 2 2020-05-16 09:00:10 246,92  
## 3 2020-05-16 09:00:20 236,29  
## 4 2020-05-16 09:00:30 241,6  
## 5 2020-05-16 09:00:40 247,41  
## 6 2020-05-16 09:00:50 243,03

Como el fichero de datos no tiene nombres de variables, añade nombres a las columnas de forma automática, designándolas como V1, V2, V3…, es decir, variable 1, variable 2, variable 3…

Verificamos la estructura de la tabla:

str(df)

## 'data.frame': 3000 obs. of 3 variables:  
## $ V1: chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2: chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3: chr "254,55" "246,92" "236,29" "241,6" ...

Aquí podemos ver que *todas las columnas han sido leídas como variables character*. En el caso de las dos primeras, parece normal porque R no sabe interpretar una variable de texto como fecha o como hora si no le especificamos el formato, lo hemos visto en la práctica anterior.

Pero ¿por qué no interpreta los valores de peso como números, sino como texto? La explicación tiene truco: el sistema de control de peso ha impreso los pesos con una coma decimal, tal como se usa en España y en Europa, pero , que es un programa del entorno anglosajón, separa los decimales con un punto. Por eso interpreta que tres dígitos, una coma y dos dígitos no son un número sino una cadena de texto.

La opción más fácil para corregir esto es volver a leer la tabla con utilizando una opción de la función que indica que los decimales están separados por una coma:

df <- read.table("peso.txt", dec = ",")  
head(df)

## V1 V2 V3  
## 1 2020-05-16 09:00:00 254.55  
## 2 2020-05-16 09:00:10 246.92  
## 3 2020-05-16 09:00:20 236.29  
## 4 2020-05-16 09:00:30 241.60  
## 5 2020-05-16 09:00:40 247.41  
## 6 2020-05-16 09:00:50 243.03

La estructura de la tabla nos dice que ahora la columna se ha leído correctamente como número:

str(df)

## 'data.frame': 3000 obs. of 3 variables:  
## $ V1: chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2: chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3: num 255 247 236 242 247 ...

Hay otras opciones posibles, como convertir la cadena de números con coma a números con punto usando una función de que substituye un carácter por otro, y luego convertir esta cadena alfanumérica a número.Tenemos que hacerlo en dos etapas:

* primero, convertimos la cadena alfanumérica con coma a una cadena alfanumérica con punto
* finalmente, convertimos la cadena alfanumérica con punto a número decimal

La secuencia sería ésta:

df <- read.table("peso.txt") # leemos la tabla sin corregir formato  
  
df$V3 <- sub(",", ".", df$V3) # sustituimos las comas por puntos  
str(df)

## 'data.frame': 3000 obs. of 3 variables:  
## $ V1: chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2: chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3: chr "254.55" "246.92" "236.29" "241.6" ...

sigue siendo una variable *character*, pero hemos sustituido las comas por puntos. Ya podemos convertir la columna a numérica:

df$V3 <- as.numeric(df$V3) # convertimos la columna a numérica  
str(df)

## 'data.frame': 3000 obs. of 3 variables:  
## $ V1: chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2: chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3: num 255 247 236 242 247 ...

Otra opción es utilizar la [función parse\_number()](https://readr.tidyverse.org/reference/parse_number.html) de la biblioteca readr, que permite hacer múltiples conversiones de formato; no lo detallaremos aquí, pero es muy útil conocerla.

Con cualquiera de las opciones indicadas, ahora nuestros datos tienen la estructura correcta y ya es una variable numérica que podremos analizar.

Necesitamos dos acciones adicionales:

* convertir las dos columnas de fecha y hora, que tienen formato *character*, en una única columna con formato *datetime*, que nos permite todas las opciones de gráficos y agrupaciones de fechas, y
* renombrar las columnas

Creamos una columna de fecha pegando las dos columnas de fecha y hora que tenemos en formato texto mediante la [función paste()](https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/paste):

df$fecha <- paste(df$V1, "", df$V2)  
str(df)

## 'data.frame': 3000 obs. of 4 variables:  
## $ V1 : chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2 : chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3 : num 255 247 236 242 247 ...  
## $ fecha: chr "2020-05-16 09:00:00" "2020-05-16 09:00:10" "2020-05-16 09:00:20" "2020-05-16 09:00:30" ...

Ya tenemos nuestra columna de en formato *character*, y esto sabemos convertirlo a *datetime* usando la misma función que hemos usado otras veces:

df$fecha <- as.POSIXct(df$fecha, tz = "GMT", format = "%Y-%m-%d %H:%M")  
str(df)

## 'data.frame': 3000 obs. of 4 variables:  
## $ V1 : chr "2020-05-16" "2020-05-16" "2020-05-16" "2020-05-16" ...  
## $ V2 : chr "09:00:00" "09:00:10" "09:00:20" "09:00:30" ...  
## $ V3 : num 255 247 236 242 247 ...  
## $ fecha: POSIXct, format: "2020-05-16 09:00:00" "2020-05-16 09:00:00" ...

Ahora sólo nos queda eliminar las columnas y , que ya no necesitamos, y corregir el nombre de la columna . La forma más sencilla, y probablemente también la más legible, es usar la función selectde la librería dplyr de tidyverse; decimos a que seleccione el *dataframe* *menos* las dos columnas indicadas por la función vector c(), y que copie el resultado en el propio *dataframe* :

library(tidyverse)  
df <- select(df, -c(V1,V2))

A continuación, renombramos la columna con la [función rename()](https://dplyr.tidyverse.org/reference/rename.html) de la biblioteca dplyr:

df <- rename(df, peso = V3)  
head(df)

## peso fecha  
## 1 254.55 2020-05-16 09:00:00  
## 2 246.92 2020-05-16 09:00:00  
## 3 236.29 2020-05-16 09:00:00  
## 4 241.60 2020-05-16 09:00:00  
## 5 247.41 2020-05-16 09:00:00  
## 6 243.03 2020-05-16 09:00:00

Como vemos, en muchas ocasiones la fase de preparación de los datos puede llegar a ser bastante larga e incluso engorrosa y tediosa; la ventaja de es que una vez que hemos hecho correctamente todos los pasos, nos quedan guardados en el *script*; en sucesivas ocasiones, el proceso será automático y nos llevará unos pocos segundos. Además, como tenemos a la vista cada instrucción, podemos hacer cualquier cambio que necesitemos sin interrumpir el flujo de proceso.

Si agrupamos todas las instrucciones que hemos ejecutado, serían éstas:

df <- read.table("peso.txt", dec = ",")  
df$fecha <- paste(df$V1, "", df$V2)  
df$fecha <- as.POSIXct(df$fecha, tz = "GMT", format = "%Y-%m-%d %H:%M")  
df <- select(df, -c(V1,V2))  
df <- rename(df, peso = V3)  
head(df)

## peso fecha  
## 1 254.55 2020-05-16 09:00:00  
## 2 246.92 2020-05-16 09:00:00  
## 3 236.29 2020-05-16 09:00:00  
## 4 241.60 2020-05-16 09:00:00  
## 5 247.41 2020-05-16 09:00:00  
## 6 243.03 2020-05-16 09:00:00

Podemos reescribir con el [estilo tidyverse](https://style.tidyverse.org/pipes.html) mediante [pipes](https://www.datacamp.com/community/tutorials/pipe-r-tutorial) (ver [nota de formación en Datacamp](https://www.datacamp.com/community/tutorials/pipe-r-tutorial) sobre pipes), que son signos que se utilizan en R y otros lenguajes de programación para indicar que la salida de una función se redirige directamente a la función siguiente. Esto reduce la cantidad de código que escribimos, haciendo más claro el flujo. El signo pipeoriginal utilizado por tidyverse es un “mayor que” entre dos signos de porcentaje %>%. Aunque este signo sigue siendo válido y se sigue utilizando. ahora se prefiere el nuevo signo |> que forma parte de de forma nativa desde la versión 4.1.0, y que es igual al utilizado en otros lenguajes de programación como [F#](https://docs.microsoft.com/es-es/dotnet/fsharp/language-reference/symbol-and-operator-reference/) y [Julia](https://julialang.org/).

df <- read.table("peso.txt", dec = ",") |>  
 mutate (fecha = paste(V1, "", V2)) |>  
 mutate (fecha = as.POSIXct(fecha, tz = "GMT", format = "%Y-%m-%d %H:%M")) |>  
 select(-c(V1,V2)) |>  
 rename(peso = V3)  
  
head(df)

## peso fecha  
## 1 254.55 2020-05-16 09:00:00  
## 2 246.92 2020-05-16 09:00:00  
## 3 236.29 2020-05-16 09:00:00  
## 4 241.60 2020-05-16 09:00:00  
## 5 247.41 2020-05-16 09:00:00  
## 6 243.03 2020-05-16 09:00:00

En estas líneas hemos introducido también la [función mutate()](https://dplyr.tidyverse.org/reference/mutate.html) de la librería dplyr. Esta función se utiliza para crear o modificar columnas, y encuentra su uso más frecuente dentro de una secuencia de *pipes* en donde no es posible utilizar el operador de asignación <- de .

Revisemos los pasos que hemos dado para la importación de los datos:

* leemos el fichero de texto,
* pegamos las dos columnas con la fecha y la hora en una nueva columna de texto,
* convertimos esta columna a formato fecha,
* eliminamos las dos columnas que ya no necesitamos,
* renombramos la columna con el peso, y
* finalmente, asignamos el resultado a nuestro *dataframe*.

Apenas cinco líneas de código de csript con una sintaxis realmente clara de lo que hace cada instrucción.

Con esto hemos acabado el trabajo de preparación de los datos y podemos pasar a la fase de análisis.

## Análisis de los datos de peso

Como siempre, lo primero que hacemos es un resumen de los datos:

summary(df)

## peso fecha   
## Min. :223.6 Min. :2020-05-16 09:00:00   
## 1st Qu.:244.5 1st Qu.:2020-05-16 11:04:45   
## Median :250.1 Median :2020-05-16 13:09:30   
## Mean :250.0 Mean :2020-05-16 13:09:30   
## 3rd Qu.:255.5 3rd Qu.:2020-05-16 15:14:15   
## Max. :279.4 Max. :2020-05-16 17:19:00

El peso medio es exactamente de , con un mínimo de y un máximo de .

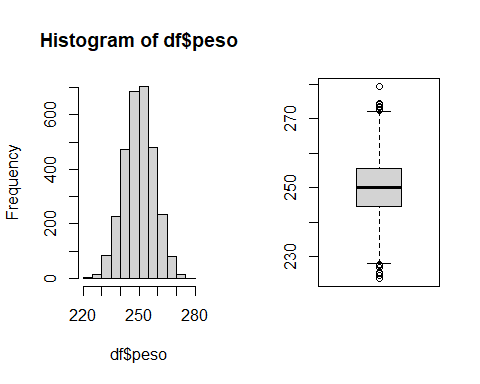
La desviación típica es:

sd(df$peso)

## [1] 8.147409

Verificamos la distribución mediante el histograma y el diagrama de caja:

par(mfrow=c(1,2))  
hist(df$peso)  
boxplot(df$peso)



par(mfrow=c(1,1))

Los datos son aproximadamente simétricos y el aspecto del histograma es el de la distribucón normal, lo cual nos indica que la utilización de la media y la desviación típica como descriptores de la distribución parece correcta. tiene métodos precisos para hacer estas comprobaciones, que podremos ver en detalle en otro momento.

## Comprobación de la adecuación de nuestros datos de peso a la normativa legal

no tiene una función específica para contar el número de elementos de un vector inferiores a un valor, pero podemos usar una alternativa muy sencilla, mediante el uso de las constantes lógicasTRUE o FALSE, que se pueden evaluar como valores o respectivmente.

Lo explicaremos con más detalle. Imaginemos un vector formado por varias cantidades:

v <- c(11, 3, 5, 15, 23, 7, 13)

Si hacemos una pregunta lógica, por ejemplo si los elementos del vector son inferiores a un valor determinado, nos responde con unn vector de TRUEy FALSE resultado de comparar cada elemento con el valor de prueba:

v < 10

## [1] FALSE TRUE TRUE FALSE FALSE TRUE FALSE

Si queremos saber cuántos valores son inferiores al valor de control, no tenemos más que sumar este resultado, ya que los FALSEse calcularán como y sólo se sumarán los TRUE que se computan como :

sum(v<10)

## [1] 3

Lo que es correcto: los valores del vector inferiores a son tres: , y

Podemos utilizar esta forma de calcular para estimar los valores de peso en nuestro *dataframe* por debajo del peso de rechazo ( g), y el número de envases deficientes, que son los inferiores a gramos.

La norma nos indica las cantidades en dos muestras sucesivas de unidades, haciendo un total de 100. Podemos ponernos en el caso más desfavorable y calcular una muestra de unidades a partir de nuestras observaciones de peso.

Para calcular una muestra, podemos utilizar la función sample(), que extrae una muestra aleatoria de un vector; indicamos el vector y el tamaño de la muestra:

muestra <- sample(df$peso, 100)

Una vez que hemos importado los datos y hecho nuestra muestra, el cálculo de los criterios de rechazo y la comparación con la norma es ya muy sencillo:

sum(muestra < 232)

## [1] 1

sum(muestra < 241)

## [1] 10

Nuestra muestra tiene 1 elementos por debajo del criterio de rechazo del valor nominal menos dos veces el EMDT (232). Por lo tanto, según este criterio, **rechazamos** el lote como válido.

Además tiene 10 envases deficientes, lo que es superior o igual al criterio de rechazo de 7envases deficientes; por lo tanto, según este criterio, **rechazamos** el lote como válido.

## El trabajo de mejora

Hasta aquí hemos conseguido determinar la validez de un lote de producto envasado de acuerdo con la ley de contenido efectivo. Si nos quedásemos aquí, habríamos hecho razonablemente bien el papel de inspector o inspectora del Ministerio de Consumo, pero de nosotros se espera algo más. Si el lote no es válido, ¿qué podemos hacer para que en el futuro nos ajustemos a la ley? ¿Cuál es el sobrepeso que estamos entregando y cuánto cuesta a la empresa? ¿Podemos conseguir algún ahorro en el proceso de dosificación manteniendo el proceso dentro de las especificaciones legales? ¿Cómo estableceríamos las etapas básicas de la mejora? Este es el papel que la empresa espera que desempeñemos, ¡no el de un mero fiscalizador!

Para entrar en esta parte apasionante de nuestro trabajo, neesitamos repasar algunos conceptos y descubrir algunas nuevas herramientas, ¡no muchas!

## Cómo determinamos los límites de tolerancia de un proceso

Hemos visto que, en determinadas condiciones, una distribución de frecuencias puede ser descrita de forma aproximada mediante una distribución normal, que utiliza como parámetros y la media y la [desviación típica](https://es.wikipedia.org/wiki/Desviaci%C3%B3n_t%C3%ADpica) de la distribución. La *aproximación normal* nos permite usar una propiedad de la *curva normal*, que ya hemos visto: el porcentaje de casos que esperamos encontrar en los intervalos definidos por la desviación típica:

Fuente: Wikipedia

Recordemos que, en una distribución concreta,

* en el intervalo encontraremos el de los datos
* en el intervalo encontraremos el de los datos
* en el intervalo encontraremos el de los datos

La magnitud de la desviación típica nos define la amplitud de la dispersión, es decir, el aspecto más o menos aplanado de la distribución:

Fuente: Wikipedia

La gráfica anterior nos representa dos procesos con valor medio y con dos valores distintos de desviación típica. La distribución representada en rojo tiene una , mientras que la distrobución representada en azul tiene una

Si ambas gráficas representasen dos procesos de llenado de envases de 100 gramos, el primer proceso tendría el 99,6% de los casos en el intervalo . Sin embargo, el segundo proceso tendría en este intervalo solamente en torno al 70% (aproximadamente) de los datos, ya que el intervalo de

Supongamos ahora que la especificación de calidad del proceso de envasado tuviese como límite inferior de peso los 50 gramos. En el primer caso, estaríamos prácticamente seguros de que ninguno de nuestros envases estaría por debajo de ese peso, y por tanto cumpliríamos la especificación sin ningún problema. En el segundo caso, en cambio, es el límite inferior de *una* desviación típica, por lo que, según lo que vemos en nuestra curva normal, el de los casos estarían por debajo de esta cantidad, es decir, tendríamos un fuera de la especificación.

Esto tiene dos lecturas diferentes:

* Para construir los límites de nuestro proceso, necesitamos conocer su desviación típica, con ella podemos construir los intervalos dentro de los cuales debe mantenerse el proceso.
* Si nuestro proceso debe ajustarse a una especificación externa, ya sea una especificación del cliente o una normativa legal, podremos hacerlo solamente si la desviación típica se ajusta a lo que se nos exige.

En un proceso de envasado, no hay límite legal superior al peso que entregamos, sólo límite inferior. Por eso, cuando no somos capaces de reducir nuestra variabiidad, nos vemos obligados a subir el peso dosificado para que el límite inferior esté dentro de la especificación.

### Introducción a la capacidad de un proceso

Cuando un proceso se ajusta a los límites especificados, se dice que es un **proceso capaz**. Si nuestro proceso **no es capaz**, nuestro trabajo consiste en reducir la variabilidad hasta que el proceso *entre* en la especificación. La reducción de la variabilidad es uno de los principales objetivos de los procesos **Seis Sigma**.

Existen medidas concretas para definir la [capacidad del proceso](https://es.wikipedia.org/wiki/Capacidad_del_proceso), que pueden revisarse en la bibliografía; no las veremos aquí por el momento.

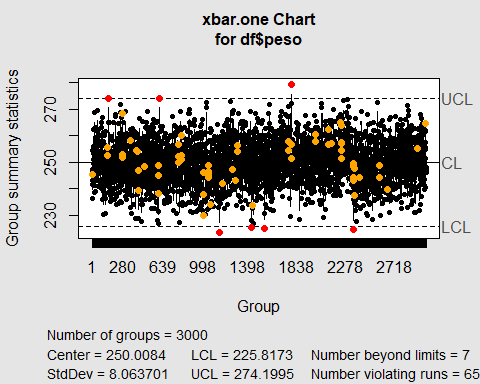
proporciona gráficos específicos para analizar la capacidad de un proceso, y para hacer el seguimiento de un proceso y analizar sus desviaciones.

Vamos a analizar la primera de ellas, estudiando la capacidad de nuestro proceso de envasado de acuerdo con la normativa legal.

Utilizaremos dos funciónes de la [librería qcc](https://cran.r-project.org/web/packages/qcc/vignettes/qcc_a_quick_tour.html), la propia función qcc() y la función process.capability()

La primera función nos calcula un gráfico de control, como hemos visto en la introducción. En este caso analizaremos el proceso completo de envasado que hemos realizado en el día, y que consta de 3.000 envases. El primer gráfico que construimos representa todos los valores individuales, para ello utilizamos la opción type="xbar.one":

library(qcc)  
qcc(df$peso, type = "xbar.one")



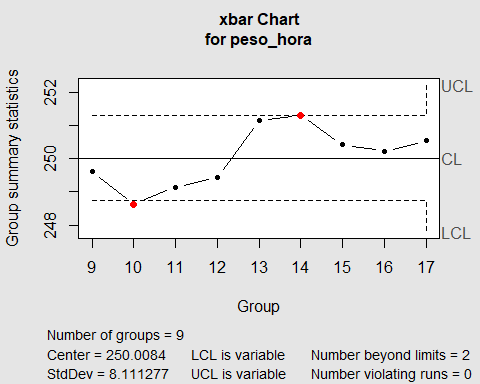
Aunque el gráfico que obtenemos es poco legible porque hay demasiados datos y está un poco emborronado, el resultado nos proporciona una primera información útil: el análisis considera que nuestro proceso tiene un límite inferior de (el superior es de ), y hemos tenido valores fuera de estos límites. La cantidad tan grande de valores no nos permite ver mucho más (ver concepto de [*overplotting*](https://www.displayr.com/what-is-overplotting/))

Este límite inferior es un problema según la normativa legal; si recordamos lo que hemos visto antes, no deberíamos tener ningún valor por debajo de .

Vamos a hacer un nuevo gráfico de control que nos simplifique la información de nuestro proceso, agrupando los pesos por horas. Asignaremos a las 9h todos los pesos que hemos obtenido entre las 9:00 y las 10:00, y así sucesivamente. Esto nos permite calcular la variabilidad que hemos tenido cada hora, y compararla con la variabilidad total.

Para ello agruparemos los valores por horas mediante la función qcc.groups() y utilizaremos la opción type="xbar"de la función qcc(), para hacer un gráfico de las medias horarias previamente agrupadas. Guardaremos el resultado en un objeto qcc, asignando la salida de la función a una variable que llamaremos :

library(qcc)  
peso\_hora <- qcc.groups (df$peso, hour(df$fecha))  
peso\_grafico <- qcc(peso\_hora, type = "xbar", nsigmas = 3)



El gráfico de control nos muestra de forma mucho más clara lo que ha sucedido en nuestro proceso de envasado. Desde el inicio del envasado a las 9:00 hasta aproximadamente las 13:00 hemos estado envasando por debajo del peso nominal (recordemos que el envasado entre las 12:00 y las 13:00 se representa dentro de las 12:00). Además, el peso medio en el periodo de 10:00 a 11:00 ha estado fuera del límite de control inferior, y en el de 14:00 a 15:00 ha estado fuera del límite de control superior.

Comprobemos el peso medio den el periodo de 9:00 a 13:00, mediante el uso básico de los criterios de selección de los vectores de :

mean(df$peso[hour(df$fecha) < 13])

## [1] 249.2016

El peso medio durante este tiempo ha sido de , y esto no cumple la exigencia legal de que la media del peso envasado sea superior al peso nominal. Posiblemente, el responsable de envasado se dio cuenta de esto y decidió subir el peso durante las dos hora siguientes para compensar. Comprobamos el peso de la última parte del envasado, esta vez usando las funciones y estilo de biblioteca tidyverse, que resulta mucho más clara:

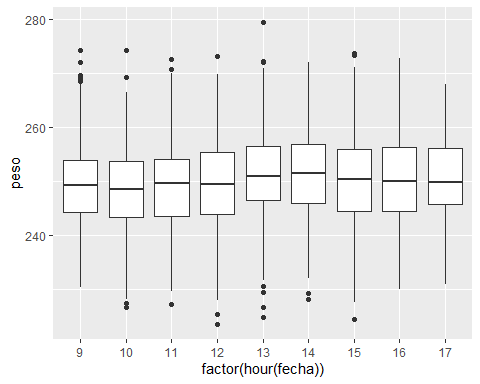
df |>  
 filter(hour(fecha)>= 13) |>  
 summarize( casos = n(), peso\_medio = mean(peso))

## casos peso\_medio  
## 1 1560 250.7532

En la segunda parte del envasado se han envasado 1560 envases con un peso medio de de 250,75 gramos. Aunque esta parte del envasado también se ha desviado del objetivo de peso nominal, indirectamente tiene una lectura favorable: si el proceso ha tenido una dispersión alta debido a que el envasado ha sido irregular, por debajo del peso nominal al principio y por encima al final, es posible que, si eliminamos estas irregularidades, la desviación típica del proceso pueda ser menor de la observada.

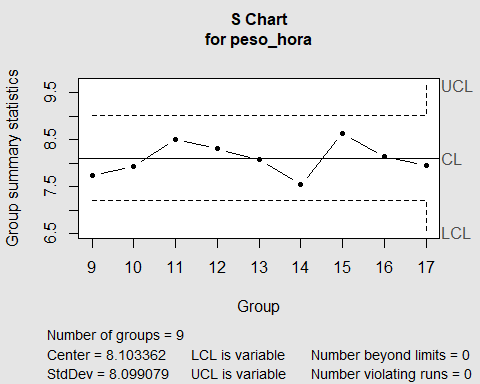
Veamos la dispersión de los datos de forma horaria, utilizando el diagrama de caja.

df |>  
 ggplot(aes(x=factor(hour(fecha)), y = peso)) +  
 geom\_boxplot()



Revisemos también la evolución de la desviación típica horaria mediante el gráfico de control correspondiente, utilizando la opción type = S:

qcc(peso\_hora, type = "S")



La variabilidad durante el proceso ha sido más o menos regular, no ha habido ningún valor que se haya situado fuera de los límites de control. Vemos que de 14:00 a 15:00 nuestro envasado ha tenido una desviación típica en torno a , la más baja de la jornada de envasado. Veamos una tabla de medias y desviaciones típicas horarias en detalle, usando de nuevo la formulación tidyverse:

df |>  
 group\_by(hora = hour(fecha)) |>  
 summarise(media=mean(peso), desv\_t=sd(peso))

## # A tibble: 9 × 3  
## hora media desv\_t  
## <int> <dbl> <dbl>  
## 1 9 250. 7.74  
## 2 10 249. 7.94  
## 3 11 249. 8.51  
## 4 12 249. 8.30  
## 5 13 251. 8.07  
## 6 14 251. 7.54  
## 7 15 250. 8.64  
## 8 16 250. 8.13  
## 9 17 251. 7.95

Efectivamente, la desviación típica en el periodo horario entre 14:00 y 15:00 ha sido de , más baja que la desviación típica global global de 8.15. Esto quiere decir que nuestro proceso debe ser capaz de repetir esta desviación si somos capaces de mantener durante todo el envasado las condiciones que se dieron en ese periodo.

Continuemos con el análisis de nuestro proceso, calculando la capacidad del proceso. Usamos la función process.capability()de la librería qcc.

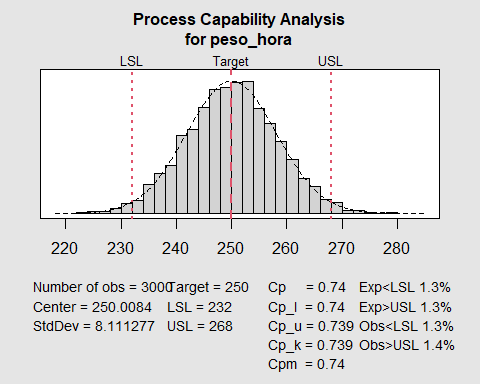
Esta función nos calcula varias medidas de nuestro proceso, y nos muestra un posible ajuste a nuestros datos de una distribución normal. Para utilizarla necesitamos el objeto peso\_grafico que guardamos con el resultado del gráfico de control de la media, y que tiene esta estructura, bastante compleja, que es simplemente el almacenamiento en una variable de todos los cálculos asociados con el gráfico de control. Solo a título informativo, veamos su estructura:

str(peso\_grafico)

## List of 11  
## $ call : language qcc(data = peso\_hora, type = "xbar", nsigmas = 3)  
## $ type : chr "xbar"  
## $ data.name : chr "peso\_hora"  
## $ data : num [1:9, 1:360] 255 259 243 245 252 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ Group : chr [1:9] "9" "10" "11" "12" ...  
## .. ..$ Samples: NULL  
## $ statistics: Named num [1:9] 250 249 249 249 251 ...  
## ..- attr(\*, "names")= chr [1:9] "9" "10" "11" "12" ...  
## $ sizes : Named int [1:9] 360 360 360 360 360 360 360 360 120  
## ..- attr(\*, "names")= chr [1:9] "9" "10" "11" "12" ...  
## $ center : num 250  
## $ std.dev : num 8.11  
## $ nsigmas : num 3  
## $ limits : num [1:9, 1:2] 249 249 249 249 249 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:9] "" "" "" "" ...  
## .. ..$ : chr [1:2] "LCL" "UCL"  
## $ violations:List of 2  
## ..$ beyond.limits : int [1:2] 6 2  
## ..$ violating.runs: num(0)   
## - attr(\*, "class")= chr "qcc"

La [función process.capability()](https://luca-scr.github.io/qcc/reference/process.capability.html) utiliza la información almacenada en esta estructura para hacer sus cálculos. Veamos el gráfico de capacidad de proceso:

process.capability(peso\_grafico, spec.limits = c(232, 268)) # prob x < (2 \* EMDT) debería ser 0%



Este gráfico nos presenta un histograma con los valores de peso de la muestra, una curva normal ajustada a estos valores (construida a partir de la media y la desviación típica de la muestra). Si proporcionamos los límites superior (*upper specification limit*) e inferior (*lower specification limit*), el gráfico nos dibuja estos límites, lo que nos permite una comparación visual muy conveniente.

Hemos suministrado a la función como límites del proceso los de la media ; sabemos que no podemos tener ningún peso por debajo del límite inferior de este intervalo ().

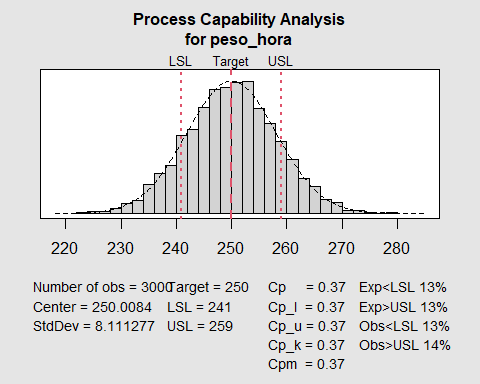
EL gráfico nos imprime además otros valores estadísticos, de los cuales por el momento sólo utilizaremos:

* Center: es el valor promedio de la muestra ()
* StdDev: es la desviación típica de la muestra ()
* Obs<LSL: el porcentaje de casos por debajo del límite inferior de la especificación que hemos indicado, en este caso , que es el límite de rechazo.
* Obs>USL: el porcentaje de casos por encima del límite superior de la especificación que hemos indicado, en este caso, .

En nuestro caso, no deberíamos encontrar ningún valor por debajo del LSL. Sin embargo tenemos un de valores en esta zona. Esto quiere decir que nuestro proceso *no es capaz* de mantenerse dentro de los límites de la especificación.

Podemos repetir el gráfico, esta vez utilizando los valores de como límites de especificación La ley de contenido efectivo nos dice que con nuestro tamaño de lote, rechazamos el lote si en nuestra muestra de 100 tenemos 7 o más valores por debajo de este límite, lo que es lo mismo, debemos tener una cantidad de valores por debajo de que debe ser **menor** de 7 casos en 100, es decir, del de los casos.

process.capability(peso\_grafico, spec.limits = c(241, 259)) # prob x < (1 \* EMDT) debería ser menor del 7%



Nuestro gráfico de capacidad de proceso, con los nuevos límites, nos dice que tenemos un de casos por debajo del valor límite, lo que de nuevo nos sitúa fuera de la especificación: debemos rechazar el lote.

## Plan de mejora: dispersión objetivo y peso a dosificar

Como hemos visto, nuestro proceso no está dentro de las especificaciones que nos exige la normativa legal. tenemos dos opciones para corregir esta situación:

1. Subir el peso medio dosificado, lo lo que nos llevará a sobredosificar, y esto aumentará nuestros costes de producción. Si no podemos aumentar el precio de venta, sobredosoficar nos ocasionará *pérdidas no previstas*.
2. Reducir la variabilidad (la desviación típica), de manera que nuestro proceso *sea capaz* de entrar en la especificación

De forma inmediata no podemos asegurar una reducción de la variabilidad porque todavía no conocemos las condiciones que han producido la desviación típica de entre las 14:00 y las 15:00 horas.

Lo que sí podemos (y tenemos) que hacer de manera inmediata para cumplir la ley es aumentar el peso dosificado, y, una vez dentro de la ley, pondremos en marcha nuestro plan de mejora. Haremos todo esto en varias etapas:

1. Subiremos lo necesario para evitar los pesos por debajo de
2. Verificaremos si con el nuevo peso, se reduce el número de envases deficientes (por debajo de ) hasta situarnos en los límites de tolerancia.
3. En caso de que la subida de peso no consiga el efecto que buscamos en el punto 2, procederemos a una nueva subida del peso medio hasta que cumplamos con ambas especificaciones.
4. Sólo entonces, una vez dentro de la ley, pondremos en marcha un plan de mejora de la variabilidad con el objetivo de alcanzar una desviación típica de o inferior. Esto nos permitirá recorrer el camino inverso, es decir, bajar el peso medio de nuevo hasta el límite que sea posible sin incumplir la norma.

Una vez que hayamos dado los tres primeros pasos, nos encontraremos en una situación de **sobredosificación**. Calcularemos el sobrepeso real que estamos entregando y su coste, de forma que podamos estableer un objetivo económico para el punto 4º.

Puesto que no debemos tener ningún valor por debajo de , y nuestro valor mínimo es de 223.63, debemos aumentar la dosificación en más de 8.37 gramos. Simularemos el proceso en simplemente sumando a los valores originales este aumento de dosificación; guardaremos los pesos modificados en una nueva columna de nuestro *dataframe* que llamaremos peso2:

incremento\_peso <- 232 - min(df$peso) + 1 # añadimos un margen de confianza como seguridad  
df$peso2 <- df$peso + incremento\_peso

Más adelante veremos cómo construir un nuevo vector de peso de forma diferente, mediante una simulación probabilística,

Hemos aumentado nuestro peso en 9.37 gramos. Veamos la media y desviación típica de nuestro nuevo peso simulado, y el sumario de :

mean(df$peso2)

## [1] 259.3784

sd(df$peso2)

## [1] 8.147409

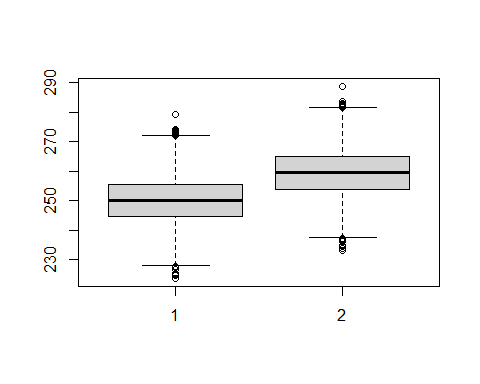
summary(df$peso2)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 233.0 253.8 259.4 259.4 264.9 288.8

Como vemos, la desviación típica no se ha modificado.

Comparemos la distribución de pesos original con la que hemos modificado mediante un diagrama de caja de las dos columnas. En este caso, utilizamos la función básica boxplot()

boxplot(df$peso, df$peso2)



El diagrama de caja nos permite comprobar el aumento de peso que hemos aplicado a la nueva variable peso2.

La comparación de los histogramas de las dos distribuciones no puede hacerse con la función básica hist() porque no puede imprimir dos histogramas solapados, necesitamos utilizar ggplot(), y también cambiar el formato de nuestro dataframe a formato ordenado, Si verificamos el *dataframe*. vemos que ahora la medida de peso no está en una columna única, sino en dos:

head(df)

## peso fecha peso2  
## 1 254.55 2020-05-16 09:00:00 263.92  
## 2 246.92 2020-05-16 09:00:00 256.29  
## 3 236.29 2020-05-16 09:00:00 245.66  
## 4 241.60 2020-05-16 09:00:00 250.97  
## 5 247.41 2020-05-16 09:00:00 256.78  
## 6 243.03 2020-05-16 09:00:00 252.40

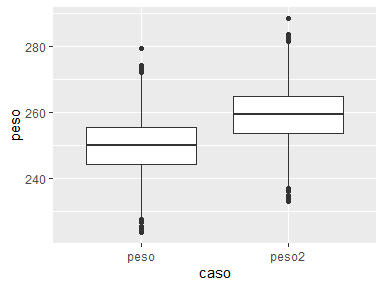
La estructura de *datos ordenados* nos exige que cada tipo de variable (en este caso el peso) esté en una sola columna. Para modificar la tabla, necesitamos \*pivotar” las medidas de peso a una sola columna, y colocar las descripciones en una nueva columna que llamaremos . Lo hacemos con la [función pivot\_longer()](https://tidyr.tidyverse.org/reference/pivot_longer.html) de la librería dplyr:

df |>  
 pivot\_longer(cols = c(peso,peso2), names\_to = "caso", values\_to = "peso") |>  
 head()

## # A tibble: 6 × 3  
## fecha caso peso  
## <dttm> <chr> <dbl>  
## 1 2020-05-16 09:00:00 peso 255.  
## 2 2020-05-16 09:00:00 peso2 264.  
## 3 2020-05-16 09:00:00 peso 247.  
## 4 2020-05-16 09:00:00 peso2 256.  
## 5 2020-05-16 09:00:00 peso 236.  
## 6 2020-05-16 09:00:00 peso2 246.

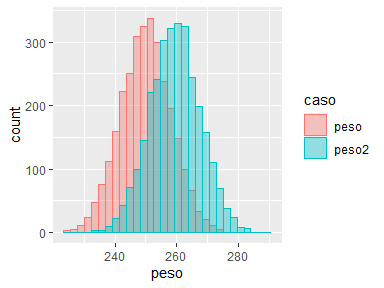
Con esta estructura, ggplot nos permite varias alternativas para nuestro histograma. No necesitamos hacer permanente la estructura del *dataframe*. podemos pasarla a ggplotmediante un pipe sin guardarla en un dataframe, los cálculos se hacen instantáneamente. veamos en primer lugar el diagrama de caja en versión ggplot:

df |>  
 pivot\_longer(cols = c(peso,peso2), names\_to = "caso", values\_to = "peso") |>  
 ggplot(aes(x = caso, y = peso))+  
 geom\_boxplot()



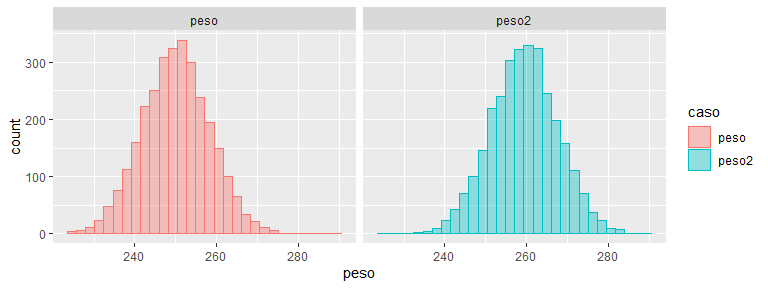
Ahora podemos hacer la comparación de los histogramas, añadiendo un poco de color para diferenciarlos:

df |>  
 pivot\_longer(cols = c(peso,peso2), names\_to = "caso", values\_to = "peso") |>  
 ggplot(aes(x = peso, colour= caso, fill = caso))+  
 geom\_histogram(position = "identity", alpha = 0.4)



A título informativo, otra presentación que puede tener interés en otras situaciones es la presentación de los gráficos clasificados lado a lado:

df |>  
 pivot\_longer(cols = c(peso,peso2), names\_to = "caso", values\_to = "peso") |>  
 ggplot(aes(x = peso, colour = caso, fill = caso))+  
 geom\_histogram(alpha = 0.4) +  
 facet\_wrap(~caso)



Volviendo a nuestro caso,

Si nuestros cálculos han sido correctos, los pesos de la nueva columna no deben tener valores por debajo de 232:

sum(df$peso2<=232)

## [1] 0

Veamos cuántos valores tenemos por debajo del límite que define los envases deficientes:

sum(df$peso2 <= 241)

## [1] 34

La ley nos dice que en una muestra de 100 debemos tener menos de 7 envases deficientes. Verificamos que el porcentaje de envases deficientes es inferior al 7%:

pct\_deficientes <- sum(df$peso2 < 241) / length(df$peso2) \*100  
pct\_deficientes

## [1] 1.1

El porcentaje de unidades defectuosas es del 1.1%, con lo que podemos estar razonablemente seguros de que el número de defectuosos en cualquier muestra que podamos tomar no excederá las 7 unidades defectuosas de cada 100.

### Análisis económico

Con la sobredosificación hemos conseguido que nuestro lote esté dentro de la ley. ¿Cuál es el coste de la decisión?

No tenemos más que calcular el peso real que hemos dosificado y el peso calculado en la simulación:

¿Qué peso real total en kilos hemos dosificado inicialmente?

peso\_real <- sum(df$peso) /1000  
peso\_real

## [1] 750.0253

¿Qué peso total teórico en kilos habríamos dosificado con la propuesta de subir el peso en 9.37 g por envase?

peso\_simul <- sum(df$peso2) /1000  
peso\_simul

## [1] 778.1353

Es decir, la decisión de incrementar el peso dosificado en 9.37 g nos hará dosificar 28.11 kg más. Suponiendo que el coste de producción de 1 kg sea de 0,85 euros, el coste del sobrepeso es de

peso\_mas <- peso\_simul - peso\_real  
coste\_unitario <- 0.85  
coste\_total2 <- peso\_mas \* coste\_unitario  
coste\_total2

## [1] 23.8935

En resumen, nuestra decisión de aumentar la dosificación en 9.37 g por envase para que el peso medio de nuestro lote de 3000 unidades cumpla con la normativa de contenido efectivo ha aumentado el coste del producto que entregamos como sobrepeso (gratuitamente) en 23.89 euros por cada lote de 3000 unidades. Si envasamos diariamente durante un año, y suponiendo 250 días útiles, el coste anual que estaríamos asumiendo con esta decisión sería de 5973.37 euros.

## Un plan de ahorro

Si fuésemos capaces de reducir la dispersión de peso, reduciendo la desviación típica de 8.15 a 6, mejorando ligeramente la mejor situación actual, podríamos bajar el peso medio, teniendo como límite el marcado por el peso mínimo de 232 g, lo que nos permitiría obtener un ahorro sobre esta cantidad.

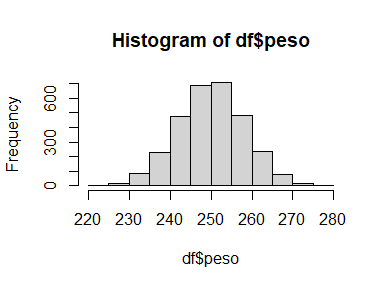
Para simular esta nueva columna, con una desviación típica más reducida, utilizaremos una *función de probabilidad*

Para entender su uso, volvamos al histograma.

Hemos visto que cuando construimos un histograma, agrupamos el número de observaciones en lo que hemos llamado intervalos de frecuencia. la función básica de histograma de R hace estos cálculos automáticamente y los guarda dentro de la estructura del histograma, presentándonos sólo el gráfico. Podemos repasar la [construcción básica de un histograma](https://www.r-bloggers.com/2015/03/how-to-make-a-histogram-with-basic-r/) [aquí](https://www.r-bloggers.com/2015/03/how-to-make-a-histogram-with-basic-r/)

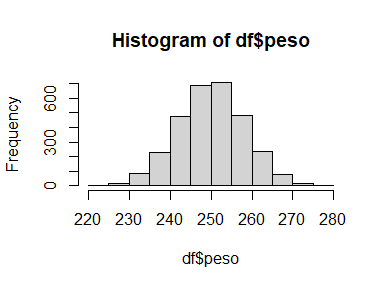
Nosotros podemos recuperar estos datos guardando el \*objeto+ histograma en una variable e inspeccionando su estructura, cmomo hemos hecho otra veces:

h <- hist(df$peso)



Hemos “guardado” nuestro histograma en un *objeto* , que podemos reusar para ver la gráfica

plot(h)



Veamos su estructura:

str(h)

## List of 6  
## $ breaks : int [1:13] 220 225 230 235 240 245 250 255 260 265 ...  
## $ counts : int [1:12] 3 16 83 227 471 686 704 479 235 79 ...  
## $ density : num [1:12] 0.0002 0.00107 0.00553 0.01513 0.0314 ...  
## $ mids : num [1:12] 222 228 232 238 242 ...  
## $ xname : chr "df$peso"  
## $ equidist: logi TRUE  
## - attr(\*, "class")= chr "histogram"

La variable breaks contiene los límites de los intervalos, mientras que la variable midscontiene los puntos medios de esos mismos intervalos. Podemos acceder a estos elementos del histograma usando la notación habitual de :

h$breaks

## [1] 220 225 230 235 240 245 250 255 260 265 270 275 280

h$mids

## [1] 222.5 227.5 232.5 237.5 242.5 247.5 252.5 257.5 262.5 267.5 272.5 277.5

Podríamos utilizar las opciones de la función hist()para definir los intervalos que nos interesan, y recuperarlos utilizando estas variables (queda como ejercicio voluntario).

Como alternativa, vamos a utilizar las funciones de la biblioteca dplyrpara hacer el cálculo de una tabla de frecuencias. Para ello, vamos a agrupar los valores originales de peso en grupos definidos por la parte entera del peso, y calcularemos el número de valores en cada clase:

df |>  
 mutate (intervalo = cut(peso, c(0,232,241,250,max(peso)))) |>  
 group\_by(intervalo) |>  
 summarize (frecuencia = n()) |>  
 mutate(frecuencia\_relativa = round(frecuencia/ sum(frecuencia) \* 100,2))

## # A tibble: 4 × 3  
## intervalo frecuencia frecuencia\_relativa  
## <fct> <int> <dbl>  
## 1 (0,232] 40 1.33  
## 2 (232,241] 359 12.0   
## 3 (241,250] 1087 36.2   
## 4 (250,279] 1514 50.5

La notación significa y (intervalo abierto a la izquierda y cerrado a la derecha): el primer intervalo, (0,232], incluye todos los valores mayores que cero y menores o iguales que 232, y así sucesivamente.

La tabla nos dice que en el peso original (columna peso) tenemos

* un 1% de valores iguales o menores que 232 (criterio de rechazo), y
* un 13% de valores iguales o menores que 241 (criterio para envases deficientes).

Ambos valores están fuera de los límites de aceptación del lote, es decir, el lote está fuera de la norma, como hemos visto antes.

En nuestra nueva columna, con el peso modificado, la distribución de frecuencias es la siguiente:

df |>  
 mutate (intervalo = cut(peso2, c(0,232,241,250,max(peso)))) |>  
 group\_by(intervalo) |>  
 summarize (frecuencia = n()) |>  
 mutate(frecuencia\_relativa = round(frecuencia/ sum(frecuencia) \* 100,2))

## # A tibble: 4 × 3  
## intervalo frecuencia frecuencia\_relativa  
## <fct> <int> <dbl>  
## 1 (232,241] 34 1.13  
## 2 (241,250] 343 11.4   
## 3 (250,279] 2606 86.9   
## 4 <NA> 17 0.57

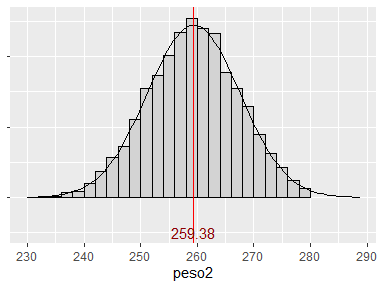
La nueva tabla nos dice que en el peso modificado (columna peso2) tenemos

* un 0% de valores iguales o menores que 232 (criterio de rechazo), y
* un 1% de valores iguales o menores que 241 (criterio para envases deficientes).

## La distribución normal y el cálculo de probabilidad: introducción

Para hacer los cálculos anteriores, hemos utilizado la distribución de frecuencias de las dos columnas de peso.

Si suponemos que nuestra distribución de pesos se ajusta a una *distribución normal*, que ya conocemos, podemos construir una *curva normal* que se ajuste a nuestros datos:



Esta [curva o distribución normal](https://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal) es una distribución de probabilidad que se construye utilizando la media y la desviación típica; el área bajo la curva nos da la probabilidad de ocurrencia de un suceso. No entraremos en los aspectos estadísticos de la construcción de la función de probabilidad y el cálculo de probabilidades, sólo necesitamos saber que tiene dos funciones que nos son útiles:

* [La función rnorm()](https://www.rdocumentation.org/packages/compositions/versions/1.40-3/topics/rnorm) permite construir un conjunto de valores que se ajusta a una distribución normal
* [La función pnorm()](https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/Normal) calcula la probabilidad de que en una muestra podamos encontrar un valor determinado.

Veamos el uso de cada función, empezando por la segunda:

Si la media de nuestro conjunto de valores de es de 250.01 y la desviación típica 8.15, podemos calcular la probabilidad de que encontremos un valor inferior o igual a :

p1 <- pnorm(232, mean(df$peso), sd(df$peso))  
p1

## [1] 0.01354124

aproximadamente, 1.

La probabilidad de encontrar envases deficientes, con un peso inferior o igual a es de

p2 <- pnorm(241, mean(df$peso), sd(df$peso))  
p2

## [1] 0.1344323

Aproximadamente, del 13.

Es decir, según el cálculo de probabilidad, en una muestra de 100 unidades, cabe esperar que encontremos 41 envases por debajo del límite de rechazo 232 g y 403 envases deficientes. Comparemos con el resultado de la tabla de frecuencias, más arriba.

Veamos ahora la situación en nuestra columna de peso modificada:

p1 <- pnorm(232, mean(df$peso2), sd(df$peso2))  
p2 <- pnorm(241, mean(df$peso2), sd(df$peso2))

En una muestra de 100 unidades, esperamos encontrar 1 envases por debajo del límite de rechazo 232 g y 36 envases deficientes. , lo que coincide bastante aproximadamente en ambos casos con los cálculos que habíamos realizado sobre la tabla de frecuencias, de forma mucho más sencilla.

La **suposición de normalidad** nos ha simplificado enormemente nuestros cálculos, y en vez de la engorrosa tabla de frecuencias, sólo tenemos que utilizar una simple fórmula.

La distribución normal tiene otra utiidad poderosa: podemos construir una distribución de valores que se ajuste a una distribución normal, dada la media y la desviación típica.

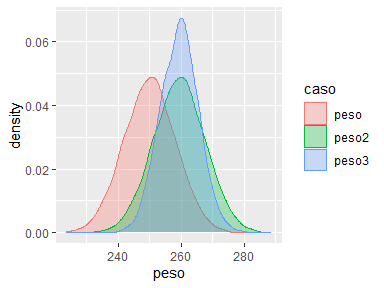
En nuestro ejemplo, intentamos encontrar los ahorros que se pueden deducir de una reducción de la variabilidad. Vamos a suponer que reducimos nuestra desviación típica de 8 a 6.

Construyamos un nuevo conjunto de datos con el peso medio de nuestra distribución de peso modificada y la nueva desviación típica:

df$peso3 <- rnorm (3000, mean(df$peso2), 6)

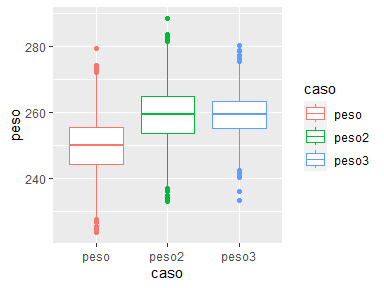
Comparemos las distribuciones de las tres columnas de peso, la original, , la modificada que hemos obtenido sumando un peso fijo, , y la tercera que hemos calculado mediante la función de distribución normal, , utilizando la función de densidad:

df |>  
 pivot\_longer(cols = c(peso,peso2, peso3), names\_to = "caso", values\_to = "peso") |>  
 ggplot(aes(x = peso, colour= caso, fill = caso))+  
 geom\_density(alpha = 0.3)



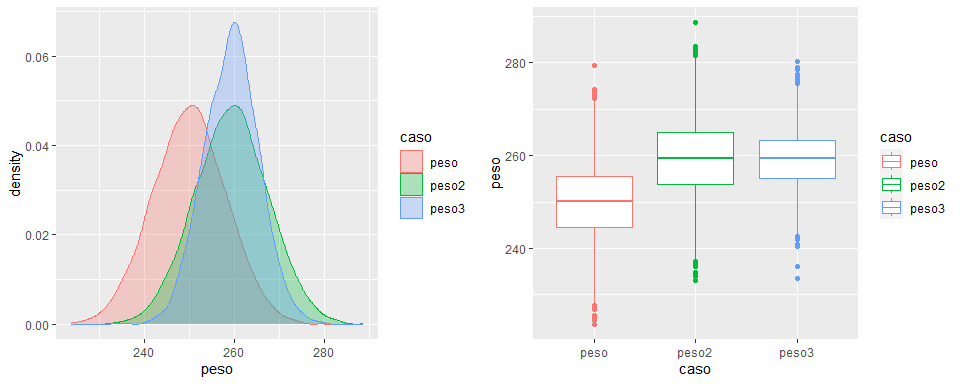
y el diagrama de caja:

df |>  
 pivot\_longer(cols = c(peso,peso2, peso3), names\_to = "caso", values\_to = "peso") |>  
 ggplot(aes(y = peso, x= caso, colour = caso))+  
 geom\_boxplot()



Con alguna instrucción adicional, podemos mostrar las curvas de densidad y los diagramas de caja lado a lado:

df2 <- df |>  
 select(peso,peso2, peso3) |>  
 pivot\_longer(cols = c(peso,peso2,peso3), names\_to = "caso", values\_to = "peso")  
  
graf1 <- ggplot(df2, aes(x = peso, colour = caso, fill = caso)) +  
 geom\_density(alpha=0.3)  
  
df2 <- df |>  
 select(peso,peso2, peso3) |>  
 pivot\_longer(cols = c(peso,peso2, peso3), names\_to = "caso", values\_to = "peso")  
  
graf2 <- ggplot(df2, aes(x = caso, y = peso, colour = caso)) +  
 geom\_boxplot()  
  
gridExtra::grid.arrange(graf1, graf2, ncol=2)



La distribución de es más estrecha, como corresponde a la reducción de variabilidad. Ahora el valor mínimo está alejado del valor de rechazo , por lo que podemos bajar el peso medio

Repetimos la simulación, creando nuestro conjunto de peso con una media de peso con 4 gramos menos que y una desviación típica de 6:

df$peso3 <- rnorm (3000, mean(df$peso2)-4, 6)

Veamos el número de envases por debajo del valor de rechazo en el total del lote. Debe ser cero; si encontramos algún valor, debemos revisar la propuesta de recuperación de 4 g y reducirla un poco (lo comparamos conla funcion de probabilidad):

sum(df$peso3 <= 232)

## [1] 0

pnorm(232, mean(df$peso3), 6 )

## [1] 4.873291e-05

En ambos casos, el resultado es aproximadamente cero.

Y ahora el número de envases deficientes:

sum(df$peso3 <= 241)

## [1] 22

pnorm(241, mean(df$peso3), 6 ) \* length(df$peso3)

## [1] 24.80737

(el número de envases deficientes esperado es igual a la probabilidad de tener un envase deficiente por el número total de envases)`

Ambos valores coinciden aproximadamente (lo que no es extraño puesto que hemos construido nuestro ejemplo de la distribución de pesos a partir de la función normal, por supuesto)

Comparando las tres columnas de peso,

df |>  
 select(peso,peso2, peso3) |>  
 pivot\_longer(cols = c(peso,peso2,peso3), names\_to = "caso", values\_to = "peso") |>  
 group\_by(caso) |>  
 summarize (peso\_medio = mean(peso), desv\_t = sd(peso))

## # A tibble: 3 × 3  
## caso peso\_medio desv\_t  
## <chr> <dbl> <dbl>  
## 1 peso 250. 8.15  
## 2 peso2 259. 8.15  
## 3 peso3 255. 6.12

Ahora calcularemos el ahorro conseguido:

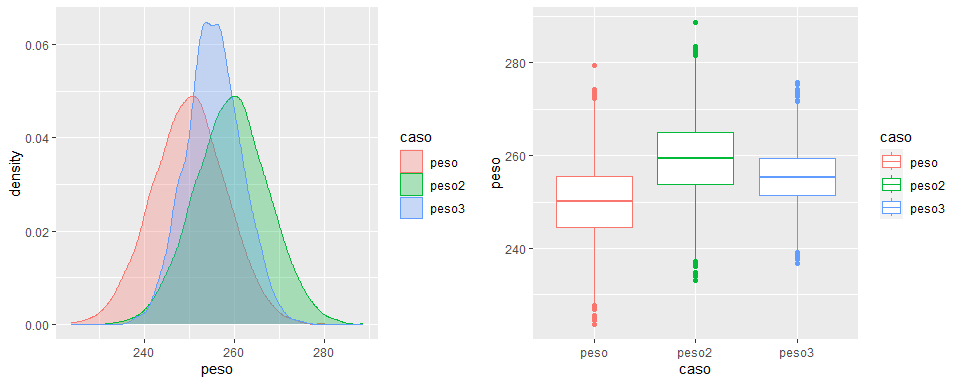
peso\_simul2 <- sum(df$peso3) /1000  
peso\_mas <- abs(peso\_simul2 - peso\_simul)  
coste\_unitario <- 0.85  
coste\_total3 <- peso\_mas \* coste\_unitario  
coste\_total3

## [1] 10.19374

La reducción de la variabilidad a una nos permitiría reducir en 4 gramos el valor de sobrepeso dosificado, con lo cual obtendríamos una mejora de 10.19 euros por lote, o lo que es lo mismo, una recuperación de 2548 euros anuales frente a nuestra pérdida anterior.

Veamos la distribución de pesos con los diagramas de densidad y de caja, incluyendo la nueva simulación:

df2 <- df |>  
 select(peso,peso2, peso3) |>  
 pivot\_longer(cols = c(peso,peso2,peso3), names\_to = "caso", values\_to = "peso")  
  
graf1 <- ggplot(df2, aes(x = peso, colour = caso, fill = caso)) +  
 geom\_density(alpha=0.3)  
  
df2 <- df |>  
 select(peso,peso2, peso3) |>  
 pivot\_longer(cols = c(peso,peso2, peso3), names\_to = "caso", values\_to = "peso")  
  
graf2 <- ggplot(df2, aes(x = caso, y = peso, colour = caso)) +  
 geom\_boxplot()  
  
gridExtra::grid.arrange(graf1, graf2, ncol=2)



Vemos claramente la reducción de la variabilidad en nuestra hipótesis 3 y cómo hemos podido reducir el peso medio manteniendo nuestra distribucion de pesos dentro de la norma.

En un artículo posterior veremos cómo optimizar aún más este ahorro mediante la correcta utilización de un *discriminador de peso*, que es un sistema de peso en continuo y un expulsor que nos permite apartar envases bajos de peso en función del criterio de configuracion que establezcamos. nos permite también optimizar este sistema.

Las acciones para reducir la variabilidad en el proceso constituyen uno de los objetivos de los proyectos Seis Sigma, e incluyen técnicas diversas que incluyen las gráficas de control de proceso y otras herramientas.

## Resumen

* Hemos importado un fichero de texto procedente de un sistema de control de peso, que en este caso no tenía formato sino que se nos ha entregado como fichero plano sin formato.
* Hemos utilizado para importar estos datos a un *dataframe* adaptando el formato numérico y convirtiendo los formatos de fecha al formato interpretable por
* Hemos estudiado la Ley de Contenido Efectivo para comprobar si nuestro lote se ajusta a la norma.
* Hemos descubierto que, aunque el peso medio del lote que hemos analizado es de , lo que es correcto según la norma, la dispersión de peso, medida por su desviación típica, es excesivamente alta, y esto nos produce que algunos envases tengan pesos demasiado bajos y hacen que el lote esté fuera de la norma.
* Hemos corregido el problema dosificando algo más de 8 gramos por encima de lo que estábamos dosificando. Esto ha hecho que desaparezca el problema con los pesos bajos fuera de norma, pero a cambio hemos *sobredosificado* nuestro producto, ocasionando un sobrecoste de casi seis mil euros anuales.
* Hemos planteado una propuesta de reducción de variabilidad; al reducir la dispersión de peso, seremos capaces de reducir el peso dosificado en exceso (*sobredosificado*). En concreto, hemos sido capaces de estimar, mediante una simulación, que una reducción de la desviación típica del peso de (actual) a (futuro) nos permitiría bajar el sobrepeso dosificado en , y esto nos produciría una economía en torno a los euros anuales, casi la mitad de la desviación.
* En este proceso, hemos utilizado algunas herramientas de control de proceso, como el gráfico de control y el gráfico de capacidad de proceso, mediante las funciones libreria qccde ; hemos utilizado funciones sencillas de para calcular el número de envases deficientes, hemos utilizado algunas funciones básicas para el cálculo de probabilidad, y hemos visto algunos cálculos y gráficos más complejos que nos permiten presentar con claridad nuestras propuestas y resultados.