Limpieza de datos

Los datos limpios son un elemento esencial de un buen análisis de datos. La mala calidad de los datos es una de las principales razones de los problemas en el análisis de inteligencia de negocios. La limpieza de datos es el proceso de transformar datos sin procesar en datos utilizables. La limpieza de datos, la comprobación de la calidad y la estandarización de los tipos de datos representan la mayor parte del cronograma de un proyecto analítico.

Anthony Goldbloom, director ejecutivo de Kaggle, afirmó: El ochenta por ciento de la ciencia de datos consiste en limpiar datos y el otro veinte por ciento en quejarse por la limpieza de datos (comunicación personal, 14 de febrero de 2016).

Este documento cubre cuatro temas clave utilizando algunos de los paquetes más nuevos disponibles en el entorno R:

- Resumir los datos para su inspección
- Encontrar y corregir datos defectuosos
- Convertir las entradas a tipos de datos adecuados para el análisis
- Adaptar las variables de cadena a un estándar

Los analistas empresariales dedican mucho tiempo a limpiar datos antes de pasar a la fase de análisis. La limpieza de datos no tiene por qué ser una tarea temida. Este documento proporciona a los analistas empresariales un proceso estructurado llamado **resumir-corregir-convertir-adaptar** para convertir los datos sin procesar en conjuntos de datos listos para el análisis.

Resumen de los datos para su inspección

Vivimos en una era de la información. Los conjuntos de datos grandes y accesibles se utilizan ampliamente en inteligencia de negocios y toma de decisiones. Cuando comiences el proceso de limpieza de datos, necesitarás una forma de resumir tus datos. Deberás comprender su contenido y estructura al comienzo del proceso. Los conjuntos de datos grandes requieren formas de resumir los datos para su inspección. ¡Afortunadamente, el lenguaje R se los proporciona! Aprenderás a limpiar datos a través de un caso de uso llamado Bike Sharing Analysis Project.

Caso de uso: Proyecto de análisis de bicicletas compartidas

Imagínate que eres un analista de negocios en el proyecto de análisis de bicicletas compartidas. Acaban de llegar nuevos datos para su análisis. A diferencia del conjunto de datos que vimos en el documento 1, Extraer, transformar y cargar, que se procesaron previamente para una competencia de Kaggle, estos datos que recibiste llegaron en forma cruda.

Antes de comenzar la fase de análisis, debes realizar la debida diligencia para limpiar los datos y prepararlos. Seguiremos un enfoque estructurado de cuatro pasos al que los autores denominan resumir-corregir-convertir-adaptar (SFCA, Summarize-Fix-Convert-Adapt).

Puedes recordar este proceso si recuerdas que San Francisco, California, es una ciudad amigable para el análisis de datos. En este caso de uso, resumirás los datos y limpiarás las deficiencias comunes, como corregir los datos faltantes, convertir fechas y discordancias de tipos de datos y adaptar cadenas. Buena suerte con tus nuevos datos. Te irá muy bien.

El primer paso en la limpieza de datos es observar los datos en un contexto más amplio. Para ello, debes aprender todo lo que puedas sobre los datos a partir de lo que cabe en tu pantalla. En otras palabras, debes resumir los datos.

Resumen mediante la función str()

El lenguaje R es un potente entorno de computación estadística. También es una herramienta de inspección de datos muy buena. En el documento 1, Extraer, transformar y cargar, aprendiste a leer un archivo de datos de valores separados por comas como un frame de datos. Ahora, asignarás tu conjunto de datos a un frame de datos llamado bike y utilizarás la función str() para resumir e inspeccionar el frame de datos:

```
> bike <- read.csv("raw_bikeshare_data.csv", stringsAsFactors = FALSE)
```

Esta función aumentará tu comprensión de los datos con una pequeña cantidad de código.

Consejo de BI: Los datos son una forma poderosa de proporcionar inteligencia de negocios. Las partes interesadas clave obtienen información de los datos para tomar decisiones informadas. Los datos deben adaptarse a la situación y el formato de la organización para ser más útiles. La limpieza de datos es la parte más grande y difícil de cualquier proyecto de inteligencia empresarial o análisis de datos. El éxito de los datos para informar a las organizaciones depende de las habilidades del analista empresarial.

Inspección e interpretación de los resultados

Aparecerá una pequeña pantalla de contenido en la consola R. Este resumen de pantalla de los datos aparece aquí para su inspección:

> str(bike)

```
'data.frame':   17379 obs. of  13 variables:
 $ datetime  : chr  "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
 $ weather   : int  1 1 1 1 1 2 1 1 1 1 ...
 $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : chr  "81" "80" "80" "75" ...
 $ windspeed : num  0 0 0 0 0 ...
 $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
 $ sources   : chr  "ad campaign" "www.yahoo.com" "www.google.fi" "AD campaign" ...
```

Puedes interpretar de inmediato algunas características clave de los datos de bicicletas compartidas:

- 17379 observaciones y 13 variables
- Tres tipos de datos: tres de caracteres (chr), siete enteros (int) y tres numéricos (num)
- Las primeras observaciones de cada variable te dan una idea del contenido

Ahora tenemos una idea de lo que contiene el conjunto de datos. También podemos ver algunos problemas con los datos. Las fechas son caracteres. Además, la variable de humedad (humidity) es numérica, pero se almacena como un tipo de datos de caracteres. Debe haber algo que esté causando los problemas con estas columnas. Otras funciones permiten ver las mismas características de diferentes maneras:

- dim() te proporciona las dimensiones de fila y columna de un frame de datos
- head() muestra las seis líneas superiores del frame de datos para ver los datos circundantes
- tail() muestra las seis líneas inferiores del frame de datos

Aprovecha la oportunidad de probar estas y otras funciones que se encuentran en la documentación de R.

Comienza a crear tu propio estilo al resumir datos. Esta sección te dio una idea para comenzar a corregir los datos.

Búsqueda y reparación de datos defectuosos

El paso de resumen reveló algunos posibles fallos en los datos. Ten en cuenta que los datos de bicicletas compartidas están en buenas condiciones en comparación con muchos conjuntos de datos que encontrarás como analista de negocios. El siguiente paso en el enfoque SFCA es corregir cualquier defecto que pueda afectar tu análisis. Primero, debes encontrar los defectos antes de poder corregirlos.

Búsqueda de defectos en los conjuntos de datos

No existe una única mejor manera de encontrar defectos en los datos. Esta actividad requiere arte combinado con algunos métodos computacionales. El enfoque presentado en esta sección comparte algunos métodos, pero solo representan algunas de las muchas formas posibles de encontrar datos defectuosos.

Consejo de R: Mantén una mente abierta y esfuérzate por convertirte en un estudiante de por vida de análisis de negocios. Los métodos cambian y se adaptan continuamente. Los mejores analistas de negocios tienen un conjunto de habilidades que va más allá de la codificación excelente.

Un consejo para los analistas de negocios es encontrar paquetes R que te ayuden a realizar tu trabajo. Existen muchos paquetes R. Como dice el refrán, para la mayoría de los problemas de datos, existe un paquete R. ¿Cuántos paquetes R existen? En un dato de hace algunos años, el 18 de junio de 2015, Joseph Rickert contó 6789 de ellos. Estos paquetes continúan evolucionando y se complementan entre sí. Por ejemplo, el primer paquete que utilizarás para limpiar los datos es stringr, publicado en abril de 2015.

Valores faltantes

Los datos faltantes, denominados NA o valores nulos, plantean muchas dificultades para los métodos de análisis. ¿Cómo se encuentra el promedio de NA? Encontrar datos faltantes es tan importante que existen funciones de R diseñadas específicamente para encontrarlos. La función is.na() está optimizada para buscar conjuntos de datos grandes y encontrar rápidamente valores faltantes:

```
> table(is.na(bike))
FALSE   TRUE
225373    554
```

Parece que hay 554 valores NA de los casi un cuarto de millón de elementos de tus datos. No son muchos, pero debes lidiar con ellos. Tus opciones incluyen eliminar las observaciones problemáticas o imputar los datos.

Imputar datos significa reparar los datos faltantes según algún método: tomar muestras aleatorias, usar una media variable o medias de grupo, o generar valores basados en modelos predictivos como la regresión lineal.

El enfoque de imputación que elijas depende de la aleatoriedad de los datos faltantes y de tu conocimiento. La imputación de datos puede ser simple o difícil: varios capítulos de varios libros están dedicados a la imputación de datos. ¿Dónde están los datos faltantes? ¿Qué variable(s) incluyen valores NA?.

El paquete stringr tiene un conjunto de funciones para detectar cadenas. Puedes utilizar el paquete en este caso, ya que el valor NA es una cadena dentro del conjunto de datos. Al ejecutar la función str_detect() se buscarán todos los valores de la cadena NA en el frame de datos de la bicicleta:

```
> install.packages("stringr")
> library(stringr)
> str_detect(bike, "NA")
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

Interpreta el resultado de la función str_detect() de la siguiente manera:

- No se detectó la cadena NA en las variables 1 a 12
- El valor TRUE en la posición 13 indica que la columna 13 contiene valores NA

Ingresar el código R table(is.na(bike$sources)) produce una tabla similar a la que creaste con table(is.na(bike)) con 554 coincidencias. Esto significa que los 554 valores NA están en la variable de fuentes. Necesitarás algunas habilidades adicionales de manipulación de cadenas para corregir este error. Te enseñaremos esa habilidad en la sección adaptación de variables de cadena a un estándar en unas páginas más adelante.

Valores erróneos

Descubriste que la variable de humedad era problemática cuando revisaste el resumen de datos. La humedad es un valor numérico, pero el resumen mostró que estaba almacenada como un tipo de datos de caracteres. Ahora determinarás qué está causando el problema en esta columna.

Una estrategia que puedes utilizar es buscar todas las instancias de valores de caracteres en la humedad y guardarlas en un objeto bad_data. La función str_subset() de stringr aplica una cadena de expresión regular a una columna y extrae todas las instancias coincidentes en un objeto:

```
> bad_data <- str_subset(bike$humidity, "[a-z A-Z]")
```

Las **expresiones regulares** son un léxico compuesto de caracteres y símbolos para formar cadenas de búsqueda potentes. Puedes obtener más información sobre este tema útil en el artículo de Hadley Wickham (2015), Introduction to stringr.

La impresión del objeto bad_data revela un solo error donde el valor x61 está incluido en la columna. Estás familiarizado con la idea de imputar datos y eres consciente de que la técnica puede utilizarse incorrectamente. En este caso, parece que el error es tipográfico y eliges imputar el valor en lugar de eliminar la fila. Pasar los elementos de bad_data a la función str_detect() asignará la posición de fila de x61 a location:

```
> bad_data
[1] "x61"
> location <- str_detect(bike$humidity, bad_data)
```

Otros valores erróneos pueden ocurrir a partir de la codificación de la forma en que se almacenan los caracteres de texto en la memoria. Las URL como este ejemplo, .../cat%20pictures.html codifican el espacio como %20. Los estilos de codificación incluyen ASCII, ANSI y UTF-8. La codificación puede convertir un análisis aparentemente simple en un desafío. La codificación puede introducir valores erróneos difíciles de encontrar, como una codificación desconocida o una codificación mal representada, en los datos.

El uso de subconjuntos [,] en R te permite inspeccionar solo aquellas observaciones que contienen el valor erróneo. Puedes esperar ver una sola instancia, pero deseas saber dónde:

```
> bike[location, ]
```

El resultado es el siguiente:

```
            datetime season holiday workingday weather  temp atemp humidity windspeed casual registered count
14177 8/18/2012 21:00      3       0          0       1 27.06 31.06      x61         0     90        248   338
          sources
14177 www.bing.com
```

Ve cómo R te permite navegar por los datos con un propósito. No es necesario buscar sin rumbo con funciones. No es necesario recorrer interminables hojas de cálculo. El enfoque

SFCA estructurado de cuatro pasos te ayudó a encontrar una falla en la fila 14177. Ahora puedes solucionarla.

Corrección de errores en conjuntos de datos

El paquete stringr también contiene una función str_replace_all() para reemplazar cadenas según los criterios que proporciones como parámetros de entrada a la función:

```
> bike$humidity <- str_replace_all(bike$humidity, bad_data, "61")
```

Nuevamente, utilizarás la creación de subconjuntos para inspeccionar los resultados. ¿Corregiste correctamente la variable de humedad al reemplazar la única instancia de bad_data con la nueva cadena 61?

```
> bike[location, ]
```

Al escribir ese código, se obtendrá el siguiente resultado:

```
             datetime season holiday workingday weather  temp atemp humidity windspeed casual registered count
14177 8/18/2012 21:00      3       0          0       1 27.06 31.06       61         0     90        248   338
           sources
14177 www.bing.com
```

¡Sí! Hiciste un buen trabajo. Aprendiste una estrategia básica para corregir los datos. Ahora, necesitarás estrategias para aquellos casos en los que los datos no contienen un defecto, pero su formato no es útil para el diseño de tu análisis. En estos casos, es cuando deseas convertir los datos.

Conversión de entradas a tipos de datos adecuados para el análisis

Otro aspecto de la limpieza de datos es investigar los datos de entrada y darles forma para que se ajusten a las necesidades de diseño de tu análisis. El tercer paso del enfoque SFCA es la conversión. Específicamente, convertir los datos de un tipo de datos a otro.

Conversión entre tipos de datos

El entorno R almacena datos en uno de los diversos tipos de datos. Experimentarás cinco tipos de datos diferentes en el Proyecto de análisis de bicicletas compartidas:

Cada tipo de datos tiene diferentes propiedades que son coherentes con las definiciones utilizadas en disciplinas como la computación, las matemáticas y las estadísticas. Algunos paquetes de la biblioteca R requieren que los datos de entrada sean de un tipo determinado.

Data type	Explanation	Example
Numeric	A number having a decimal value	`9.84`
Integer	A number without decimals	`3`
Character	A string variable	`"www.google.com"`
Factor	A categorical variable that has a character and integer representation	`"ad campaign", "blog": 1,2`
Date	A date or time in various formats	`2016-02-16 18:56:57 EST`

Cuando tienes datos que no se ajustan a un requisito de la biblioteca R o a tu diseño de análisis, puedes convertirlos a otro tipo. Los datos para el proyecto de análisis de bicicletas compartidas requieren las siguientes tres conversiones:

- **De carácter a numérico**: El error tipográfico en los datos de humedad los convirtió en un tipo de carácter, pero necesitas convertirlos nuevamente a numérico.
- **Carácter a factor**: Una estrategia común para usar read.csv() es establecer stringsAsFactors = FALSE. Esto significa que las características categóricas no se detectan automáticamente y luego elegirás cuáles son los factores.
- **Carácter a fecha**: Una situación común es la conversión incorrecta de fecha durante la lectura de un archivo. Todas las fechas son caracteres. Esta es una conversión difícil, pero se detalla en la sección conversiones de fecha y hora.

Las primeras dos conversiones son relativamente fáciles, así que comienza por ahí. R contiene una familia de funciones de conversión de tipos. Toman la forma general as.{type}(), donde type es la conversión aplicada. Aplicarás la función as.numeric() a la variable de humedad:

```
> bike$humidity <- as.numeric(bike$humidity
```

A continuación, aplicarás la función factor() a cuatro variables que en realidad son variables categóricas. Primero, aplicarás una conversión de factor simple a las dos variables de escala nominal, holiday y workingday. El uso de los parámetros levels=c(0, 1), labels =c("no", "yes") también transformará las etiquetas numéricas en valores legibles para humanos:

```
> bike$holiday <- factor(bike$holiday, levels = c(0, 1),
+  labels = c("no", "yes"))
> bike$workingday <- factor(bike$workingday, levels = c(0, 1),
+  labels = c("no", "yes"))
```

Para las variables de escala ordinal, season y weather, establecerás el factor utilizando una capacidad de R llamada factores ordenados. Al igual que la transformación anterior, proporciona los parámetros levels y labels. También puedes colocar los niveles en el orden que desees. Por ejemplo, puede que desees que las estaciones se ordenen como primavera, verano, otoño e invierno. Si no fuerzas un orden, R ordena los factores alfabéticamente. Forzarás el orden con ordered = TRUE:

```
> bike$season <- factor(bike$season, levels = c(1, 2, 3, 4),
+    labels = c("spring", "summer", "fall", "winter"),
+    ordered = TRUE)
> bike$weather <- factor(bike$weather, levels = c(1, 2, 3, 4),
+    labels = c("clr_part_cloud",
+           "mist_cloudy",
+           "lt_rain_snow",
+           "hvy_rain_snow"),
+    ordered = TRUE)
```

¡Buen trabajo! Ahora puedes abordar una conversión de tipo de datos muy común: fechas y horas. Aquí demostraremos otro paquete R llamado lubridate, publicado en diciembre de 2015. También te recomendamos que consultes la documentación del paquete para obtener más información sobre las conversiones de fecha y hora. Pueden ser muy complicadas y pueden causar mucha frustración al usarlas; sin embargo, son poderosas cuando se usan correctamente.

Conversiones de fecha y hora

El secreto para realizar conversiones de fecha exitosas es conocer el formato de fecha en el conjunto de datos de entrada. Puedes ejecutar varias funciones de R para ver el formato de la variable datetime. Sin embargo, cuando estabas arreglando la variable de humedad en la fila 14177, puedes recordar que la variable datetime estaba en el formato 8/18/2012 21:00. Expresada en términos generales, tienes el formato {m/dd/aaaa hh:mm}. Las funciones en el paquete lubridate se nombran utilizando letras que representan el orden de los datos de entrada. Con tu conjunto de datos, utilizarás la función mdy_hm():

```
> install.packages("lubridate")
> library(lubridate)
> bike$datetime <- mdy_hm(bike$datetime)
```

El uso de la función str() producirá el siguiente resultado:

```
> str(bike)
```

```
'data.frame':   17379 obs. of  13 variables:
 $ datetime  : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" "2011-01-01 03:00:0$
 $ season    : Ord.factor w/ 4 levels "spring"<"summer"<..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ workingday: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ weather   : Ord.factor w/ 4 levels "clr_part_cloud"<..: 1 1 1 1 1 2 1 1 1 1 ...
 $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : num  81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed : num  0 0 0 0 0 ...
 $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
 $ sources   : chr  "ad campaign" "www.yahoo.com" "www.google.fi" "AD campaign" ...
```

Adaptación de variables de cadena a un estándar

Ya casi has terminado con los aspectos básicos de la limpieza de datos. En este punto del proceso, has resumido, corregido y convertido tus datos de entrada. Esto significa que es hora de que realices el cuarto paso de SFCA, adaptando tus datos a un estándar.

El término estándar tiene muchos significados posibles. Puede ser que un paquete R establezca un estándar para ti. En otros casos, es posible que desees establecer uno. Por ejemplo, observa en la vista de datos anterior que la variable de fuentes es un tipo de datos de carácter. Verás que contiene la fuente de publicidad donde el cliente se enteró sobre el uso compartido de bicicletas. Dejar esto como un tipo de datos de carácter parece razonable, pero R no puede agrupar elementos de carácter para resumirlos en el análisis.

Tu estándar implícito es que las fuentes deben ser una variable categórica. ¿Qué podría suceder si utilizas la función as.factor(bike$sources)? Esto convertirá los datos, pero antes de hacerlo, debes considerar un par de preguntas:

- ¿Cuántos tipos únicos de fuentes de publicidad hay en sources?
- ¿Cuántas categorías te gustaría tener en tu conjunto de datos de análisis?

Puedes responder la primera pregunta con unique(bike$sources) para ver todos los valores únicos:

```
> unique(bike$sources)
 [1] "ad campaign"      "www.yahoo.com"    "www.google.fi"    "AD campaign"      "Twitter"
"www.bing.com"
 [7] "www.google.co.uk" "facebook page"    "Ad Campaign"      "Twitter "         NA
"www.google.com"
[13] "direct"           "blog"
```

Vale la pena analizar este resultado en detalle para comprender la adaptación de datos antes de realizar cualquier manipulación de cadenas. ¿Qué ves en los resultados?

Observado	Explicado
Dos cadenas de Twitter	El espacio en blanco adicional en el elemento 10 las trata como valores diferentes
Varias campañas publicitarias	R distingue entre mayúsculas y minúsculas, lo que hace que los elementos 1, 4 y 9 sean todos casos únicos
El valor NA	Como se descubrió en la sección encontrar fallas en los conjuntos de datos

El paquete de R stringr contiene muchas funciones para manipular cadenas. El código siguiente muestra dos funciones, así como el uso de subconjuntos para convertir todas las cadenas a minúsculas, eliminar el exceso de espacios en blanco y reemplazar todos los valores NA con la cadena desconocida:

```
> bike$sources <- tolower(bike$sources)
> bike$sources <- str_trim(bike$sources)
> na_loc <- is.na(bike$sources)
> bike$sources[na_loc] <- "unknown"
```

Al volver a ejecutar la función unique(bike$sources) se obtienen los frutos de tu trabajo. Has reducido los 14 tipos únicos originales de fuentes publicitarias a 11, de la siguiente manera:

```
> unique(bike$sources)
 [1] "ad campaign"      "www.yahoo.com"    "www.google.fi"    "twitter"          "www.bing.com"
"www.google.co.uk"
 [7] "facebook page"    "unknown"          "www.google.com"   "direct"           "blog"
```

Esto responde a la primera pregunta. Ahora debes determinar cuántos tipos únicos de fuentes publicitarias te gustaría para su análisis. George Miller (1956) estableció la Ley de Miller. Establece que la mente humana trabaja mejor con cosas y categorías cuando aparecen en cantidades de siete, más o menos dos.

La potencia de siete, más o menos dos

Tener once fuentes publicitarias diferentes no es malo en el sentido de que no hay una respuesta correcta o incorrecta. La Ley de Miller ayuda a los analistas a pensar en el diseño de análisis. Considera los histogramas: demasiadas categorías son tan problemáticas como muy pocas categorías. Para este proyecto, decidirás tener de cinco a nueve categorías de fuentes de publicidad en el conjunto de datos.

Después de hablar con personas con experiencia en publicidad, aprenderás que, cuando se trata de sitios web, lo importante no es qué motor de búsqueda usaron. Solo es importante saber que la persona encontró los servicios de Bike Sharing en la Web. Puedes decidir adaptar

todas las fuentes de motores de búsqueda etiquetándolas todas como web. Existe un paquete R para eso.

El paquete DataCombine es una biblioteca de funciones fácil de usar para combinar datos en categorías recientemente definidas. El siguiente código logra esta adaptación:

```
> install.packages("DataCombine")
> library(DataCombine)
> web_sites <- "(www.[a-z]*.[a-z]*)"
> current <- unique(str_subset(bike$sources, web_sites))
> replace <- rep("web", length(current))
```

Busca cadenas con el formato dado, unique busca los valores unicos para las cadenas dadas en la caracteristica source, creacion de instancia para reemplazar los valores unicos por la palabra web, dando las categorias posibles que se deben reemplazar

El código anterior se describe de la siguiente manera:

- Asigna una cadena de expresión regular a la variable web_sites, que ahora es una cadena de búsqueda para encontrar todas las variantes de cadenas que comienzan con www.
- Usa str_subset() para encontrar estas instancias y asígnalas a current.
- Crea una instancia de la cadena de reemplazo web para cada elemento contenido en la variable current y asígnalas a replace. La función length(current) le indica al código cuántas instancias de web se necesitan. Deben ser tantas como elementos haya en current.

Crea una tabla de referencias cruzadas almacenando las variables current y replace como vectores en un frame de datos replacements:

```
> replacements <- data.frame(from = current, to = replace)
```

El resultado es el siguiente:

```
> replacements
            from   to
1    www.yahoo.com  web
2    www.google.fi  web
3     www.bing.com  web
4 www.google.co.uk  web
5   www.google.com  web
```

Ahora estás listo para usar la función FindReplace() para adaptar las cadenas en bike$sources de cinco nombres de motores de búsqueda web diferentes a la cadena web. Observa cómo la función FindReplace() utiliza los elementos from y to del frame de datos replacements que acabas de crear como parámetros de entrada para la función en el siguiente código:

```
> bike <- FindReplace(data = bike, Var = "sources", replacements,
+   from = "from", to = "to", exact = FALSE)
```

El resultado es el siguiente:

```
> unique(bike$sources)
[1] "ad campaign"   "web"          "twitter"      "facebook page" "unknown"      "direct"       "blog"
```

Al ejecutar la función unique(), se muestra que has reducido los 14 tipos únicos originales de fuentes de publicidad a 7. George Miller estaría orgulloso de ti. Lo último que debes hacer es adaptar por completo la variable de cadena convirtiéndola en un factor con la función as.factor():

```
> bike$sources <- as.factor(bike$sources)
```

Datos listos para el análisis

Completaste el enfoque estructurado de SFCA de cuatro pasos para resumir, corregir, convertir y adaptar tus datos. Una llamada rápida a la función str(bike) te muestra los datos listos para el análisis:

```
> str(bike)
```

```
'data.frame':   17379 obs. of  13 variables:
 $ datetime  : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" "2011-01-01 03:00:0$
 $ season    : Ord.factor w/ 4 levels "spring"<"summer"<..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ workingday: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ weather   : Ord.factor w/ 4 levels "clr_part_cloud"<..: 1 1 1 1 1 2 1 1 1 1 ...
 $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : num  81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed : num  0 0 0 0 0 ...
 $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
 $ sources   : Factor w/ 7 levels "ad campaign",..: 1 7 7 1 5 7 1 7 7 7 ...
```

¡Esto se ve genial! Termina el proceso de limpieza de datos escribiendo los resultados en un archivo CSV. Aprendiste a usar write.csv() en el Documento 1, Extraer, transformar y cargar:

```
> write.csv(bike,"clean_bike_sharing_data.csv",
+  row.names = FALSE)
```

Resumen

¡Felicitaciones! Aprendiste muchos temas en este documento. La limpieza de datos es una parte muy importante del análisis de inteligencia de negocios. En este documento, aprendiste que la limpieza de datos es un proceso de cuatro pasos que SFCA puede recordar (resumir-corregir-convertir-adaptar).

Resumir los datos te brinda una descripción general y proporciona una perspectiva de los datos. Esto da forma a tus estrategias de limpieza de datos. Corregir datos defectuosos puede ser tedioso, pero existen prácticas comunes para usar. La conversión de datos es importante para obtener el tipo de datos correcto para respaldar tu análisis. Las fechas y los horarios pueden ser difíciles, pero las herramientas ayudan con esto. Adaptar los datos a un estándar es la clave para sentar las bases de un análisis de datos exitoso.
Se pueden proporcionar estándares o se puede diseñar uno.

Continuar aprendiendo también es importante, ya que los paquetes y los métodos cambian con frecuencia. Por último, el tema de la limpieza de datos está lleno de ideas interesantes que pueden resultarte útiles (se proporcionarán algunos documentos extra) .

Este documento finaliza la primera parte del curso y concluye tu aprendizaje sobre la obtención y limpieza de datos. La parte 2 contiene cuatro temas para ayudarte a aprender las técnicas de análisis que probablemente encontrarás en un entorno empresarial.

Por ejemplo, en el próximo documento, aprenderás el análisis exploratorio de datos. Aquí es donde todo tu trabajo vale la pena, ya que puedes encontrar tendencias y relaciones que ayudan a determinar el mejor enfoque de análisis para resolver las necesidades comerciales con datos limpios.