Manipulación de datos

Esta sección resume algunas de las funciones existentes para **limpiar** datos de distintos formatos a `R`. En particular, se utiliza la conceptualización de datos limpios presentada en Hadley Wickham y col. (2014) e implementada en el paquete `tidyr` (Hadley Wickham 2016). En la figura 1 podemos ver la etapa del análisis de datos correspondiente.

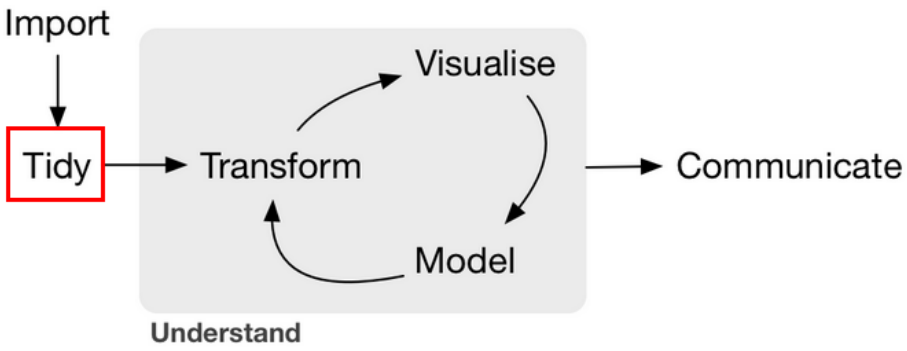

Figura 1: Limpieza de datos Grolemund y H. Wickham (2016, Introducción).

Datos limpios

Mucho del esfuerzo en analítica lidia con la limpieza de datos. Tomar datos de diferentes fuentes y poderlas poner en la forma en la que uno los necesita para realizar analítica toma mucho tiempo y esfuerzo. Existen herramientas que permiten que esta parte sea más fácil y eficiente. Entre éstas se encuentran los criterios de datos limpios.

Los conjuntos de datos limpios (*tidy datasets*) permiten manipularlos fácilmente, modelarlos y visualizarlos. Además, tienen una estructura específica: cada variable es una columna, cada observación una fila y cada tipo de unidad observacional es una tabla.

Preparación de datos

Esta actividad incluye una gran cantidad de elementos: desde revisar los outliers, hasta extraer variables de cadenas en datos no estrucutrados, imputación de valores perdidos. Los datos limpios son tan solo un subconjunto de este proceso y lidian con el cómo estructurar los datos de manera que se facilite el análisis.

El estándar de datos limpios está diseñado para facilitar la exploración inicial y el análisis de datos así como simplificar el desarrollo de herramientas para el anásis de datos que trabajen bien con datos limpios.

Los criterios de datos limpios están muy relacionados a los de las bases de datos relacionales y, por ende, al algebra relacional de Codd. Sin embargo, se expresan y enmarcan en lenguaje que le es familiar a estadísticos.

Básicamente, están creados para lidear con conjuntos de datos que se encuentran en el mundo real. Los criterios de datos limpios proporcionan un marco mental a través del cual la intuición es explícita.

Definición de datos limpios

Los datos limpios proporcionan una manera estándar de ligar la estructura de un dataset (es decir su layout físico) con su semántica (su significado).

Estructura de datos

La mayoría de los datos estadísticos están conformados por tablas rectangulares compuestas por filas y columnas. Las columnas casi siempre están etiquetadas *colnames* y las filas a veces lo están.

Tomamos el ejemplo de datos de la figura 2 en donde se presentan datos de un experimento. La tabla contiene dos columas y tres filas, ambas etiquetadas.

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Figura 2: Típica presentación de datos.

Podemos estructurar los datos de diferentes maneras pero la abstracción de filas y columnas solamente nos permite pensar en la representación transpuesta que se muestra en la figura 3. El layout cambia pero los datos son los mismos. Con columnas y filas, no podemos decir esto de manera apropiada. Además de la simple apariencia, debemos poder describir la semántica -el significado- de los valores que se muestran en una tabla.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Figura 3: Mismos datos que en 2 pero traspuestos.

Semántica

Un conjunto de datos es una colección de **valores** (normalmente cuantitativos/números o cualitativos/caracteres).

Los valores se organian de dos maneras. Cada valor pertenece a una variable y a una observación. Una variable contiene todos los valores de una medida y del mismo atributo subyacente (por ejemplo, temperatura, duración, altura, latitud) a través de unidades. Una observación, en cambio, contiene todos los valores medidos para la misma unidad (por ejemplo, una persona, un día, un municipio) a través de distintos atributos.

Los mismos datos en las figuras 2 y 3 los pensamos ahora en estos términos. Tenemos 3 variables:

1. *persona* con tres posibles valores (John, Jane, Mary)
2. *tratamiento* con dos posibles valores (a o b)
3. *resultado* con 5 o 6 valores (-, 16, 3, 2, 11, 1)

El diseño del experimento mismo nos habla de la estructura de las observaciones y los posibles valores que pueden tomar. Por ejemplo, en este caso el valor perdido nos dice que, por diseño, se debió de capturar esta variable pero no se hizo (por eso es importante guardarlo como tal). Los valores perdidos estructurales, representan mediciones de valores que no se puede hacer o que no suceden y, por tanto, se pueden eliminar (por ejemplo, hombres embarazados). En la figura 4 se muestran los mismos datos que antes pero pensados tal que las variables son columnas y las observaciones (en este caso, cada punto en el diseño experimental) son filas.

Normalmente, es fácil determinar qué son observaciones y qué son variables pero es muy dificil definir en forma precisa variables y observaciones. Por ejemplo, si tienes teléfonos de casa y celulares, se pueden considerar como dos variables distintas en muchos contextos pero en prevención de fraude necesitas una variable que

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Figura 4: Observaciones son filas, variables columnas.

guarde el tipo de teléfono y otra en la que se guarde el número pues el uso regular del mismo número de teléfono por parte de la misma persona puede ayudar a detectarlo.

En general, es más fácil describir las relaciones funcionales entre las variables que entre las filas (el radio, una combinación lineal). También es más fácil hacer comparaciones entre grupos que entre columnas (la suma, el promedio, la varianza, la moda).

Datos limpios

Éstos mapean de forma estándar el significado y la estructura de los datos. Un conjunto de datos se considera sucio o limpio dependendiendo en cómo las filas, columans y tablas mapean a observaciones, variables y tipos. En **datos limpios**:

1. Cada *variable* es una columna.
2. Cada *observación* es una fila.
3. Cada *tipo de unidad observacional* es una tabla.

Esto equivale a la tercera forma normal de Codd enfocado a un solo conjunto de datos y no a datos conectados como en bases relacionales. Los datos sucios son cualquier otro tipo de manera de organizar los datos.

La tabla 4 corresponde a datos limpios: cada fila es una observación, es decir, el resultado de un tratamiento a una persona. Cada columna es una variable. Solo tenemos un tipo de unidad observacional, es decir, cada renglón es una unidad del diseño experimental.

Con los datos así ordenados, suele ser más fácil extraer datos que, por ejemplo, la 2.

Ejercicios

1. Crea un dataframe con los valores de la tabla 2 y otro con los valores de la tabla 4.
2. Extrae el resultado para John Smith, tratamiento a en la primera configuración y en la segunda.
3. Especifica el número de tratamientos con la forma sucia y la forma limpia.
4. Cuál es la media de los resultados: usa la forma 1 y la forma 2.
5. Extrae los tratamientos del tipo a en la forma 2.

Como puedes ver, los datos limpios nos permiten preguntarle cosas a los datos de manera simple y sistemática. En particular, es una estructura muy útil para programación vectorizada como en R (el ejercicio 5) porque la forma se asegura que valores para diferentes variables de la misma observación siempre están apareados.

Por convención, las variables se acomodan de una forma particular. Las variables *fijas*, en este ejemplo, las propias al diseño experimental, van primero y posteriormente las variables *medidas*. Ordenamos éstas de forma que las que están relacionadas sean contiguas.

De sucio a limpio

Los conjuntos de datos normalmente **no cumplen** con estos criterios. Es raro obtener un conjunto de datos con el cuál podemos trabajar de manera inmediata.

Los 5 problemas más comunes para llevar datos sucios a limpios son

1. Los nombres de las columnas son valores, no nombres de variables.
2. Múltiples variables se encuentran en la misma columna.
3. Las variables están guardadas tanto en filas como en columnas.
4. Muchos tipos de unidad observacional se encuentran en la misma tabla.
5. Una sola unidad observarcional se guardó en varias tablas.

Estos problemas pueden ser resueltos con 3 herramientas: *melting*, separación de cadenas y *casting*.

Los nombres de las columnas son valores, no nombres de variables

La tabla 1 muestra datos sucios con este problema. La base acompaña al paquete `tidyr` [@tidyr] y es una muestra de los datos del reporte sobre tuberculosis de la organización mundial de la salud. Contiene observaciones anuales por país para casos de tuberculosis según distintos grupos.

Dentro de un reporte, la representación que se tiene de las variables tiene sentido. Por ejemplo, en la tabla 1 vemos los casos de tuberculosis para distintos grupos de edad de hombres en México para cierto tipo de diagnóstico.

country	year	new_sp_m014	new_sp_m1524	new_sp_m2534	new_sp_m3544	new_sp_m4554	new_sp_m5564	new_sp_m65
Mexico	2000	214	1079	1387	1162	1235	972	1126
Mexico	2001	130	1448	1639	1683	1606	1229	1566
Mexico	2002	154	1090	1292	1301	1146	986	1144
Mexico	2003	187	1207	1461	1417	1313	1005	1352
Mexico	2004	86	1053	1276	1181	1201	958	1209
Mexico	2005	100	1095	1376	1314	1238	1042	1288
Mexico	2006	129	986	1320	1333	1275	1012	1215
Mexico	2007	145	981	1286	1286	1266	942	1226
Mexico	2008	124	966	1292	1314	1267	1004	1213
Mexico	2009	103	1030	1262	1401	1360	1024	1252
Mexico	2010	125	1081	1375	1380	1392	1119	1303

Cuadro 1: Casos de tuberculosis para México del 2000 al 2010 para hombres con diagnóstico por lesiones de pulmón.

En las columnas tenemos varias variables: método de diagnóstico, género y categorías de edad. Para arreglarlo, necesitamos *juntar* (melt) las columnas con valores de variables en una sola columna que contenga esos nombres como valores. En otras palabras, debemos convertir de la columna 5 en adelante en filas.

Con el paquete **tidyr** esto se puede realizar en forma fácil con el comando `gather` de manera que obtenemos una tabla como la que se muestra en la tabla 2.

```
junta <- tidyr::gather(who, key = variables, value = casos
                       , -country, -iso2, -iso3, -year, na.rm = T)
```

Se especifica el *data.frame* como primer parámetro, la llave (parámetro *key*) será el nombre que tomará la variable con los nombres de las columnas a juntar, el valor (parámetro *value*) es el nombre de la variable que contendrá los valores correspondientes a cada valor (el diagnóstico i-ésima, el j-ésimo género y el k-ésimo grupo de edad) y, por último, especificamos las variables que **NO** se deben de juntar (en este caso, el país, su iso2, su iso3 y el año).

Hay parámetros adicionales en la función. Para estos datos en particular, es conveniente remover los grupos para los que no se tiene el dato con el parámetro `na.rm = TRUE`.

country	iso2	iso3	year	variables	casos
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1998	new_sp_m014	30
Afghanistan	AF	AFG	1999	new_sp_m014	8
Afghanistan	AF	AFG	2000	new_sp_m014	52
Afghanistan	AF	AFG	2001	new_sp_m014	129
Afghanistan	AF	AFG	2002	new_sp_m014	90

Cuadro 2: Valores de variables en una sola variable.

Ejercicio

Este tipo de formato de datos (poner valores de variables en las columnas) es útil también cuando se capturan datos al evitar la repetición de valores.

Por ejemplo, pensemos en un experimento clínico en el que seguimos a sujetos a lo largo de un tratamiento midiendo su IMC. Una forma muy sencilla de guardar los datos del experimento es utilizando un procesador de texto común. El capturista no querrá seguir criterios de datos limpios al llenar la información pues implicaría repetir el nombre de la persona, el día de la captura y el nivel de colesterol. Supongamos un experimento con 16 sujetos a lo largo de un año en donde se mide el colesterol una vez al mes (mes1, mes2, etc.). Los datos capturados se muestran en la tabla 3.

Nuevamente, queremos convertir la columan 3 a 14 en filas, es decir, observaciones. Utiliza el comando 'gather' para realizar esto y obtener el resultado que se muestra en la tabla 4.

sujetos	grupo	mes1	mes2	mes3	mes4	mes5	mes6	mes7	mes8	mes9	mes10	mes11	mes12
A	control	34.88	34.86	33.73	35.06	33.91	34.21	35.01	34.24	34.51	35.14	36.98	37.62
B	control	17.78	19.90	20.42	19.94	21.56	21.75	23.58	23.22	23.50	23.69	25.27	23.58
C	control	30.25	28.99	28.86	30.11	29.79	29.81	31.15	30.10	30.36	29.71	31.01	31.79
D	control	26.73	28.01	28.81	30.00	31.93	33.85	34.49	35.58	37.27	39.38	40.76	43.81
E	control	20.05	20.67	19.64	19.82	21.02	21.59	21.95	20.92	19.88	21.19	20.36	21.55
F	tratamiento	28.28	27.40	27.74	27.59	27.77	28.04	27.73	26.41	27.61	27.73	29.98	31.67
G	control	30.98	32.48	33.72	34.16	34.28	35.00	36.07	37.50	37.54	39.35	41.01	42.65
H	tratamiento	29.33	31.38	32.13	32.97	32.82	33.54	33.57	33.71	35.05	33.99	35.97	36.71
I	tratamiento	30.25	30.26	30.88	30.13	31.83	32.87	34.01	34.12	33.87	35.24	34.44	35.11
J	control	19.63	20.24	22.66	21.71	22.68	22.97	22.92	22.70	22.44	20.84	22.88	23.34
K	control	22.57	22.28	22.19	20.56	21.02	23.04	22.82	22.10	22.92	22.42	22.67	23.38
L	control	28.46	28.46	27.29	26.40	26.49	28.20	28.56	30.26	31.00	31.02	30.75	31.03
M	tratamiento	16.57	17.21	18.65	20.07	21.11	23.37	24.76	23.98	25.39	25.30	24.32	24.27
N	tratamiento	24.23	24.16	25.67	24.40	25.52	26.35	27.20	27.84	27.96	28.50	30.70	31.08
O	tratamiento	25.78	26.95	25.22	26.41	28.06	30.12	31.30	34.10	34.60	36.44	37.33	39.10
P	control	17.28	18.04	18.01	17.20	17.83	17.08	18.34	18.85	18.23	19.46	19.29	19.97

Cuadro 3: Mediciones de IMC en sujetos.

sujetos	grupo	mes	IMC
M	tratamiento	mes12	24.27
N	tratamiento	mes6	26.35
A	control	mes2	34.86
B	control	mes6	21.75
H	tratamiento	mes9	35.05
B	control	mes7	23.58
P	control	mes3	18.01
D	control	mes6	33.85
K	control	mes9	22.92
G	control	mes7	36.07

Cuadro 4: Muestra de datos limpios para experimentos IMC.

```
# Creamos los datos
df <- data.frame(
  sujetos = LETTERS[1:16],
  grupo = sample(c("control", "tratamiento"), size = 16, replace = T, prob = c(0.5, 0.5))
  # ,  meses = as.vector(sapply(paste0("mes",1:12), rep, 16))
  )
```

```r
m <- t(sapply(runif(16, 16, 35), FUN = function(x){cumsum(c(x, rnorm(11, mean = 0.5, sd = 1)))}))
colnames(m) <- paste0("mes",1:12)
df <- cbind(df, m)

# Respuesta: opción 1
tidyr::gather(df, key = mes, value = IMC, -sujetos, -grupo)
# opción 2
tidyr::gather(df, key = mes, value = IMC, mes1:mes12)
```

Múltiples variables se encuentran en la misma columna

Otra forma de datos sucios es cuando una columna con nombres de variables tiene realmente varias variables dentro del nombre (como en el ejemplo siguiente).

Si regresamos al ejemplo de la sección anterior, podemos notar que todavía no se tienen datos limpios. Primero, notamos una inconsistencia: todos los valores tienen el sufijo "new_" o "new" pero éste no tiene significado. Eliminamos ese pedazo de texto de los valores con la función `gsub`.

Segundo, debemos extraer los valores de las variables métdodo de diagnóstico, género y categoría de edad de la columna que acabamos de construir (que llamamos "variables"). Para eso, utilizamos la función `extract` del paquete `tidyr`. A esta función, debemos decirle cuál es el nombre de la variable que contiene varios valores (parámetro *col*), los nuevos nombres de columnas (parámtro *into*) y la expresión regular con la que irá capturando los pedazos y asignándolos a la columna correcta (parámetro *regex*).

```r
limpios <- junta %>%
  mutate(variables = gsub("new_|new", "", variables)) %>%
  extract(., col = variables
          , into = c("diagnostico", "genero", "edad")
          , regex = "([[:alnum:]]+)_([a-z])([0-9]+)")
```

De esta forma, obtenemos los datos como se ven en la tabla 5 donde tenemos una variable para el método de diagnóstico, una para el género, otra para la edad y una última con el número de casos observados.

country	iso2	iso3	year	variables	casos
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1998	new_sp_m014	30
Afghanistan	AF	AFG	1999	new_sp_m014	8
Afghanistan	AF	AFG	2000	new_sp_m014	52
Afghanistan	AF	AFG	2001	new_sp_m014	129
Afghanistan	AF	AFG	2002	new_sp_m014	90

Cuadro 5: Cada columna es una variable.

Nota

Esta forma es limpia pues cada columna es una variable, cada fila es una observación y no se mezclan unidades observacionales.

Las variables están guardadas tanto en filas como en columnas

Uno de los problemas más difíciles es cuando las variables están tanto en filas como en columnas.

Para ejemplificar este problema, se muestran los datos de temperatura máxima y mínima en algunas zonas de México (Hadley Wickham 2014, archivo data/weather.txt). Los datos que limpiaremos se ven en la tabla 6. Como se puede ver, tenemos valores del día del mes de la observación como nombres de variables: d1 (día 1), d2 (día 2), etc. Esto es homógolo al problema 1 visto anteriormente.

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
MX17004	2010	1	tmax											
MX17004	2010	1	tmin											
MX17004	2010	2	tmax		27.30	24.10								29.70
MX17004	2010	2	tmin		14.40	14.40								13.40
MX17004	2010	3	tmax					32.10					34.50	
MX17004	2010	3	tmin					14.20					16.80	
MX17004	2010	4	tmax											
MX17004	2010	4	tmin											
MX17004	2010	5	tmax											
MX17004	2010	5	tmin											
MX17004	2010	6	tmax											
MX17004	2010	6	tmin											
MX17004	2010	7	tmax			28.60								
MX17004	2010	7	tmin			17.50								
MX17004	2010	8	tmax					29.60			29.00			
MX17004	2010	8	tmin					15.80			17.30			
MX17004	2010	10	tmax					27.00		28.10				
MX17004	2010	10	tmin					14.00		12.90				
MX17004	2010	11	tmax		31.30		27.20	26.30						
MX17004	2010	11	tmin		16.30		12.00	7.90						
MX17004	2010	12	tmax	29.90					27.80					
MX17004	2010	12	tmin	13.80					10.50					

Cuadro 6: Mediciones de temperatura max y min.

También tenemos variables en las filas: la temperatura máxima y la temperatura mínima deberían ser el nombre de las columnas.

Para limpiar, lo primero que debemos hacer es juntar los dias (que son valores de la variable dia) en una sola columna. Después utilizamos la nueva variable para crear la fecha. Asi, obtenemos la tabla 8.

```
# Tidy
# Primero, juntamos la variable dias
clean1 <- tidyr::gather(raw, key = variable, value = value, d1:d31, na.rm = T)
clean1$day <- as.integer(str_replace(clean1$variable, "d", ""))
clean1$date <- as.Date(ISOdate(clean1$year, clean1$month, clean1$day))
clean1 <- dplyr::select_(clean1, "id", "date", "element", "value") %>%
  dplyr::arrange(clean1, date, element)
```

stringr

Otro paquete muy útil para realizar tareas de limpieza con cadenas. La documentación detalla todas sus funciones. En este caso, utilizamos la función `str_replace` que nos permite reemplazar una cadena de caracteres por otra.

id	date	element	value
MX17004	14642.00	tmax	27.30
MX17004	14643.00	tmax	24.10
MX17004	14642.00	tmin	14.40
MX17004	14643.00	tmin	14.40
MX17004	14915.00	tmax	31.30
MX17004	14915.00	tmin	16.30
MX17004	14944.00	tmax	29.90
MX17004	14944.00	tmin	13.80

Cuadro 7: Paso 1. Juntar las columnas, limpiar dias, crear fecha.

El segundo paso es transformar la variable element en dos columnas pues, en realidad, almacena dos variables: temperatura maxima y minima.

```
# Cast: las temperaturas van a columnas
clean2 <- tidyr::spread(clean1, key = element, value = value)
```

id	date	element	value
MX17004	14642.00	tmax	27.30
MX17004	14643.00	tmax	24.10
MX17004	14642.00	tmin	14.40
MX17004	14643.00	tmin	14.40
MX17004	14915.00	tmax	31.30
MX17004	14915.00	tmin	16.30
MX17004	14944.00	tmax	29.90
MX17004	14944.00	tmin	13.80

Cuadro 8: Paso 2. Enviar a columnas las mediciones de temperaturas.

Muchos tipos de unidad observacional se encuentran en la misma tabla

En ocasiones las bases de datos involucran diferentes tipos de unidad observacional. Para tener datos limpios, cada unidad observacional debe estar almacenada en su propia tabla.

Para este ejemplo, utilizamos la base de datos `billboard` (Hadley Wickham 2014, archivo: data/billboard.csv)

```
if (!file.exists("tidyr_datasets/billboard.csv")) download.file("https://raw.githubusercontent.com/hadl
                "tidyr_datasets/billboard.csv")
billboard <- read.csv("tidyr_datasets/billboard.csv", stringsAsFactors = F)
billboard_long <- gather(billboard, week, rank, x1st.week:x76th.week, na.rm = TRUE)
billboard_tidy <- billboard_long %>%
  mutate(
    week = extract_numeric(week),
    date = as.Date(date.entered) + 7 * (week - 1)) %>%
    select(-date.entered)
head(billboard_tidy)
```

```
##   year     artist.inverted                                track time
## 1 2000     Destiny's Child             Independent Women Part I 3:38
## 2 2000             Santana                         Maria, Maria 4:18
## 3 2000       Savage Garden                 I Knew I Loved You 4:07
## 4 2000             Madonna                               Music 3:45
## 5 2000 Aguilera, Christina Come On Over Baby (All I Want Is You) 3:38
## 6 2000               Janet                 Doesn't Really Matter 4:17
##   genre date.peaked week rank       date
## 1  Rock  2000-11-18    1   78 2000-09-23
## 2  Rock  2000-04-08    1   15 2000-02-12
## 3  Rock  2000-01-29    1   71 1999-10-23
## 4  Rock  2000-09-16    1   41 2000-08-12
## 5  Rock  2000-10-14    1   57 2000-08-05
## 6  Rock  2000-08-26    1   59 2000-06-17
```

Ejercicio

¿Cuáles son las unidades observacionales en esta tabla?

```
# Respuesta

# Tenemos por un lado una unidad observacional: las características de la
# canción.

# Por el otro tenemos otra unidad observacional: las posiciones que tuvieron
```

```
# las canciones en cada semana.
```

Debemos separar las unidades observacionales, esto significa separar la base de datos en dos: la tabla *canciones* que almacena artista, nombre de la canción y duración; la tabla *posiciones* que almacena el ranking de la canción en cada semana.

```
canciones <- billboard_tidy %>%
  select(artist.inverted, track, year, time) %>%
  unique() %>%
  arrange(artist.inverted) %>%
  mutate(song_id = row_number(artist.inverted))

head(canciones)
```

```
##   artist.inverted
## 1           2 Pac
## 2         2Ge+her
## 3    3 Doors Down
## 4    3 Doors Down
## 5        504 Boyz
## 6          98\xa1
##                                                        track year time
## 1                           Baby Don't Cry (Keep Ya Head Up II) 2000 4:22
## 2 The Hardest Part Of Breaking Up (Is Getting Back Your Stuff) 2000 3:15
## 3                                                  Kryptonite 2000 3:53
## 4                                                       Loser 2000 4:24
## 5                                                Wobble Wobble 2000 3:35
## 6                            Give Me Just One Night (Una Noche) 2000 3:24
##   song_id
## 1       1
## 2       2
## 3       3
## 4       4
## 5       5
## 6       6
```

```
posiciones <- billboard_tidy %>%
  left_join(canciones, c("artist.inverted", "track", "year", "time")) %>%
  select(song_id, date, week, rank) %>%
  arrange(song_id, date) %>%
  tbl_df
posiciones
```

```
## # A tibble: 5,307 × 4
##    song_id       date  week  rank
##      <int>     <date> <dbl> <int>
## 1        1 2000-02-26     1    87
## 2        1 2000-03-04     2    82
## 3        1 2000-03-11     3    72
## 4        1 2000-03-18     4    77
## 5        1 2000-03-25     5    87
## 6        1 2000-04-01     6    94
## 7        1 2000-04-08     7    99
## 8        2 2000-09-02     1    91
## 9        2 2000-09-09     2    87
```

```
## 10       2 2000-09-16     3    92
## # ... with 5,297 more rows
```

Una sola unidad observarcional se guardó en varias tablas

Este ejemplo y datos se toman de https://dl.dropboxusercontent.com/u/1351973/tutoriales/intro_r_2.html.

```
if(!file.exists("rprog-data-specdata.zip")) {
        temp <- tempfile()
        download.file("https://drive.google.com/open?id=0B58pFa0ldIHJYVhqLWFyckxyZm8", "tidyr")
        unzip(temp)
        unlink(temp)
}
```

Es común que los valores sobre una misma unidad observacional estén separados en varios archivos. Muchas veces, cada archivo es una variable, e.g. el mes o el nombre del paciente, etc. Para limpiar estos datos debemos:

1. Leemos los archivos en una lista de tablas.
2. Para cada tabla agregamos una columna que registra el nombre del archivo original.
3. Combinamos las tablas en un solo data frame.

La carpeta `tidyr_datasets/specdata` contiene 332 archivos csv que almacenan información de monitoreo de contaminación en 332 ubicaciones de EUA. Cada archivo contiene información de una unidad de monitoreo y el número de identificación del monitor es el nombre del archivo.

Primero creamos un vector con los nombres de los archivos en un directoriocon extension .csv.

```
paths <- dir("tidyr_datasets/specdata", pattern = "\\.csv$", full.names = TRUE)
names(paths) <- basename(paths)
specdata_US <- tbl_df(ldply(paths, read.csv, stringsAsFactors = FALSE))
specdata_US
```

```
## # A tibble: 772,087 × 5
##        .id       Date sulfate nitrate    ID
##      <chr>      <chr>   <dbl>   <dbl> <int>
## 1  001.csv 2003-01-01      NA      NA     1
## 2  001.csv 2003-01-02      NA      NA     1
## 3  001.csv 2003-01-03      NA      NA     1
## 4  001.csv 2003-01-04      NA      NA     1
## 5  001.csv 2003-01-05      NA      NA     1
## 6  001.csv 2003-01-06      NA      NA     1
## 7  001.csv 2003-01-07      NA      NA     1
## 8  001.csv 2003-01-08      NA      NA     1
## 9  001.csv 2003-01-09      NA      NA     1
## 10 001.csv 2003-01-10      NA      NA     1
## # ... with 772,077 more rows
```

Las variables quedaron un poco sucias... las limpiamos y seleccionamos solo las de interes.

```
specdata <- specdata_US %>%
  mutate(
    monitor = extract_numeric(.id),
    date = as.Date(Date)) %>%
    select(id = ID, monitor, date, sulfate, nitrate)
specdata
```

```
## # A tibble: 772,087 × 5
##       id monitor       date sulfate nitrate
```

```
##    <int>   <dbl>     <date>   <dbl>   <dbl>
## 1      1       1 2003-01-01      NA      NA
## 2      1       1 2003-01-02      NA      NA
## 3      1       1 2003-01-03      NA      NA
## 4      1       1 2003-01-04      NA      NA
## 5      1       1 2003-01-05      NA      NA
## 6      1       1 2003-01-06      NA      NA
## 7      1       1 2003-01-07      NA      NA
## 8      1       1 2003-01-08      NA      NA
## 9      1       1 2003-01-09      NA      NA
## 10     1       1 2003-01-10      NA      NA
## # ... with 772,077 more rows
```