Manipulación de datos

Un proyecto de datos tiene una gran cantidad de componentes. Sin embargo, en básicamente todos se necesita iterar sobre el ciclo que se muestra en la figura 1.

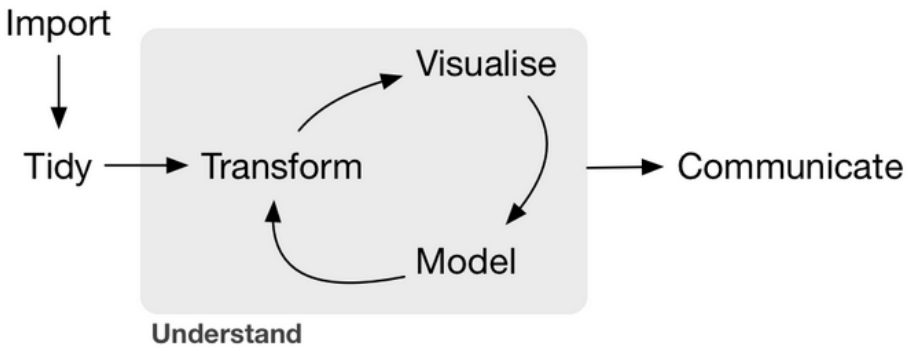

Figura 1: Modelo de las herramientas que se necesitan en un proyecto de datos según **grolemund2016r**

Primero es necesario **importar** nuestros datos a R. Los datos pueden estar en una gran cantidad de formatos o lugares.

Después, normalmente es necesario **arreglar** nuestros datos, es decir, seguir criterios de datos limpios de tal forma que como guardemos los datos equivalga a la semántica de los datos que tenemos. Es muy importante primero limpiar porque esto provee de consistencia a lo largo del análisis.

Posteriormente, en casi todo proyecto, será necesario **transformar** los datos. A veces esto implica enfocarse en un subconjunto de los datos, generar nuevas variables, calcular estadísticos, arreglar los datos de cierta manera, entre muchos otros.

Solamente después de estas etapas podemos empezar a generar conocimiento a partir de los datos. Para esto tenemos dos herramientas fundamentales: la estadística descriptiva (en el diagrama reducido a **visualización**) y la generación de **modelos**. La primera es fundamental pues permite derivar preguntas pertinentes a los datos, encontrar patrones, respuestas, plantear hipótesis. Sin embargo, éstas no escalan de la misma manera que los modelos pues estos, una vez que aceptamos sus supuestos generan los resultados que esperamos o contestan la pregunta planteada.

Por último, necesitamos **comunicar** los resultados.

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 esfuezo. 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.

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.

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 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. Se muestran distintas religiones con el numero de personas que pertenecen a distintos niveles de ingreso. Dentro de un reporte, este tipo de representación tiene mucho sentido y permite visualizar muchas cosas rápidamente.

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k	$75-100k	$100-150k	>150k	Don't know/refused
Agnostic	27	34	60	81	76	137	122	109	84	96
Atheist	12	27	37	52	35	70	73	59	74	76
Buddhist	27	21	30	34	33	58	62	39	53	54
Catholic	418	617	732	670	638	1116	949	792	633	1489
Don't know/refused	15	14	15	11	10	35	21	17	18	116
Evangelical Prot	575	869	1064	982	881	1486	949	723	414	1529
Hindu	1	9	7	9	11	34	47	48	54	37
Historically Black Prot	228	244	236	238	197	223	131	81	78	339
Jehovah's Witness	20	27	24	24	21	30	15	11	6	37
Jewish	19	19	25	25	30	95	69	87	151	162
Mainline Prot	289	495	619	655	651	1107	939	753	634	1328
Mormon	29	40	48	51	56	112	85	49	42	69
Muslim	6	7	9	10	9	23	16	8	6	22
Orthodox	13	17	23	32	32	47	38	42	46	73
Other Christian	9	7	11	13	13	14	18	14	12	18
Other Faiths	20	33	40	46	49	63	46	40	41	71
Other World Religions	5	2	3	4	2	7	3	4	4	8
Unaffiliated	217	299	374	365	341	528	407	321	258	597

Cuadro 1: Variable de ingreso en columnas.

El conjunto de datos tiene 3 variables: *religion*, *ingreso* y *frecuencia*. Para arreglarlo, necesitamos *juntar* (melt) las columnas con nombres de niveles de ingreso en una sola columna que contenga esos nombres como valores. En otras palabras, debemos convertir de la columna 2 en adelante en filas.

Con el paquete **tidyr** esto se puede realizar en forma fácil con el comando `gather`.

```
limpios <- tidyr::gather(raw, key = income, value = freq, -religion)
```

Con este comando, obtenemos la tabla 2. 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 (la religión i-ésima, grupo de ingreso j-ésimo) y, por último, especificamos las variables que **NO** se deben de juntar (en este caso, religión).

Religion	Income	Freq
Other Faiths	$100-150k	40
Other Christian	$40-50k	13
Historically Black Prot	$40-50k	197
Don't know/refused	$50-75k	35
Don't know/refused	$75-100k	21
Jehovah's Witness	$30-40k	24
Buddhist	$20-30k	30
Historically Black Prot	$75-100k	131
Jewish	$50-75k	95
Mainline Prot	$100-150k	753

Cuadro 2: Datos limpios para religión, ingreso y frecuencia.

Nota

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

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.

sujetos	grupo	mes1	mes2	mes3	mes4	mes5	mes6	mes7	mes8	mes9	mes10	mes11	mes12
A	tratamiento	16.64	16.19	15.98	16.51	15.91	16.43	16.39	16.33	16.39	16.05	15.52	16.59
B	control	17.09	16.19	15.70	12.55	13.11	14.04	14.19	14.26	16.07	15.59	17.06	18.40
C	control	27.55	26.38	28.05	27.42	27.60	27.25	26.80	27.22	26.20	26.20	26.90	27.39
D	tratamiento	31.04	31.49	31.54	33.30	34.09	33.49	34.42	35.26	35.16	37.44	38.32	40.61
E	tratamiento	17.51	18.96	18.53	18.35	18.61	19.35	20.47	20.44	22.05	22.36	24.09	25.53
F	control	34.83	35.73	36.81	36.24	37.68	38.37	38.34	39.37	40.07	39.51	41.40	41.97
G	tratamiento	18.14	17.59	19.33	19.50	20.39	22.39	23.10	24.81	25.37	25.89	26.11	27.71
H	control	22.17	23.16	23.42	24.90	23.86	24.74	24.88	29.70	31.18	33.48	34.09	33.97
I	tratamiento	19.12	19.82	18.75	19.58	19.52	20.96	21.34	22.68	23.51	22.70	23.44	23.85
J	tratamiento	34.96	35.02	37.66	36.45	36.84	36.98	36.36	36.71	37.01	37.74	37.29	37.13
K	control	20.82	23.25	22.48	21.31	20.13	19.46	21.18	21.59	23.25	23.62	25.06	25.16
L	control	22.96	25.50	23.87	24.36	23.95	25.41	25.91	26.05	28.06	27.98	27.64	27.61
M	control	25.28	25.53	27.46	27.49	28.30	28.14	29.90	32.10	32.05	31.79	31.37	32.63
N	control	27.08	25.76	28.81	30.29	29.90	30.16	31.33	31.52	32.03	30.89	31.50	30.76
O	tratamiento	32.31	32.29	33.01	33.65	34.18	33.21	34.17	34.65	37.09	36.37	36.85	38.04
P	control	32.55	33.40	35.94	35.35	34.98	35.12	34.00	36.72	38.13	38.73	40.54	40.26

Cuadro 3: Mediciones de IMC en sujetos.

Ejercicio

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	mes	IMC
D	tratamiento	mes4	33.30
J	tratamiento	mes8	36.71
I	tratamiento	mes10	22.70
N	control	mes7	31.33
K	control	mes9	23.25
O	tratamiento	mes11	36.85
D	tratamiento	mes12	40.61
C	control	mes5	27.60
K	control	mes11	25.06
F	control	mes4	36.24

Cuadro 4: Muestra de datos limpios para experimentos IMC.

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).

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0		
AE	2000	2	4	4	6	5	12	10		3
AF	2000	52	228	183	149	129	94	80		93
AG	2000	0	0	0	0	0	0	1		1
AL	2000	2	19	21	14	24	19	16		3
AM	2000	2	152	130	131	63	26	21		1
AN	2000	0	0	1	2	0	0	0		0
AO	2000	186	999	1003	912	482	312	194		247
AR	2000	97	278	594	402	419	368	330		121
AS	2000					1	1			

El primer paso es pasar las columnas que son valores de variable a una sola columna (tabla 5).

Posteriormente, debemos separar en las columnas apropiadas las variables que estan contenidas en los antiguos nombres de variables (tabla 6).

stringr

Otro paquete muy útil para realizar tareas de limpieza con cadenas. La documentación detalla todas sus funciones.

Las variables están guardadas tanto en filas como en columnas

El problema más difícil 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.

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 9.

country	year	column	cases
AD	2000	m014	0
AE	2000	m014	2
AF	2000	m014	52
AG	2000	m014	0
AL	2000	m014	2
AM	2000	m014	2
AN	2000	m014	0
AO	2000	m014	186
AR	2000	m014	97
AS	2000	m014	
AT	2000	m014	1
AU	2000	m014	3
AZ	2000	m014	0
BA	2000	m014	4
BB	2000	m014	0

Cuadro 5: Paso 1. Juntar las columnas cuyos nombres son 2 variables.

country	year	sex	age	cases
AD	2000	m	0-14	0
AE	2000	m	0-14	2
AF	2000	m	0-14	52
AG	2000	m	0-14	0
AL	2000	m	0-14	2
AM	2000	m	0-14	2
AN	2000	m	0-14	0
AO	2000	m	0-14	186
AR	2000	m	0-14	97
AS	2000	m	0-14	
AT	2000	m	0-14	1
AU	2000	m	0-14	3
AZ	2000	m	0-14	0
BA	2000	m	0-14	4
BB	2000	m	0-14	0

Cuadro 6: Paso 2. Separar las columnas.

```
# 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)
```

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	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 7: Mediciones de temperatura max y min.

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 1. Juntar las columnas, limpiar dias, crear fecha.

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.

```
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
```

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 9: Paso 2. Enviar a columnas las mediciones de temperaturas.

```
## 6  Rock  2000-08-26    1   59 2000-06-17
```

¿Cuáles son las unidades observacionales en esta tabla?

Separemos esta base de datos en dos: la tabla canción que almacena artista, nombre de la canción y duración; la tabla rango que almacena el ranking de la canción en cada semana.

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

head(cancion)
```

```
##   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
```

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

```
  tbl_df
rango
```

```
## # 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.

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
```