
CURSO: CC50 – ADMINISTRACION DE LA INFORMACION

CLASE: SESION #7-01 (PRACTICA)

TEMA: PRE-PROCESAMIENTO DE DATOS – LIMPIEZA DE DATOS #2 CON R / RSTUDIO

PROFESOR/A: Ing. PATRICIA REYES SILVA

Esta clase es la continuación de la Sesión #6, por tanto, continuaremos viendo en la práctica:

- Como se identifican los ‘ruidos’ en los datos al explorar un conjunto de datos.
- Como aplicar ciertas técnicas de limpieza de datos para dejar preparados los datos para el análisis.

OBJETIVO PRINCIPAL

Explorar un conjunto de datos, para identificar el ‘ruido’ (valores inconsistentes o posibles errores) y solucionarlo a partir de la aplicación de tareas/técnicas de limpieza de datos utilizando R/RStudio.

COMPETENCIAS

- Aprender a identificar los principales errores o inconsistencias en los datos.
- Aprender a limpiar los datos que presenten inconsistencias y generar un nuevo conjunto de datos limpio.

ACTIVIDADES

1. Transformación lineal de datos (reescalado lineal)
2. Normalización / estandarización de datos
 - **Caso#1:** Normalización ajustando al promedio y desviación típica
 - **Caso#2:** Normalización ajustando al promedio
 - **Caso#3:** Normalización ajustando a la desviación típica

APLICANDO LIMPIEZA DE DATOS EN R/RSTUDIO

Pasos iniciales

- Configuramos nuestro directorio de trabajo en R. Por ejemplo:

```
> setwd("E:/Patricia/developer/r-course/scripts")
```
- Abrimos el dataset contenido en el CSV data-conversion.csv. En este archivo se muestran datos de estudiantes:

	A	B	C	D	E
1	Age	State	Gender	Height	Income
2	23	NJ	F	61	5000
3	13	NY	M	55	1000
4	36	NJ	M	66	3000
5	31	VA	F	64	4000
6	58	NY	F	70	30000
7	29	TX	F	63	10000
8	39	NJ	M	67	50000
9	50	VA	M	70	55000
10	23	TX	F	61	2000
11	36	VA	M	66	20000

1. Transformación lineal de datos (reescalado lineal)

A veces es necesario el cálculo de distancias durante la preparación de los datos. Como algunos datos presentan valores más elevados que otros, estos tienden a dominar en los cálculos basados en la distancia, por ello, se suelen 'reescalar', es decir, se transforman sus valores para que se encuentren dentro del rango 0 y 1. El reescalado de datos, por tanto, es una técnica de análisis de datos basados en la distancia y que podemos aplicarla a variables numéricas del dataset.

- Creamos un nuevo script R y lo grabamos en nuestra carpeta de scripts con el nombre:

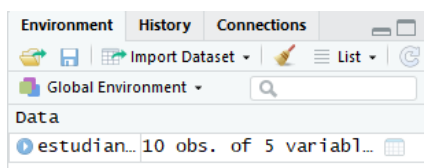
04-rescaling-data.R. En este script:

- Utilizaremos el paquete **scales**, así que si no lo tenemos deberemos instalarlo y abrirlo.

```
> install.packages("scales")
> library(scales)
```

- Cargamos dicho archivo en el dataframe **estudiantes** y visualizamos su contenido (tiene 10 observaciones y 5 variables)

```
> estudiantes <- read.csv("../data/tema01/data-conversion.csv")
```



```
> View(estudiantes)
```

	Age	State	Gender	Height	Income
1	23	NJ	F	61	5000
2	13	NY	M	55	1000
3	36	NJ	M	66	3000
4	31	VA	F	64	4000
5	58	NY	F	70	30000
6	29	TX	F	63	10000
7	39	NJ	M	67	50000
8	50	VA	M	70	55000
9	23	TX	F	61	2000
10	36	VA	M	66	20000

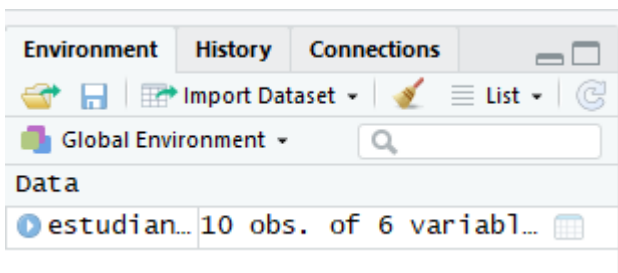
Showing 1 to 10 of 10 entries

Observamos que la variable **Income** tiene valores muy distantes uno de otros. Hay estudiantes que teniendo 13 años tienen un ingreso de 1000 y otros, mayores de 50 años, que ganan entre 30,000 y 55,000. Entonces, para 'reescalar' la variable Income del dataframe estudiantes a valores entre 0 y 1 ejecutamos lo siguiente:

- Creamos una nueva variable llamada `Income.rescaled` y haremos uso de la función **rescaled()** de la librería `scales`, al que le pasamos el valor de la variable que deseamos reescalar.

```
> estudiantes$Income.rescaled <- rescale(estudiantes$Income)
```

Se verifica que ahora existen 6 variables, en lugar de las 5 variables originales:



Y que la nueva variable creada **Income.rescaled** ahora contiene valores entre 0 y 1.

```
> view(estudiantes)
```

	Age	State	Gender	Height	Income	Income.rescaled
1	23	NJ	F	61	5000	0.07407407
2	13	NY	M	55	1000	0.00000000
3	36	NJ	M	66	3000	0.03703704
4	31	VA	F	64	4000	0.05555556
5	58	NY	F	70	30000	0.53703704
6	29	TX	F	63	10000	0.16666667
7	39	NJ	M	67	50000	0.90740741
8	50	VA	M	70	55000	1.00000000
9	23	TX	F	61	2000	0.01851852
10	36	VA	M	66	20000	0.35185185

La función **rescale()** hace que el mínimo valor de la variable a reescalar tome el valor de 0 y el máximo el valor de 1. El resto de valores son reescalados proporcionalmente de forma lineal.

Los valores de reescalado resultan de aplicar la siguiente formula:

Valor reescalado = $\frac{\text{Valor original} - \text{valor mínimo}}{\text{Valor máximo} - \text{Valor mínimo}}$

Esta fórmula matemática en R hace lo mismo que la función **rescale()**. Aplicada a la variable Income sería:

```
> (estudiantes$Income - min(estudiantes$Income)) / (max(estudiantes$Income) - min(estudiantes$Income))
[1] 0.07407407 0.00000000 0.03703704 0.05555556 0.53703704 0.16666667 0.90740741
[8] 1.00000000 0.01851852 0.35185185
```

- Pero a veces, nos interesaría reescalar valores entre 0 y 100 (típico de los porcentajes). Para ello, la función **rescale()** nos ofrece un parámetro para especificar el valor mínimo y máximo para reescalar.

```
> rescale(estudiantes$Income, to = c(0,100))
[1] 7.407407 0.000000 3.703704 5.555556 53.703704 16.666667 90.740741
[8] 100.000000 1.851852 35.185185
```

Vemos que hemos obtenido los valores reescalados anteriormente en `estudiantes$Income.rescaled`, pero ahora multiplicados por 100.

Reescalado de más de una variable en un dataframe

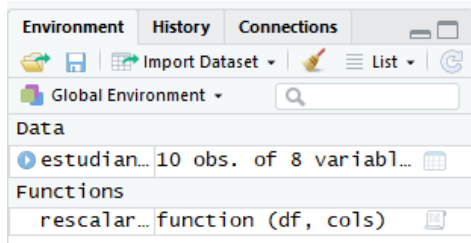
- A menudo es necesario reescalar mas de una variable dentro de un dataframe, para ello, es útil crear una función que nos solucione dicha tarea, en lugar de aplicar el reescalado una a una.

```
> rescalar.cols <- function(df, cols){
+   nombres <- names(df)
+   for (col in cols){
+     nombre <- paste(nombres[col], 'rescaled', sep = '.')
+     df[nombre] <- rescale(df[,col])
+   }
+   cat(paste("Hemos reescalado ", length(cols), " variable(s)"))
+   df
+ }
```

- Utilicemos la función **rescalar.cols()** para reescalar las variables edad (Age) y altura (Height) del dataframe `estudiantes`.

```
> estudiantes <- rescalar.cols(estudiantes, c(1,4))
Hemos reescalado 2 variable(s)
```

- Ahora observamos que tenemos 2 nuevas variables en el dataframe `estudiantes`:



```
> view(estudiantes)
```

	Age	State	Gender	Height	Income	Income.rescaled	Age.rescaled	Height.rescaled
1	23	NJ	F	61	5000	0.07407407	0.2222222	0.4000000
2	13	NY	M	55	1000	0.00000000	0.0000000	0.0000000
3	36	NJ	M	66	3000	0.03703704	0.5111111	0.7333333
4	31	VA	F	64	4000	0.05555556	0.4000000	0.6000000
5	58	NY	F	70	30000	0.53703704	1.0000000	1.0000000
6	29	TX	F	63	10000	0.16666667	0.3555556	0.5333333
7	39	NJ	M	67	50000	0.90740741	0.5777778	0.8000000
8	50	VA	M	70	55000	1.00000000	0.8222222	1.0000000
9	23	TX	F	61	2000	0.01851852	0.2222222	0.4000000
10	36	VA	M	66	20000	0.35185185	0.5111111	0.7333333

- En las nuevas variables Age.rescaled y Height.rescaled ahora tenemos los valores reescalados entre 0 y 1 de las variables originales Age y Height respectivamente.

2. Normalización / estandarización de datos

Los cálculos que involucran distancias juegan un papel importante en las técnicas de análisis de datos. Las variables con valores grandes tienden a dominar sobre las variables con valores pequeños, y a veces, la técnica de reescalado lineal con valores entre un mínimo/máximo no es suficiente, de modo que lo que se quiere es estandarizar o normalizar los valores.

Para practicar sobre la normalización/estandarización de valores, trabajaremos con un nuevo dataset llamado **BostonHousing.csv**. Este es un dataset del año 1996 creado por la Universidad de Toronto, en base al censo de Estados Unidos. Los detalles del dataset se encuentran en: <https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>

Este dataset cuenta con 506 observaciones y 14 variables. El detalle de su estructura se describe a continuación:

Nro.	Variable	Descripción
1	CRIM	Tasa de delincuencia per cápita por ciudad
2	ZN	Proporción de terrenos residenciales divididos en zonas para lotes de más de 25,000 pies cuadrados
3	INDUS	Proporción de acres comerciales no minoristas por ciudad
4	CHAS	Variable ficticia de Charles River (= 1 si el tramo limita con el río; 0 en caso contrario)
5	NOX	Concentración de óxido nítrico (partes por 10 millones)
6	RM	Número promedio de habitaciones por vivienda
7	AGE	Proporción de unidades ocupadas por el propietario construidas antes de 1940
8	DIS	Distancias ponderadas a cinco centros de empleo de Boston

9	RAD	Índice de accesibilidad a carreteras radiales
10	TAX	Tasa de impuesto a la propiedad de valor total por \$ 10,000
11	PTRATIO	Proporción alumno-maestro por ciudad
12	B	1000 (Bk - 0,63) ² , donde Bk es la proporción de [personas de ascendencia afroamericana] por ciudad
13	LSTAT	Porcentaje de menor estatus de la población
14	MEDV	Valor medio de las viviendas ocupadas por sus propietarios en \$ 1000

De este dataset se pueden realizar varias inferencias, tales como:

- ¿Cuánto se estaría dispuesto a pagar por una casa en Boston?
- ¿Cuál sería la probabilidad que ocurriera un crimen cerca?

Como se puede observar, todas las variables son numéricas y los valores son muy dispares, por ello, las estandarizaremos vía la técnica de normalización de datos.


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
19	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5
20	0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21	288.99	11.69	20.2
21	0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95	11.28	18.2
22	1.25179	0	8.14	0	0.538	5.57	98.1	3.7979	4	307	21	376.57	21.02	13.6
23	0.85204	0	8.14	0	0.538	5.965	89.2	4.0123	4	307	21	392.53	13.83	19.6

- Creamos un nuevo script R y lo grabamos en nuestra carpeta de scripts con el nombre:

05-normalizing-data.R. En este script:

- cargamos dicho archivo en el dataframe **casas** y visualizamos su contenido (tiene 506 observaciones y 14 variables)

```
> casas <- read.csv("../data/tema01/BostonHousing.csv")
```

Environment	History	Connections
 Import Dataset ▾		
Global Environment ▾		
Data		
casas	506 obs. of 14 variables	

```
> view(casas)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.00632	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
2	0.02731	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.06905	0.0	2.18	0	0.4580	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
6	0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
7	0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
8	0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
10	0.17004	12.5	7.87	0	0.5240	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2
16	0.62739	0.0	8.14	0	0.5380	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9
17	1.05393	0.0	8.14	0	0.5380	5.935	29.3	4.4986	4	307	21.0	386.85	6.58	23.1
18	0.78420	0.0	8.14	0	0.5380	5.990	81.7	4.2579	4	307	21.0	386.75	14.67	17.5

Showing 1 to 19 of 506 entries

Caso#1: Normalización ajustando al promedio y desviación típica

- A la hora de estandarizar los valores se suele utilizar la función `scale`, teniendo presente que esta función se utiliza cuando en el dataframe todas sus variables son numéricas (da error si hay alguna variable categórica).

```
> casas.z <- scale(casas)
```

Environment	History	Connections
<div> <div>Import Dataset</div> <div>List</div> </div>		
Global Environment		
Data		
casas	506 obs. of 14 variables	
casas.z	num [1:506, 1:14] -0.419 -0.417 -0.417 -0.41...	

```
> view(casas.z)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	-0.19366929	0.28454827	-1.28663623	-0.2723291	-0.14407485	0.413262920	-0.11989477	0.1400749840	-0.9818712	-0.66594918	-1.45755797	0.440615895	-1.074498970	0.159527789
2	-0.416926670	-0.48724019	-0.59279438	-0.2723291	-0.73953036	0.194082387	0.36680343	0.5566090496	-0.8670245	-0.98635338	-0.30279450	0.440615895	-0.491952525	-0.101423917
3	-0.416928959	-0.48724019	-0.59279438	-0.2723291	-0.73953036	1.281445551	-0.26554897	0.5566090496	-0.8670245	-0.98635338	-0.30279450	0.396035074	-1.207532413	1.322937477
4	-0.416338404	-0.48724019	-1.30558569	-0.2723291	-0.83445805	1.015297761	-0.80908783	1.0766711351	-0.7521778	-1.10502160	0.11292035	0.415751408	-1.360170785	1.181588636
5	-0.412074053	-0.48724019	-1.30558569	-0.2723291	-0.83445805	1.227362043	-0.51067434	1.0766711351	-0.7521778	-1.10502160	0.11292035	0.440615895	-1.025486649	1.486032293
6	-0.416631374	-0.48724019	-1.30558569	-0.2723291	-0.83445805	0.206891639	-0.35080997	1.0766711351	-0.7521778	-1.10502160	0.11292035	0.410165113	-1.042290874	0.670558212
7	-0.409837246	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.388026950	-0.07015919	0.8384142195	-0.5224844	-0.57694801	-1.50374851	0.426376321	-0.031236706	0.039924923
8	-0.403296561	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.160306916	0.97784057	1.0236248974	-0.5224844	-0.57694801	-1.50374851	0.440615895	0.909799859	0.496590409
9	-0.395543302	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.930285282	1.11638970	1.0861216287	-0.5224844	-0.57694801	-1.50374851	0.328123258	2.419379930	-0.655946292
10	-0.400333140	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.399412952	0.61584134	1.3283202075	-0.5224844	-0.57694801	-1.50374851	0.328999540	0.622727693	-0.394994586
11	-0.393956378	0.04872402	-0.47618230	-0.2723291	-0.26489191	0.131459378	0.91389483	1.2117799501	-0.5224844	-0.57694801	-1.50374851	0.392639483	1.091845624	-0.819041108
12	-0.406444832	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.392296701	0.50890509	1.1547920492	-0.5224844	-0.57694801	-1.50374851	0.440615895	0.086392864	-0.394994586
13	-0.409198989	0.04872402	-0.47618230	-0.2723291	-0.26489191	-0.563086727	-1.05066066	0.7863652700	-0.5224844	-0.57694801	-1.50374851	0.370513376	0.428078760	-0.090550930
14	-0.346886928	-0.48724019	-0.43682573	-0.2723291	-0.14407485	-0.477691714	-0.24068118	0.4333252240	-0.6373311	-0.60068166	1.17530274	0.440615895	-0.615183504	-0.231899770
15	-0.345933611	-0.48724019	-0.43682573	-0.2723291	-0.14407485	-0.268473932	0.56574575	0.3166898668	-0.6373311	-0.60068166	1.17530274	0.255720500	-0.335113097	-0.471105500
16	-0.347162460	-0.48724019	-0.43682573	-0.2723291	-0.14407485	-0.641365488	-0.42896578	0.3341187865	-0.6373311	-0.60068166	1.17530274	0.426595391	-0.585776111	-0.286264709
17	-0.297573695	-0.48724019	-0.43682573	-0.2723291	-0.14407485	-0.497617217	-1.39525519	0.3341187865	-0.6373311	-0.60068166	1.17530274	0.330533033	-0.850442645	0.061670899
18	-0.328932014	-0.48724019	-0.43682573	-0.2723291	-0.14407485	-0.419338455	0.466527459	0.2198105553	-0.6373311	-0.60068166	1.17530274	0.329437681	0.282442149	-0.547216415

Showing 1 to 19 of 506 entries

Observamos que el dataframe `casas.z` ahora tiene las variables normalizadas. La función `scale` normaliza en la normal 0,1 (a cada columna le resta la media y la divide entre su desviación típica). Los valores que están cercanos a cero, serian los muy cercanos a la media, y los muy mayores a cero los más distantes a la media de dicha variable en el dataset. Mayor información sobre la desviación estándar o típica en: https://es.wikipedia.org/wiki/Desviaci%C3%B3n_t%C3%ADpica

La función `scale()` tiene dos argumentos booleanos muy importantes `centre` y `scale` (por defecto están en `TRUE`). `Centre = TRUE`, habilita la resta de la media y `scale = TRUE` habilita la división entre la desviación típica.

Si utilizamos la función `scale()` con `center = FALSE` y `scale = FALSE`, obtenemos el dataframe inicial (no realiza la normalización). Las otras dos opciones serian:

Caso #2: Normalización ajustando al promedio (`center = TRUE` y `scale = FALSE`)

```
> casas.mean <- scale(casas, center = TRUE, scale = FALSE)
```

Global Environment	
Data	
casas	506 obs. of 14 variables
casas.mean	num [1:506, 1:14] -3.61 -3.59 -3.59 -3.58 -3...
casas.z	num [1:506, 1:14] -0.419 -0.417 -0.417 -0.41...

```
> view(casas.mean)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	-3.60720356	6.636364	-8.8267787	-0.06916996	-0.016695059	0.290365613	-3.3749012	0.294957312	-8.549407	-112.237154	-3.1555336	40.2259684	-7.67306324	1.46719368
2	-3.58621356	-11.363636	-4.0667787	-0.06916996	-0.085695059	0.136365613	10.3250988	1.172057312	-7.549407	-166.237154	-0.6555336	40.2259684	-3.51306324	-0.93280632
3	-3.58623356	-11.363636	-4.0667787	-0.06916996	-0.085695059	0.900365613	-7.4749012	1.172057312	-7.549407	-166.237154	-0.6555336	36.1559684	-8.62306324	12.16719368
4	-3.58115356	-11.363636	-8.9567787	-0.06916996	-0.096695059	0.713365613	-22.7749012	2.267157312	-6.549407	-186.237154	0.2444664	37.9559684	-9.71306324	10.86719368
5	-3.58447356	-11.363636	-8.9567787	-0.06916996	-0.096695059	0.862365613	-14.3749012	2.267157312	-6.549407	-186.237154	0.2444664	40.2259684	-7.32306324	13.66719368
6	-3.58367356	-11.363636	-8.9567787	-0.06916996	-0.096695059	0.145365613	-9.8749012	2.267157312	-6.549407	-186.237154	0.2444664	37.4459684	-7.44306324	6.16719368
7	-3.52523356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.272634387	-1.9749012	1.765457312	-4.549407	-97.237154	-3.2555336	38.9259684	-0.22306324	0.36719368
8	-3.46897356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.112634387	27.5250988	2.155457312	-4.549407	-97.237154	-3.2555336	40.2259684	6.49693676	4.56719368
9	-3.40228356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.653634387	31.4250988	2.287057312	-4.549407	-97.237154	-3.2555336	29.9559684	17.27693676	-6.03280632
10	-3.44348356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.280634387	17.3250988	2.797057312	-4.549407	-97.237154	-3.2555336	30.0359684	4.44693676	-3.63280632
11	-3.38863356	1.136364	-3.2667787	-0.06916996	-0.030695059	0.092365613	25.7250988	2.551657312	-4.549407	-97.237154	-3.2555336	35.8459684	7.79693676	-7.53280632
12	-3.49605356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.275634387	14.3250988	2.431657312	-4.549407	-97.237154	-3.2555336	40.2259684	0.61693676	-3.63280632
13	-3.51974356	1.136364	-3.2667787	-0.06916996	-0.030695059	-0.395634387	-29.5749012	1.655857312	-4.549407	-97.237154	-3.2555336	33.8259684	3.05693676	-0.83280632
14	-2.98376356	-11.363636	-2.9967787	-0.06916996	-0.016695059	-0.335634387	-6.7749012	0.912457312	-5.549407	-101.237154	2.5444664	40.2259684	-4.39306324	-2.13280632
15	-2.97556356	-11.363636	-2.9967787	-0.06916996	-0.016695059	-0.188634387	15.9250988	0.668857312	-5.549407	-101.237154	2.5444664	23.3459684	-2.39306324	-4.33280632
16	-2.98613356	-11.363636	-2.9967787	-0.06916996	-0.016695059	-0.450634387	-12.0749012	0.703557312	-5.549407	-101.237154	2.5444664	38.9459684	-4.18306324	-2.63280632
17	-2.55959356	-11.363636	-2.9967787	-0.06916996	-0.016695059	-0.349634387	-39.2749012	0.703557312	-5.549407	-101.237154	2.5444664	30.1759684	-6.07306324	0.56719368
18	-2.82923356	-11.363636	-2.9967787	-0.06916996	-0.016695059	-0.294634387	13.1250988	0.462857312	-5.549407	-101.237154	2.5444664	30.0759684	2.01693676	-5.03280632

Caso #3: Normalización ajustando a la desviación típica o estándar (`center = FALSE` y `scale = TRUE`)

```
> casas.sd <- scale(casas, center = FALSE, scale = TRUE)
```

Global Environment	
Data	
casas	506 obs. of 14 variables
casas.mean	num [1:506, 1:14] -3.61 -3.59 -3.59 -3.58 -3...
casas.sd	num [1:506, 1:14] 0.000677 0.002927 0.002925...
casas.z	num [1:506, 1:14] -0.419 -0.417 -0.417 -0.41...

```
> view(casas.sd)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.0006773027	0.6936817	0.17647585	0.0000000	0.9485077	1.0387096	0.87882214	0.9416646	0.07733878	0.6696347	0.8225709	1.0770264	0.3425033	0.9852965
2	0.0029267624	0.0000000	0.54012306	0.0000000	0.8268590	1.0143809	1.06348262	1.1436045	0.15467755	0.5474716	0.9569779	1.0770264	0.6286104	0.8867668
3	0.0029246190	0.0000000	0.54012306	0.0000000	0.8268590	1.1350766	0.82355879	1.1436045	0.15467755	0.5474716	0.9569779	1.0659821	0.2771663	1.4245745
4	0.0034690332	0.0000000	0.16654431	0.0000000	0.8074657	1.1055346	0.61733212	1.3957357	0.23201633	0.5022260	1.0053645	1.0708666	0.2022007	1.3712043
5	0.0073999612	0.0000000	0.16654431	0.0000000	0.8074657	1.1290734	0.73055460	1.3957357	0.23201633	0.5022260	1.0053645	1.0770264	0.3665748	1.4861555
6	0.0031989695	0.0000000	0.16654431	0.0000000	0.8074657	1.0158027	0.79120950	1.3957357	0.23201633	0.5022260	1.0053645	1.0694826	0.3583217	1.1782504
7	0.0094618765	0.4817234	0.60124024	0.0000000	0.9238254	0.9497676	0.89769256	1.2802264	0.38669388	0.7035689	0.8171946	1.0734988	0.8548827	0.9401371
8	0.0154911570	0.4817234	0.60124024	0.0000000	0.9238254	0.9750442	1.29531914	1.3700184	0.38669388	0.7035689	0.8171946	1.0770264	1.3170558	1.1125640
9	0.0226382014	0.4817234	0.60124024	0.0000000	0.9238254	0.8895778	1.34788672	1.4003174	0.38669388	0.7035689	0.8171946	1.0491578	2.0584585	0.6773913
10	0.0182228733	0.4817234	0.60124024	0.0000000	0.9238254	0.9485038	1.15783469	1.5177377	0.38669388	0.7035689	0.8171946	1.0493749	1.1760655	0.7759210
11	0.0241010467	0.4817234	0.60124024	0.0000000	0.9238254	1.0074298	1.27105718	1.4612378	0.38669388	0.7035689	0.8171946	1.0651409	1.4064643	0.6158103
12	0.0125890433	0.4817234	0.60124024	0.0000000	0.9238254	0.9492937	1.11739809	1.4336095	0.38669388	0.7035689	0.8171946	1.0770264	0.9126543	0.7759210
13	0.0100502297	0.4817234	0.60124024	0.0000000	0.9238254	0.9303363	0.52567582	1.2549926	0.38669388	0.7035689	0.8171946	1.0596594	1.0804672	0.8908722
14	0.0674902182	0.0000000	0.62186728	0.0000000	0.9485077	0.9398150	0.83299399	1.0838352	0.30935510	0.6945197	1.1290189	1.0770264	0.5680878	0.8375020
15	0.0683689970	0.0000000	0.62186728	0.0000000	0.9485077	0.9630378	1.13896428	1.0272893	0.30935510	0.6945197	1.1290189	1.0312209	0.7056393	0.7471832
16	0.0673262296	0.0000000	0.62186728	0.0000000	0.9485077	0.9216474	0.76155600	1.0357390	0.30935510	0.6945197	1.1290189	1.0735530	0.5825307	0.8169750
17	0.1129477351	0.0000000	0.62186728	0.0000000	0.9485077	0.9376033	0.39493081	1.0357390	0.30935510	0.6945197	1.1290189	1.0497548	0.4525445	0.9483479
18	0.0840412682	0.0000000	0.62186728	0.0000000	0.9485077	0.9462921	1.10122345	0.9803212	0.30935510	0.6945197	1.1290189	1.0494834	1.0089404	0.7184454

Normalización de más de una variable conservando las variables originales

- A menudo es necesario normalizar más de una variable dentro de un dataframe, pero conservando los valores originales de las variables, o teniendo variables categóricas, no se puede utilizar la función `scale()` para todo el dataframe, para ello, es útil crear una función que nos solucione dicha tarea, en lugar de aplicar el escalado variable por variable del dataframe.

```
> normalizar.cols <- function(df, cols){
+   nombres <- names(df)
+   for (col in cols){
+     nombre <- paste(nombres[col], "z", sep = ".")
+     df[nombre] <- scale(df[,col])
+   }
+   cat(paste("Hemos normalizado ", length(cols), " variable(s)"))
+   df
+ }
```

- Normalizamos con esta función, las variables 1, 3 y de la 5 a la 8 del dataframe original `casas`

```
> casas <- normalizar.cols(casas, c(1,3,5:8))
Hemos normalizado 6 variable(s)
```

- Observamos que el dataframe original `casas` contiene ahora 20 variables (las 14 originales más las 6 creadas con los valores normalizados).

Global Environment	
Data	
casas	506 obs. of 20 variables
casas.mean	num [1:506, 1:14] -3.61 -3.59 -3.59 -3.58 -3...
casas.sd	num [1:506, 1:14] 0.000677 0.002927 0.002925...
casas.z	num [1:506, 1:14] -0.419 -0.417 -0.417 -0.41...
Functions	
normalizar.cols	function (df, cols)

```
> view(casas)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTATIO	B	LSTAT	MEDV	CRIM.z	INDUS.z	NOX.z	RM.z	AGE.z	DIS.z
1	0.00632	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	-0.419366929	-1.28663623	-0.14407485	0.413262920	-0.11989477	0.1400749840
2	0.02731	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	-0.416926670	-0.59279438	-0.73953036	0.194082387	0.36680343	0.5566090496
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	-0.416928995	-0.59279438	-0.73953036	1.281445551	-0.26554897	0.5566090496
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	-0.416338404	-1.30558569	-0.83445805	1.015297761	-0.80908783	1.0766711351
5	0.06905	0.0	2.18	0	0.4580	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	-0.412074053	-1.30558569	-0.83445805	1.227362043	-0.51067434	1.0766711351
6	0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7	-0.416631374	-1.30558569	-0.83445805	0.206891639	-0.35080997	1.0766711351
7	0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9	-0.409837246	-0.47618230	-0.26489191	-0.388026950	-0.07015919	0.8384142195
8	0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1	-0.403296561	-0.47618230	-0.26489191	-0.160306916	0.97784057	1.0236248974
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5	-0.395543302	-0.47618230	-0.26489191	-0.930285282	1.11638970	1.0861216287
10	0.17004	12.5	7.87	0	0.5240	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9	-0.400333140	-0.47618230	-0.26489191	-0.399412952	0.61548134	1.3283202075
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0	-0.393956378	-0.47618230	-0.26489191	0.131459378	0.91389483	1.2117799501
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9	-0.406444832	-0.47618230	-0.26489191	-0.392296701	0.50890509	1.1547920492
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7	-0.409198989	-0.47618230	-0.26489191	-0.563086727	-1.05066066	0.7863652700
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4	-0.346886928	-0.43682573	-0.14407485	-0.477691714	-0.24068118	0.4333252240
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2	-0.345933611	-0.43682573	-0.14407485	-0.268473932	0.56574575	0.3168899868
16	0.62739	0.0	8.14	0	0.5380	5.834	56.5	4.4986	4	307	21.0	395.62	8.47	19.9	-0.347162460	-0.43682573	-0.14407485	-0.641365488	-0.42896588	0.3341187865
17	1.05393	0.0	8.14	0	0.5380	5.935	29.3	4.4986	4	307	21.0	386.85	6.58	23.1	-0.297573695	-0.43682573	-0.14407485	-0.497617217	-1.39525719	0.3341187865
18	0.78420	0.0	8.14	0	0.5380	5.990	81.7	4.2579	4	307	21.0	386.75	14.67	17.5	-0.328932014	-0.43682573	-0.14407485	-0.419338455	0.46627459	0.2198105553

Showing 1 to 19 of 506 entries