De Excel a R : : **CHEAT SHEET**

Lectura de Datos

Fecha (Lunes)	Tipo SEM	Inversión	Impresiones	Clicks
02/01/2017	Brand	7677	3152	98
02/01/2017	Generic	10320	3699	196
09/01/2017	Brand	7648	3889	84
09/01/2017	Generic	9255	3726	219
...	...	...	...	...

DESDE EXCEL

df ← readxl::**read_excel**("excel.xlsx","hoja")

Opciones avanzadas:
- skip: salta líneas iniciales
- range: especifica un rango concreto de lectura

readxl::**excel_sheets**("excel.xlsx")
Devuelve todas las hojas del excel

DESDE CSV

df ← readr::**read_csv**("datos.csv")
Lee csv en formato inglés: separador coma, decimal punto

df ← readr::**read_csv2**("datos.csv")
Lee csv en formato español: separador punto y coma, decimal coma

NOMBRES DE LAS COLUMNAS

Para no tener problemas, es importante tratar los nombres de las columnas

Cambia espacios por barras (_), elimina signos de puntuación

Quita los acentos y ñ de los nombres de las columnas

```
df %>%
janitor::clean_names() %>%
Conento::quita_acentos_nombres()
```

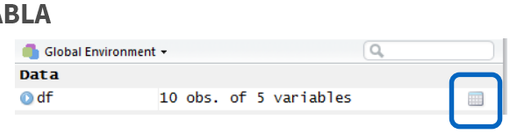
ENCADENAR OPERACIONES

Todas las funciones de esta hoja se pueden encadenar a través del operador **PIPE (%>%)**

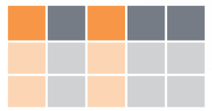
Atajo de teclado:
CTRL + MAYUS + M

INSPECCIONAR UNA TABLA

df %>% **View**



Seleccionar columnas



```
df %>%
dplyr::select(fecha_lunes, inversion)
```

Muestra sólo las columnas indicadas.
select(-tipo_sem**)** Muestra todas las columnas menos tipo_sem

SELECTER HELPERS

Funciones de ayuda para usar dentro de select( )

contains("*grps*")	Variables cuyo nombre contiene *grps*
starts_with("*tv*")	Variables cuyo nombre empieza por *tv*
ends_with("*brand*")	Variables cuyo nombre termina por *brand*
matches("*campana_\\d+*")	Variables cuyo nombre cumple una expresión regular

Filtros

```
df %>%
dplyr::filter(tipo_sem == "Brand", impresiones > 3500)
```

Muestra sólo las filas que cumplan los criteros establecidos.
Funciones lógicas que se pueden usar con filter:

== (igualdad)
!= (desigualdad)
<, >,<=,>=
columna %>% between(a,b) (columna entre a y b)
is.na(x) (valores missing)
!is.na(x) (valores no missing)
& (y lógico)
| (ó lógico(

Quitar duplicados

```
df %>%
dplyr::distinct()
```

Elimina filas duplicadas. Sin argumentos, quita duplicados puros

```
df %>%
dplyr::distinct(tipo_sem)
```

→

tipo_sem
Brand
Generic

Elimina valores duplicados de las columnas especificadas

Ordenar

```
df %>%
dplyr::arrange(tipo_sem, -impresiones)
```

Se pueden ordenar en varios niveles, y de manera descendiente colocando un menos (–) delante de la variable

Transformar variables

```
df %>%
dplyr::mutate(ctr = clicks/impresiones)
```

Crea nuevas variables o modifica las existentes.

transmute(ctr = clicks/impresiones) — Crea la variable ctr y elimina las variables usadas

mutate_all(funs(...)) — Aplica las funciones de funs a todas las variables

df %>% mutate_all(funs(as(.,10))

mutate_at(vars(…),funs(…)) — Aplica las funciones de funs a las variables indicadas en vars

df %>% mutate_at(vars(contains("grps")),funs(as(.,10))

mutate_if(predicado,funs(…)) — Aplica las funciones de funs a las variables que cumplen el predicado

df %>% mutate_if(is.numeric,funs(as(.,10))

OPERACIONES CON FECHAS

```
df %>%
dplyr::mutate(fecha = lubridate::ymd(fecha))
```

Transforma fechas de tipo texto en tipo fecha. También **dmy, mdy…**

```
df %>%
dplyr::mutate(fecha_lunes = Conento::semanalizar(fecha))
```

Convierte cualquier fecha al lunes de su misma semana. El argumento *dia_semana* indica a que día de la semana hay que convertir la fecha (valor de 1 (lunes, por defecto) a 7 (domingo))

```
df %>%
Conento::semanal_mensual("fecha_lunes")
```

Convierte un dataframe semanal en mensual teniendo en cuenta la variable fecha indicada. Sólo devuelve las columnas numéricas.

```
df %>%
Conento::mensual_semanal("fecha_mes")
```

Reparte un dataframe mensual en semanal.

RStudio® is a trademark of RStudio, Inc. • CC BY Samuel Huerga• samuelhuerga87@gmail.com • Updated: 2017-10

Transformar variables

OPERACIONES COMUNES

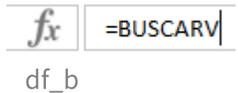
df %>%
dplyr::mutate(clicks_ad = Conento::**as**(clicks,10))

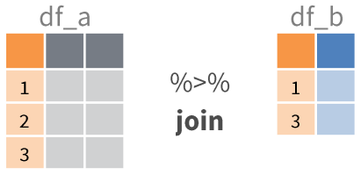
Aplica un adstock a una variable. Se puede especificar un número entre 0 y 1, o entre 0 y 100.

df %>%
dplyr::mutate(var = **ifelse**(condición, valor_si,valor_no))

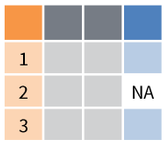
Condicional, igual que la función SI de excel.

Unir tablas





df_a %>%
dplyr::**left_join**(df_b)



Se unen las tablas por las variables que tengan en común. Conserva la tabla izquierda intacta y pega de la tabla derecha los valores que coincidan

df_a %>%
dplyr::**inner_join**(df_b)

Se unen las tablas por las variables que tengan en común. Conserva solos los valores que tienen una coincidencia en ambas tablas

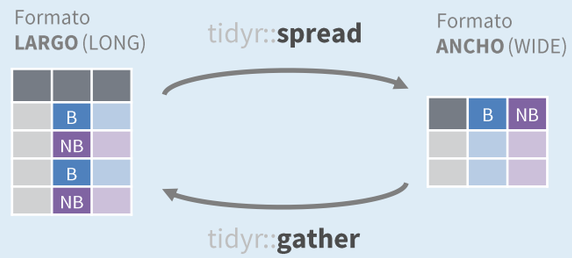
Texto en columnas

df %>%
tidyr::**separate**(columna, c("parte1","parte2") sep = ";")

Crea dos variables: parte1 y parte2, a partir de la variable columna separando el texto por punto y coma.

Tablas dinámicas

Para tratar los datos en una tabla dinámica, usaremos el formato largo:



Más sencillo para tratar los datos (agrupaciones)

Más sencillo para visualizar los datos (graficas)

Agrupaciones y agregaciones

df %>%
dplyr::**group_by**(tipo_sem) %>%
dplyr::**summarise**(impresiones = sum(impresiones))

Crea grupos de filas. Las operaciones que se hagan, respetarán estos grupos

Resume todos los elementos de un grupo a una única fila, aplicando una función de agregación

tipo_sem	impresiones
Brand	7041
Generic	7425

Las funciones de agregación mas comunes son:

sum()	Suma
mean()	Media
n()	Número de elementos
n_distinct()	Número de elementos distintos
median()	Mediana
sd()	Desviación típica
var()	Varianza
quantile()	Percentil
IQR()	Rango intercuartílico

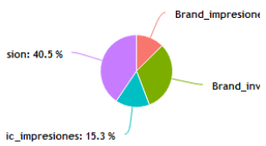
También se pueden usar las funciones **summarise_all**, **summarise_at** y **summarise_if**

Gráficos interactivos

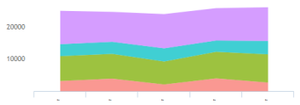
Seleccionar antes las columnas que se quieran dibujar. La primera columna siempre debe ser la fecha. Preferiblemente, el dataframe en formato ancho

df %>%
Conento::**hchart_lineas**()

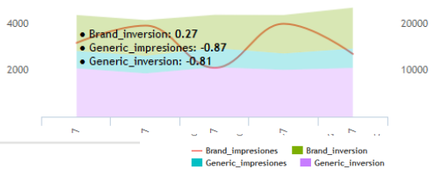
df %>%
Conento::**hchart_tartas**()



df %>%
Conento::**hchart_areas**()



df %>%
Conento::**descriptivos_n_variables**()



1 Filtrar valores usando dplyr::**filter**

2 Crear agrupaciones usando dplyr::**group_by**

3 Crear agregaciones usando dplyr::**summarise**

4 Añadir columna en la agrupación del paso 2, en dplyr::**group_by**

Después, transformar a formato WIDE usando tidyr::**spread**

5 En el caso de tener más de una columna en 3 o en 4, hay unificar todas las posibles combinaciones de variables. Lo tratamos en formato largo

fecha_lunes	tipo_sem	impresiones
02/01/2017	Brand	3152
02/01/2017	Generic	3699
09/01/2017	Brand	3889
09/01/2017	Generic	3726

→

fecha_lunes	Brand	Generic
02/01/2017	3152	3699
09/01/2017	3889	3726

df %>%
filter(impresiones > 100) %>%
group_by(fecha_lunes, tipo_sem) %>%
summarise(impresiones = sum(impresiones)) %>%
spread(tipo_sem, impresiones,fill = 0)

fecha_lunes	tipo_sem	inversion	impresiones
02/01/2017	Brand	7677	3152
02/01/2017	Generic	10320	3699
09/01/2017	Brand	7648	3889
09/01/2017	Generic	9255	3726

→

fecha_lunes	Brand_ impresiones	Brand_ inversion	Generic_ impresiones	Generic_ inversion
02/01/2017	3152	7677	3699	10320
09/01/2017	3889	7648	3726	9255

df %>%
filter(impresiones > 100) %>%
gather("nombre_variable","valor_variable",inversion,impresiones) %>%
group_by(fecha_lunes,tipo_sem,nombre_variable) %>%
summarise(sum = sum(valor_variable)) %>%
unite(nombre_variable,tipo_sem, nombre_variable) %>%
spread(nombre_variable,sum, fill = 0)

RStudio® is a trademark of RStudio, Inc. • CC BY Samuel Huerga • samuelhuerga87@gmail.com • Updated: 2017-10