Curso Herramientas de análisis con R

Clase 9: Un breve reporte

Alex Bajaña

2020/07/27 (Actualizado: 2020-07-27)

Contents


```
# Ingresar un nuevo chunk de codigo ctrl + alt+ I
```

Revisión

En esta clase vamos a ver la utilidad de las funciones `mutate`, `group_by` y `summarise` en colaboración con el resto de funciones aprendidas en la clase:

mutate()

```
.data %>%
  mutate(...)
```

- .data es un data frame o tibble
- ... asignaciones de varibales

Ejemplo:

```
tabla %>%
  mutate(pseudo_u = ventas_t - compras_t)

# Creo una variable que se llama "pseudo_u" que es función de compras y ventas
# dentro de mutate se crea un environment donde los objetos son las columnas de
# la tabla
```

group_by() %>% summarise()

`group by` crea grupos de acuerdo a una o más variables categoricas, en el caso de variables continuas se pueden crear factores con intervalos.

```
agrupada <- .data %>%
  group_by(...)
```

Mientras `summarise` toma una tabla agrupada y aplica una función de resumen.

```
agrupada %>%
  summarise( n = n())

# Devuelve una tibble o dataframe con: las columnas de agrupación y una nueva variable "n" que contiene

agrupada %>%
  summarise( n = n_distinct(variable))

# Sobre una variable número ica se puede pasar cualquier función como mínimo y maximo
```

Ejemplo:

Preambulo

Se inicia con la apertura del archivos

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts -------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# library(knitr)

# La versión de readr nos muestra que coerción hizo sobre los datos:

tabla <- read_csv("saiku-export.csv")
```

```
## Parsed with column specification:
## cols(
##   FAMILIA = col_character(),
##   PROVINCIA = col_character(),
##   `TIPO CONTRIBUYENTE` = col_character(),
##   `PERTENECE GRUPO ECONOMICO` = col_character(),
##   `GRAN CONTRIBUYENTE` = col_character(),
##   `CLASE CONTRIBUYENTE` = col_character(),
##   `ANIO FISCAL` = col_double(),
##   `MES FISCAL` = col_double(),
##   `ESTADO CONTRIBUYENTE` = col_character(),
##   `TOTAL COMPRAS LOCALES E IMPORTAC (519)` = col_double(),
##   `TOTAL VENTAS Y EXPORTACIONES (419)` = col_double(),
```

```
##   `IMPUESTO CAUSADO (601)` = col_double()
## )
```

```
# Además guarda los nombres tal como aparecen en la base de datos
```

Recordando los principios de la tidy data debemos corregir los nombres de las variables. Sin embargo en toda transformación de datos se debe procurar mantener tambien el principio de reproducibilidad. Es por ello que siempre debemos mantener un registro de nuestas transformaciones en orden de que si alguien desea replicar la transformación de datos va a obtener el mismo resultado.

```
# Guardo los nombres originales de la tabla

originales <- names(tabla)

# Creo un vector con nuevos nomrbes:

nuevos <- c("activ","provin","tipo_c","grupo_e",
                  "gran_c","clase","anio","mes","estado",
                  "compras_t","ventas_t","impuesto_c")

# Genero una equivalencia que me servira de guía

names(tabla) <- nuevos

tibble(`Nombres originales` = originales,
       `Nuevos nombres` = nuevos) %>%
  mutate(`Nombres originales` = str_to_sentence(`Nombres originales`))
```

```
## # A tibble: 12 x 2
##    `Nombres originales`                   `Nuevos nombres`
##    <chr>                                  <chr>
##  1 Familia                                activ
##  2 Provincia                              provin
##  3 Tipo contribuyente                     tipo_c
##  4 Pertenece grupo economico              grupo_e
##  5 Gran contribuyente                     gran_c
##  6 Clase contribuyente                    clase
##  7 Anio fiscal                            anio
##  8 Mes fiscal                             mes
##  9 Estado contribuyente                   estado
## 10 Total compras locales e importac (519) compras_t
## 11 Total ventas y exportaciones (419)     ventas_t
## 12 Impuesto causado (601)                 impuesto_c
```

Ahora vamos a realizar algunas transformaciones para llegar a un set de datos con el que realziar nuestro análisis:

```
tabla <- tabla %>%
  mutate(fecha = str_c(anio,"-",mes,"-01"),  # Pego las variables anio, mes
         fecha = as.Date(fecha)
         )             # Transformo a fecha
```

Ejercicio:

1. Genero una variable que se llame región natural

Hint: Utilizar la función `which` para encontrar las posiciones de las provincias que pertenecen a cada región.

Empleando **R-base:**

```
provincias <- unique(tabla$provin)

sierra <- provincias[c(1,2,3,6,10,14,15,16,17,20)]
costa <- provincias[c(4,5,7,8,11,12,18)]
oriente <- provincias[c(9,13,21,22,23,24)]
insular <- provincias[19]

tabla$region <- NA_character_

tabla$region[which(tabla$provin %in% sierra)] <- "Sierra"
tabla$region[which(tabla$provin %in% costa)] <- "Costa"
tabla$region[which(tabla$provin %in% oriente)] <- "Oriente"
tabla$region[which(tabla$provin %in% insular)] <- "Insular"

table(tabla$region)
```

```
##
##   Costa Insular Oriente  Sierra
##   51137    4377   29385   68032
```

Empleando **dplyr:**

```
tabla <- tabla %>%
  mutate(region_2 = case_when(provin %in% sierra ~ "Sierra",
                              provin %in% costa ~ "Costa",
                              provin %in% oriente ~ "Oriente",
                              TRUE               ~ "Insular"
                              ) )

table(tabla$region_2,useNA = "ifany")
```

```
##
##   Costa Insular Oriente  Sierra
##   51137    4377   29385   68032
```

2. Agrego las compras por fecha y región natural

Hint: Utilizar la función `aggregate` para encontrar las sumas agrupadas por fecha y region natural.

Empleando **R-base:**

```
resumen_1 <- aggregate(formula = compras_t ~ fecha + region,
          data = tabla,
          FUN = function(x) sum(x)/10e6)

resumen_1
```

```
##          fecha  region  compras_t
## 1   2015-01-01   Costa 453.175698
## 2   2015-02-01   Costa 471.826917
## 3   2015-03-01   Costa 517.316663
## 4   2015-04-01   Costa 496.249797
## 5   2015-05-01   Costa 500.786871
## 6   2015-06-01   Costa 577.292494
## 7   2015-07-01   Costa 500.107189
```

```
## 8   2015-08-01   Costa 482.333247
## 9   2015-09-01   Costa 517.031129
## 10  2015-10-01   Costa 518.518075
## 11  2015-11-01   Costa 511.616949
## 12  2015-12-01   Costa 716.888535
## 13  2016-01-01   Costa 401.669279
## 14  2016-02-01   Costa 417.678476
## 15  2016-03-01   Costa 446.557208
## 16  2016-04-01   Costa 434.644572
## 17  2016-05-01   Costa 503.258008
## 18  2016-06-01   Costa 520.992226
## 19  2016-07-01   Costa 445.628165
## 20  2016-08-01   Costa 480.803947
## 21  2016-09-01   Costa 484.080290
## 22  2016-10-01   Costa 497.455630
## 23  2016-11-01   Costa 521.873539
## 24  2016-12-01   Costa 765.952015
## 25  2017-01-01   Costa 438.576665
## 26  2017-02-01   Costa 426.714503
## 27  2017-03-01   Costa 507.138461
## 28  2017-04-01   Costa 468.008379
## 29  2017-05-01   Costa 521.748683
## 30  2017-06-01   Costa 607.588898
## 31  2017-07-01   Costa 488.324899
## 32  2017-08-01   Costa 529.081988
## 33  2017-09-01   Costa 509.875471
## 34  2017-10-01   Costa 558.292962
## 35  2017-11-01   Costa 574.100990
## 36  2017-12-01   Costa 786.777349
## 37  2015-01-01 Insular   2.137529
## 38  2015-02-01 Insular   2.174500
## 39  2015-03-01 Insular   2.426115
## 40  2015-04-01 Insular   3.730898
## 41  2015-05-01 Insular   2.178398
## 42  2015-06-01 Insular   2.792656
## 43  2015-07-01 Insular   2.655913
## 44  2015-08-01 Insular   2.562802
## 45  2015-09-01 Insular   2.344817
## 46  2015-10-01 Insular   3.534839
## 47  2015-11-01 Insular   2.686240
## 48  2015-12-01 Insular   3.750230
## 49  2016-01-01 Insular   2.025958
## 50  2016-02-01 Insular   2.457288
## 51  2016-03-01 Insular   2.510556
## 52  2016-04-01 Insular   2.223979
## 53  2016-05-01 Insular   2.872110
## 54  2016-06-01 Insular   2.481926
## 55  2016-07-01 Insular   2.165175
## 56  2016-08-01 Insular   2.291347
## 57  2016-09-01 Insular   4.347515
## 58  2016-10-01 Insular   2.413894
## 59  2016-11-01 Insular   2.950592
## 60  2016-12-01 Insular   3.806879
## 61  2017-01-01 Insular   2.030144
```

```
## 62  2017-02-01 Insular   2.024959
## 63  2017-03-01 Insular   2.467519
## 64  2017-04-01 Insular   2.169546
## 65  2017-05-01 Insular   3.234284
## 66  2017-06-01 Insular   2.883282
## 67  2017-07-01 Insular   2.426484
## 68  2017-08-01 Insular   2.526884
## 69  2017-09-01 Insular   2.420597
## 70  2017-10-01 Insular   3.089821
## 71  2017-11-01 Insular   2.662397
## 72  2017-12-01 Insular   4.185645
## 73  2015-01-01 Oriente  16.972038
## 74  2015-02-01 Oriente  17.750933
## 75  2015-03-01 Oriente  20.798613
## 76  2015-04-01 Oriente  19.221057
## 77  2015-05-01 Oriente  19.527439
## 78  2015-06-01 Oriente  25.233935
## 79  2015-07-01 Oriente  18.582015
## 80  2015-08-01 Oriente  19.035836
## 81  2015-09-01 Oriente  20.154028
## 82  2015-10-01 Oriente  18.743734
## 83  2015-11-01 Oriente  18.811977
## 84  2015-12-01 Oriente  31.130327
## 85  2016-01-01 Oriente  12.452713
## 86  2016-02-01 Oriente  13.722089
## 87  2016-03-01 Oriente  15.912722
## 88  2016-04-01 Oriente  15.765217
## 89  2016-05-01 Oriente  17.308443
## 90  2016-06-01 Oriente  18.411348
## 91  2016-07-01 Oriente  14.717342
## 92  2016-08-01 Oriente  16.862510
## 93  2016-09-01 Oriente  16.228880
## 94  2016-10-01 Oriente  17.531687
## 95  2016-11-01 Oriente  19.328501
## 96  2016-12-01 Oriente  30.571158
## 97  2017-01-01 Oriente  13.189182
## 98  2017-02-01 Oriente  14.917074
## 99  2017-03-01 Oriente  18.774449
## 100 2017-04-01 Oriente  17.243128
## 101 2017-05-01 Oriente  18.582315
## 102 2017-06-01 Oriente  24.818443
## 103 2017-07-01 Oriente  18.026492
## 104 2017-08-01 Oriente  19.830414
## 105 2017-09-01 Oriente  18.523838
## 106 2017-10-01 Oriente  20.222327
## 107 2017-11-01 Oriente  21.859887
## 108 2017-12-01 Oriente  35.286098
## 109 2015-01-01  Sierra 640.212553
## 110 2015-02-01  Sierra 650.736207
## 111 2015-03-01  Sierra 721.505267
## 112 2015-04-01  Sierra 707.787412
## 113 2015-05-01  Sierra 712.038293
## 114 2015-06-01  Sierra 766.945298
## 115 2015-07-01  Sierra 721.286108
```

```
## 116 2015-08-01  Sierra 641.162878
## 117 2015-09-01  Sierra 639.030333
## 118 2015-10-01  Sierra 704.472555
## 119 2015-11-01  Sierra 662.554346
## 120 2015-12-01  Sierra 931.328502
## 121 2016-01-01  Sierra 529.974756
## 122 2016-02-01  Sierra 504.276736
## 123 2016-03-01  Sierra 546.582848
## 124 2016-04-01  Sierra 593.910480
## 125 2016-05-01  Sierra 595.756449
## 126 2016-06-01  Sierra 626.393117
## 127 2016-07-01  Sierra 530.757057
## 128 2016-08-01  Sierra 590.550528
## 129 2016-09-01  Sierra 612.678890
## 130 2016-10-01  Sierra 618.080235
## 131 2016-11-01  Sierra 653.889157
## 132 2016-12-01  Sierra 931.147060
## 133 2017-01-01  Sierra 538.250357
## 134 2017-02-01  Sierra 536.923768
## 135 2017-03-01  Sierra 651.317159
## 136 2017-04-01  Sierra 599.257344
## 137 2017-05-01  Sierra 605.973288
## 138 2017-06-01  Sierra 781.651494
## 139 2017-07-01  Sierra 623.228932
## 140 2017-08-01  Sierra 635.148172
## 141 2017-09-01  Sierra 630.059543
## 142 2017-10-01  Sierra 654.765214
## 143 2017-11-01  Sierra 742.776224
## 144 2017-12-01  Sierra 938.379454
```

Empleando **dplyr:**

```
resumen_2 <- tabla %>%
  group_by(fecha,region_2) %>%
  summarise(suma_compras = sum(compras_t)/10e6)
```

```
## `summarise()` regrouping output by 'fecha' (override with `.groups` argument)
```

```
resumen_2
```

```
## # A tibble: 144 x 3
## # Groups:   fecha [36]
##    fecha      region_2 suma_compras
##    <date>     <chr>           <dbl>
##  1 2015-01-01 Costa          453.
##  2 2015-01-01 Insular          2.14
##  3 2015-01-01 Oriente         17.0
##  4 2015-01-01 Sierra         640.
##  5 2015-02-01 Costa          472.
##  6 2015-02-01 Insular          2.17
##  7 2015-02-01 Oriente         17.8
##  8 2015-02-01 Sierra         651.
##  9 2015-03-01 Costa          517.
## 10 2015-03-01 Insular          2.43
## # ... with 134 more rows
```

3. ¿Durante que mes entre marzo y mayo de 2016 tuvieron las compras más bajas las distintas regiones?

Hint: Filtro la tabla para las fechas señaladas, empleo la función `split` para guardar en una lista la información de cada una de las regiones. Con `lapply` y la función `which.min` hallo la el mes en que se presenta el minimo

Empleando **R-base:**

```
fechas <- as.Date(c("2016-03-01","2016-04-01","2016-05-01"))

ind_1 <- which(resumen_1$fecha %in% fechas)

resumen_1 <- resumen_1[ind_1,]


lista_1 <- split(resumen_1, resumen_1$region)

lapply(X = lista_1,
       FUN = function(tabla){
         y <- which.min(tabla$compras_t)
         tabla[y,]
       })
```

```
## $Costa
##         fecha region compras_t
## 16 2016-04-01  Costa  434.6446
##
## $Insular
##         fecha  region compras_t
## 52 2016-04-01 Insular  2.223979
##
## $Oriente
##         fecha  region compras_t
## 88 2016-04-01 Oriente  15.76522
##
## $Sierra
##          fecha region compras_t
## 123 2016-03-01 Sierra  546.5828
```

Empleando **dplyr:**

```
lista_2 <- resumen_2 %>%
  filter(between(fecha,as.Date("2016-03-01"),
                       as.Date("2016-05-01"))) %>%
  split(.$region_2)

# El minimo solo se puede sacar de un
lapply(lista_2,function(tabla){
  tabla %>%
    group_by(region_2) %>%
    filter(suma_compras == min(suma_compras))
})
```

```
## $Costa
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
```

```
## 1 2016-04-01 Costa            435.
##
## $Insular
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-04-01 Insular          2.22
##
## $Oriente
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-04-01 Oriente          15.8
##
## $Sierra
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-03-01 Sierra           547.
```

Todo el proceso en una sola secuencia:

```
read_csv("saiku-export.csv") %>%
 rename_all(~nuevos) %>%
 mutate(fecha = str_c(anio,"-",mes,"-01"),
        fecha = as.Date(fecha),
        region_2 = case_when(provin %in% sierra ~ "Sierra",
                             provin %in% costa ~ "Costa",
                             provin %in% oriente ~ "Oriente",
                             TRUE               ~ "Insular"
        )) %>%
 group_by(fecha,region_2) %>%
 summarise(suma_compras = sum(compras_t)/10e6) %>%
 ungroup %>%
 filter(between(fecha,as.Date("2016-03-01"),
                as.Date("2016-05-01"))) %>%
 split(.$region_2) %>%
 map(~.x %>%
       group_by(region_2) %>%
       filter(suma_compras == min(suma_compras)))
```

```
## Parsed with column specification:
## cols(
##   FAMILIA = col_character(),
##   PROVINCIA = col_character(),
##   `TIPO CONTRIBUYENTE` = col_character(),
##   `PERTENECE GRUPO ECONOMICO` = col_character(),
##   `GRAN CONTRIBUYENTE` = col_character(),
##   `CLASE CONTRIBUYENTE` = col_character(),
##   `ANIO FISCAL` = col_double(),
```

```
##   `MES FISCAL` = col_double(),
##   `ESTADO CONTRIBUYENTE` = col_character(),
##   `TOTAL COMPRAS LOCALES E IMPORTAC (519)` = col_double(),
##   `TOTAL VENTAS Y EXPORTACIONES (419)` = col_double(),
##   `IMPUESTO CAUSADO (601)` = col_double()
## )
```

```
## `summarise()` regrouping output by 'fecha' (override with `.groups` argument)
```

```
## $Costa
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-04-01 Costa            435.
##
## $Insular
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-04-01 Insular          2.22
##
## $Oriente
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-04-01 Oriente          15.8
##
## $Sierra
## # A tibble: 1 x 3
## # Groups:   region_2 [1]
##   fecha      region_2 suma_compras
##   <date>     <chr>           <dbl>
## 1 2016-03-01 Sierra           547.
```