Sesión 5

Curso: POL304 - Estadística para el análisis político 2

Jefes de práctica: Alexander Benites, Chiara Zamora y Airám Bello

Ciclo 2023-2

En las sesiones previas, hemos revisado diferentes estrategias de extracción y limpieza de datos. Hoy trabajaremos algunas funciones de manipulación de bases de datos con la librería *dplyr* y aplicaremos todo lo aprendido.

1. ¿Qué es dplyr?

Dplyr es un potente paquete de R para manipular, limpiar y resumir datos no estructurados. En resumen, hace que la exploración de datos y la manipulación de datos sea fácil y rápida en R.

¿Qué tiene de especial dplyr?

El paquete "dplyr" comprende muchas funciones que realizan las operaciones de manipulación de datos más utilizadas, como aplicar filtros, seleccionar columnas específicas, ordenar datos, añadir o eliminar columnas y agregar datos. Otra de las ventajas más importantes de este paquete es que es muy fácil aprender y utilizar las funciones de dplyr. También es fácil recordar estas funciones. Por ejemplo, filter() se utiliza para filtrar filas.

dplyr Function	Description	Equivalent SQL
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
summarise()	Summarise (or aggregate) data	-
arrange()	Sort the data	ORDER BY
join()	Joining data frames (tables)	JOIN
mutate()	Creating New Variables	COLUMN ALIAS

Empezamos!

```
library(rio)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mydata = import("https://raw.githubusercontent.com/deepanshu88/data/master/sampledata.csv")
```

2. Funciones básicas

La función *sample_n* selecciona filas aleatorias de un marco de datos (o tabla). El segundo parámetro de la función indica a R el número de filas que debe seleccionar. Pidamos cinco casos de forma aleatoria:

```
sample_n(mydata,5)
```

```
##   Index      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1     N New Mexico 1819239 1226057 1935991 1124400 1723493 1475985 1237704
## 2     W  Wisconsin 1788920 1518578 1289663 1436888 1251678 1721874 1980167
## 3     N     Nevada 1426117 1114500 1119707 1758830 1694526 1765826 1903270
## 4     G    Georgia 1929009 1541565 1810773 1779091 1326846 1223770 1773090
## 5     A    Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1820856 1801430 1653384 1475715 1623388 1533494 1868612
## 2 1901394 1648755 1940943 1729177 1510119 1701650 1846238
## 3 1231480 1526066 1143343 1980195 1283813 1225348 1903804
## 4 1630325 1145473 1851245 1850111 1887157 1259353 1725470
## 5 1944173 1237582 1440756 1186741 1852841 1558906 1916661
```

Con la función *sample_frac* solicitamos un porcentaje de casos del total de la data:

```
sample_frac(mydata,0.1)
```

```
##   Index      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1     W    Wyoming 1775190 1498098 1198212 1881688 1750527 1523124 1587602
## 2     K   Kentucky 1813878 1448846 1800760 1250524 1137913 1911227 1301848
## 3     O       Ohio 1802132 1648498 1441386 1670280 1534888 1314824 1516621
## 4     W  Wisconsin 1788920 1518578 1289663 1436888 1251678 1721874 1980167
## 5     N New Mexico 1819239 1226057 1935991 1124400 1723493 1475985 1237704
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1504455 1282142 1881814 1673668 1994022 1204029 1853858
## 2 1956681 1350895 1512894 1916616 1878271 1722762 1913350
## 3 1511460 1585465 1887714 1227303 1840898 1880804 1573117
## 4 1901394 1648755 1940943 1729177 1510119 1701650 1846238
## 5 1820856 1801430 1653384 1475715 1623388 1533494 1868612
```

Con la función *discinct* eliminamos casos repetidos. Podemos hacerlo en función a una variable. Tomemos como ejemplo la variable Index. *.keep_all* es necesario para quedarnos con todas las columnas.

```
x2 = distinct(mydata, Index, .keep_all= TRUE)
table(mydata$Index)
```

```
##
## A C D F G H I K L M N O P R S T U V W
## 4 3 2 1 1 1 4 2 1 8 8 3 1 1 2 2 1 2 4
```

```
table(x2$Index)
```

```
##
## A C D F G H I K L M N O P R S T U V W
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Ya conocemos la función *select*:

```
mydata2 = select(mydata, Index, State:Y2008) #Los : para indicar un rango de columnas.
```

Con el signo negativo dropeamos variables:

```
mydata = select(mydata, -Index, -State)
```

Con *start_with* nos quedamos con las variables que empiezan con alguna letra en específico:

```
mydata3 = select(mydata, starts_with("Y"))
#mydata33 = select(mydata, -starts_with("Y")) Así si quisieramos dropear las que empiezan con Y.
```

Otras funciones para seleccionar variables:

Helpers	Description
starts_with()	Starts with a prefix
ends_with()	Ends with a prefix
contains()	Contains a literal string
matches()	Matches a regular expression
num_range()	Numerical range like x01, x02, x03.
one_of()	Variables in character vector.
everything()	All variables.

Ejemplo rapido:

```
#Nos quedamos con las que acaban en 2:
mydata3 = select(mydata, ends_with("2"))

#O que contengan un elemento en específico:
mydata4 = select(mydata, contains("1"))
```

Cambiando nombres de variables:

```
mydata = import("https://raw.githubusercontent.com/deepanshu88/data/master/sampledata.csv")
names(mydata)
```

```
##  [1] "Index" "State" "Y2002" "Y2003" "Y2004" "Y2005" "Y2006" "Y2007" "Y2008"
## [10] "Y2009" "Y2010" "Y2011" "Y2012" "Y2013" "Y2014" "Y2015"
```

```r
mydata6 = mydata
colnames(mydata6)[1] ="Index1"
#colnames(mydata6) = c("Nombres de cada vector")
names(mydata6)
```

```
##  [1] "Index1" "State"  "Y2002"  "Y2003"  "Y2004"  "Y2005"  "Y2006"  "Y2007"
##  [9] "Y2008"  "Y2009"  "Y2010"  "Y2011"  "Y2012"  "Y2013"  "Y2014"  "Y2015"
```

Con *filter* filtramos la base:

```r
mydata7 = filter(mydata, Index == "A")
head(mydata7)
```

```
##   Index    State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1     A  Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
## 2     A   Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826
## 3     A  Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 4     A Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1944173 1237582 1440756 1186741 1852841 1558906 1916661
## 2 1436541 1629616 1230866 1512804 1985302 1580394 1979143
## 3 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 4 1628980 1669295 1928238 1216675 1591896 1360959 1329341
```

El operador *%in%* se puede utilizar para seleccionar múltiples elementos. Aquí le estamos diciendo a R que seleccione filas 'A' y 'C' en la columna 'Índice'.

```r
mydata7 = filter(mydata6, Index1 %in% c("A", "C"))
head(mydata7)
```

```
##   Index1      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1      A    Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
## 2      A     Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826
## 3      A    Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 4      A   Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
## 5      C California 1685349 1675807 1889570 1480280 1735069 1812546 1487315
## 6      C   Colorado 1343824 1878473 1886149 1236697 1871471 1814218 1875146
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1944173 1237582 1440756 1186741 1852841 1558906 1916661
## 2 1436541 1629616 1230866 1512804 1985302 1580394 1979143
## 3 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 4 1628980 1669295 1928238 1216675 1591896 1360959 1329341
## 5 1663809 1624509 1639670 1921845 1156536 1388461 1644607
## 6 1752387 1913275 1665877 1491604 1178355 1383978 1330736
```

Otra alternativa:

```r
mydata7 = filter(mydata6, Index1  == "A" | Index1 == "C") #Recordamos EAP 1 y los operadores lógicos
head(mydata7)
```

```
##   Index1      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1      A    Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
## 2      A     Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826
## 3      A    Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 4      A   Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
## 5      C California 1685349 1675807 1889570 1480280 1735069 1812546 1487315
## 6      C   Colorado 1343824 1878473 1886149 1236697 1871471 1814218 1875146
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1944173 1237582 1440756 1186741 1852841 1558906 1916661
## 2 1436541 1629616 1230866 1512804 1985302 1580394 1979143
## 3 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 4 1628980 1669295 1928238 1216675 1591896 1360959 1329341
## 5 1663809 1624509 1639670 1921845 1156536 1388461 1644607
## 6 1752387 1913275 1665877 1491604 1178355 1383978 1330736
```

Podemos aplicar más condiciones en la misma línea de código. Queremos a los casos A y C en la variable Index **Y** a los casos mayores a 1300000 en el año 2002:

```
mydata8 = filter(mydata6, Index1 %in% c("A", "C") & Y2002 >= 1300000 )
head(mydata8) #Veamo:
```

```
##   Index1       State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1      A     Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 2      A    Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
## 3      C  California 1685349 1675807 1889570 1480280 1735069 1812546 1487315
## 4      C    Colorado 1343824 1878473 1886149 1236697 1871471 1814218 1875146
## 5      C Connecticut 1610512 1232844 1181949 1518933 1841266 1976976 1764457
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 2 1628980 1669295 1928238 1216675 1591896 1360959 1329341
## 3 1663809 1624509 1639670 1921845 1156536 1388461 1644607
## 4 1752387 1913275 1665877 1491604 1178355 1383978 1330736
## 5 1972730 1968730 1945524 1228529 1582249 1503156 1718072
```

Ahora queremos a los casos A y C en la variable Index **O** los casos mayores a 1300000 en el año 2002:

```
mydata9 = filter(mydata6, Index1 %in% c("A", "C") | Y2002 >= 1300000)
tail(mydata9) #Nótese que hay casos que no son ni A ni C!!
```

```
##    Index1         State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 39      T         Texas 1520591 1310777 1957713 1907326 1873544 1655483 1785986
## 40      U          Utah 1771096 1195861 1979395 1241662 1437456 1859416 1939284
## 41      W    Washington 1977749 1687136 1199490 1163092 1334864 1621989 1545621
## 42      W West Virginia 1677347 1380662 1176100 1888948 1922085 1740826 1238174
## 43      W     Wisconsin 1788920 1518578 1289663 1436888 1251678 1721874 1980167
## 44      W       Wyoming 1775190 1498098 1198212 1881688 1750527 1523124 1587602
##      Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 39 1827503 1447457 1978374 1882532 1698698 1646508 1705322
## 40 1915865 1619186 1288285 1108281 1123353 1801019 1729273
## 41 1555554 1179331 1150089 1775787 1273834 1387428 1377341
## 42 1539322 1539603 1872519 1462137 1683127 1204344 1198791
## 43 1901394 1648755 1940943 1729177 1510119 1701650 1846238
## 44 1504455 1282142 1881814 1673668 1994022 1204029 1853858
```

Y cuando NO queremos algunos casos de la variable:

```
mydata10 = filter(mydata6, !Index1 %in% c("A", "C"))
head(mydata10)
```

```
##   Index1                State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007
## 1      D             Delaware 1330403 1268673 1706751 1403759 1441351 1300836
## 2      D District of Columbia 1111437 1993741 1374643 1827949 1803852 1595981
## 3      F              Florida 1964626 1468852 1419738 1362787 1339608 1278550
## 4      G              Georgia 1929009 1541565 1810773 1779091 1326846 1223770
## 5      H               Hawaii 1461570 1200280 1213993 1245931 1459383 1430465
## 6      I                Idaho 1353210 1438538 1739154 1541015 1122387 1772050
##     Y2008   Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1762096 1553585 1370984 1318669 1984027 1671279 1803169 1627508
## 2 1193245 1739748 1707823 1353449 1979708 1912654 1782169 1410183
## 3 1756185 1818438 1198403 1497051 1131928 1107448 1407784 1170389
## 4 1773090 1630325 1145473 1851245 1850111 1887157 1259353 1725470
## 5 1919423 1928416 1330509 1902816 1695126 1517184 1948108 1150882
## 6 1335481 1748608 1436809 1456340 1643855 1312561 1713718 1757171
```

Vamos a utilizar la función de filtro con la función *grepl* para buscar a todos aquellos países que tengan "Ar" en el nombre:

```
mydata10 = filter(mydata6, grepl("Ar", State))
head(mydata10)
```

```
##   Index1    State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1      A  Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 2      A Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 2 1628980 1669295 1928238 1216675 1591896 1360959 1329341
```

Ahora podemos hacer cálculos con los vectores con la función *summarise*:

```
summarise(mydata, Y2015_mean = mean(Y2015),
                  Y2015_med=median(Y2015))
```

```
##   Y2015_mean Y2015_med
## 1    1588297   1627508
```

Con *summarise_at* podemos solicitar varias medidas de tendencia central para varias variables así:

```
summarise_at(mydata, vars(Y2005, Y2006), funs(n(), mean, median))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
```

```
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

##   Y2005_n Y2006_n Y2005_mean Y2006_mean Y2005_median Y2006_median
## 1      51      51    1522064    1530969      1480280      1531641
```

Otra forma:

```
summarise_at(mydata, vars(Y2005, Y2006), list(~n(), ~mean(.), ~median(.)))
```

```
##   Y2005_n Y2006_n Y2005_mean Y2006_mean Y2005_median Y2006_median
## 1      51      51    1522064    1530969      1480280      1531641
```

3. Pipe operator %>%

El operador pipe se utiliza en *dplyr* prestado de la librería *magrittr* y permite concatenar funciones en una línea de código:

Por ejemplo, nos vamos a quedar solo con dos variables y, sobre esa selección, sacamos una muestra de diez casos:

```
dt = mydata %>% select(Index, State) %>% sample_n(10)
head(dt)
```

```
##   Index        State
## 1     M     Missouri
## 2     F      Florida
## 3     R Rhode Island
## 4     N North Dakota
## 5     M      Montana
## 6     U         Utah
```

Así podemos manipular de muchas formas nuestra base original sin moverla realmente. Otro ejemplo:

```
dt1 = mydata %>%
  select(Index,Y2002,Y2008,Y2010) %>%
  group_by(Index) %>%
  summarise_at(vars(Y2002,Y2008,Y2010), funs(n(), mean, median)) %>%
  filter(Y2002_mean >= 1501744)

#Y así hasta el infinito

head(dt1)
```

```
## # A tibble: 6 x 10
##   Index Y2002_n Y2008_n Y2010_n Y2002_mean Y2008_mean Y2010_mean Y2002_median
```

```
##   <chr>   <int>   <int>   <int>      <dbl>      <dbl>      <dbl>        <dbl>
## 1 C           3       3       3   1546562.   1708973.   1835505.     1610512
## 2 F           1       1       1   1964626    1756185    1198403      1964626
## 3 G           1       1       1   1929009    1773090    1145473      1929009
## 4 I           4       4       4   1534438.   1416165    1652330.     1503812.
## 5 K           2       2       2   1661466    1625663    1450664      1661466
## 6 L           1       1       1   1584734    1185085    1498662      1584734
## # ... with 2 more variables: Y2008_median <dbl>, Y2010_median <dbl>
```

Función *do()*: Solicitamos algo a la data

```
t = mydata %>% filter(Index %in% c("A", "C","I"))
head(t)
```

```
##   Index      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1     A    Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
## 2     A     Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826
## 3     A    Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 4     A   Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
## 5     C California 1685349 1675807 1889570 1480280 1735069 1812546 1487315
## 6     C   Colorado 1343824 1878473 1886149 1236697 1871471 1814218 1875146
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1944173 1237582 1440756 1186741 1852841 1558906 1916661
## 2 1436541 1629616 1230866 1512804 1985302 1580394 1979143
## 3 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 4 1628980 1669295 1928238 1216675 1591896 1360959 1329341
## 5 1663809 1624509 1639670 1921845 1156536 1388461 1644607
## 6 1752387 1913275 1665877 1491604 1178355 1383978 1330736
```

Le pedimos que nos dé los dos primeros casos de cada grupo:

```
t = mydata %>% filter(Index %in% c("A", "C","I")) %>% group_by(Index) %>%
  do(head( . , 2))
head(t)
```

```
## # A tibble: 6 x 16
## # Groups:   Index [3]
##   Index State      Y2002  Y2003  Y2004  Y2005  Y2006  Y2007  Y2008  Y2009  Y2010
##   <chr> <chr>      <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
## 1 A     Alabama   1.30e6 1.32e6 1.12e6 1.49e6 1.11e6 1.44e6 1.95e6 1.94e6 1.24e6
## 2 A     Alaska    1.17e6 1.96e6 1.82e6 1.45e6 1.86e6 1.47e6 1.55e6 1.44e6 1.63e6
## 3 C     Californ~ 1.69e6 1.68e6 1.89e6 1.48e6 1.74e6 1.81e6 1.49e6 1.66e6 1.62e6
## 4 C     Colorado  1.34e6 1.88e6 1.89e6 1.24e6 1.87e6 1.81e6 1.88e6 1.75e6 1.91e6
## 5 I     Idaho     1.35e6 1.44e6 1.74e6 1.54e6 1.12e6 1.77e6 1.34e6 1.75e6 1.44e6
## 6 I     Illinois  1.51e6 1.53e6 1.49e6 1.26e6 1.54e6 1.75e6 1.87e6 1.66e6 1.42e6
## # ... with 5 more variables: Y2011 <int>, Y2012 <int>, Y2013 <int>,
## #   Y2014 <int>, Y2015 <int>
```

Lo interesante es que podemos pedirle que se quede no solo con los primeros casos en términos de posición, sino también con el tercer caso más alto de la variable, por ejemplo:

```
t = mydata %>% select(Index, Y2015) %>% #Selecciona variables
  filter(Index %in% c("A", "C","I")) %>% #Filtra por categorías
  group_by(Index) %>% #Agrupa
  do(arrange(.,desc(Y2015))) %>%  slice(3) #Ordena y se queda con el tercer valor más alto

head(t)
```

```
## # A tibble: 3 x 2
## # Groups:   Index [3]
##   Index   Y2015
##   <chr>   <int>
## 1 A     1647724
## 2 C     1330736
## 3 I     1583516
```

Otro ejemplo de concatenación de funciones:

```
t = mydata %>%
  group_by(Index)%>%
  summarise(Mean_2014 = mean(Y2014, na.rm=TRUE),
            Mean_2015 = mean(Y2015, na.rm=TRUE)) %>%
  arrange(desc(Mean_2015))

head(t)
```

```
## # A tibble: 6 x 3
##   Index Mean_2014 Mean_2015
##   <chr>     <dbl>     <dbl>
## 1 U      1801019   1729273
## 2 G      1259353   1725470
## 3 A      1506531.  1718217.
## 4 M      1596816.  1710808.
## 5 V      1494748.  1708159
## 6 P      1931500   1668232
```

4. Reshape

Otro conjunto de funciones están en la librería *reshape*. La data con la que estamos trabajando está limpia, pero no necesariamente ordenada. Es decir, muchas veces, algunos gráficos requieren de cierto tipo de estructura de datos. Si quisieramos hacer un gráfico de líneas, en la que cada línea es un año, este tipo de esturctura de datos no nos ayudaría. Reshape nos puede ayudar con esto:

Esta es nuestra data:

```
head(mydata)
```

```
##   Index      State   Y2002   Y2003   Y2004   Y2005   Y2006   Y2007   Y2008
## 1     A    Alabama 1296530 1317711 1118631 1492583 1107408 1440134 1945229
## 2     A     Alaska 1170302 1960378 1818085 1447852 1861639 1465841 1551826
## 3     A    Arizona 1742027 1968140 1377583 1782199 1102568 1109382 1752886
## 4     A   Arkansas 1485531 1994927 1119299 1947979 1669191 1801213 1188104
```

```
## 5     C California 1685349 1675807 1889570 1480280 1735069 1812546 1487315
## 6     C   Colorado 1343824 1878473 1886149 1236697 1871471 1814218 1875146
##     Y2009   Y2010   Y2011   Y2012   Y2013   Y2014   Y2015
## 1 1944173 1237582 1440756 1186741 1852841 1558906 1916661
## 2 1436541 1629616 1230866 1512804 1985302 1580394 1979143
## 3 1554330 1300521 1130709 1907284 1363279 1525866 1647724
## 4 1628980 1669295 1928238 1216675 1591896 1360959 1329341
## 5 1663809 1624509 1639670 1921845 1156536 1388461 1644607
## 6 1752387 1913275 1665877 1491604 1178355 1383978 1330736
```

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
rdata <- melt(mydata, id=c("Index","State"))
head(rdata)
```

```
##   Index      State variable   value
## 1     A    Alabama    Y2002 1296530
## 2     A     Alaska    Y2002 1170302
## 3     A    Arizona    Y2002 1742027
## 4     A   Arkansas    Y2002 1485531
## 5     C California    Y2002 1685349
## 6     C   Colorado    Y2002 1343824
```

Podemos renombrar las columnas:

```
colnames(rdata)[3] = "Year"
```

Esa Y al inicio de cada año puede ser fastidiosa. Felizmente podemos eliminarla con *subtr*:

```
rdata$Year = substr(rdata$Year,2,8)
head(rdata)
```

```
##   Index      State Year   value
## 1     A    Alabama 2002 1296530
## 2     A     Alaska 2002 1170302
## 3     A    Arizona 2002 1742027
## 4     A   Arkansas 2002 1485531
## 5     C California 2002 1685349
## 6     C   Colorado 2002 1343824
```

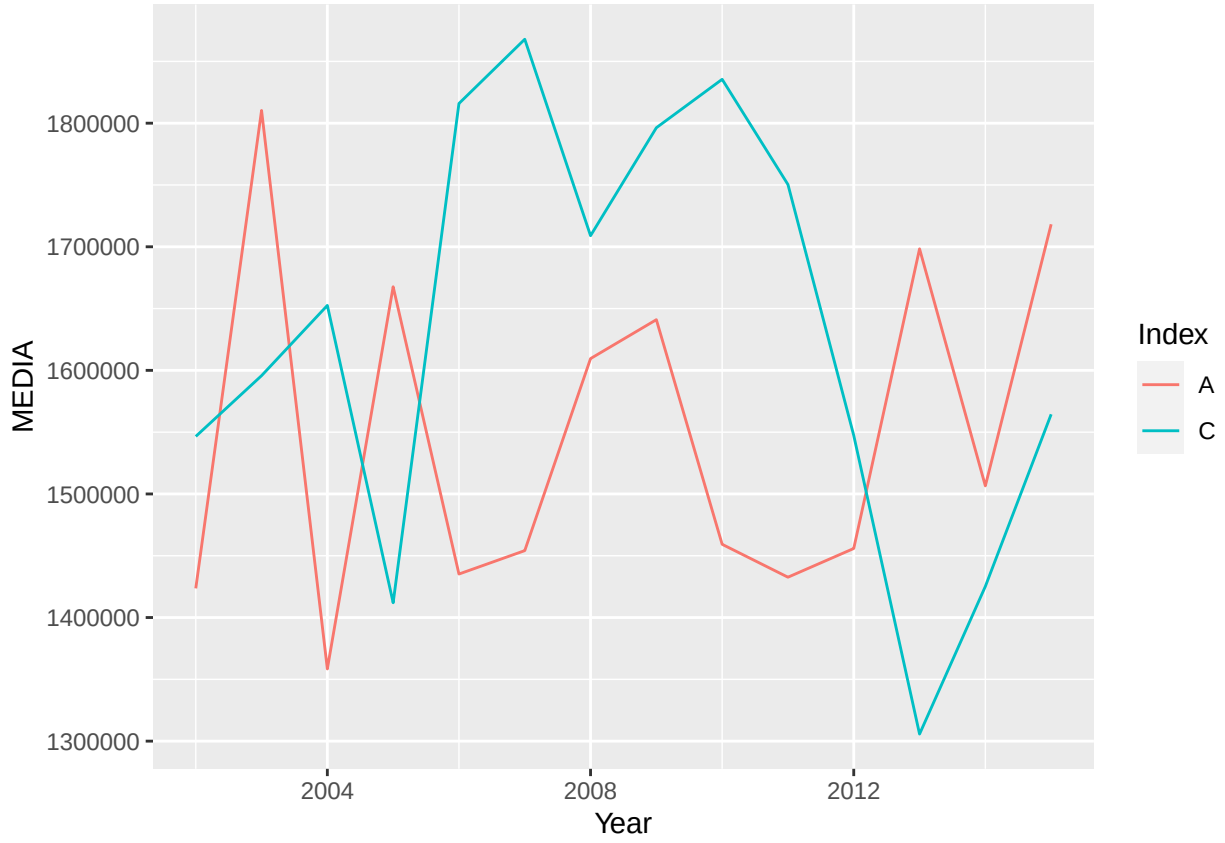
Con esto ya podríamos graficar. Agrupemos por índice:

```
rdata = rdata %>%
  group_by(Index,Year) %>%
  summarise(MEDIA = mean(value, na.rm = T)) %>%
  filter(Index %in% c("A","C"))
```

```
## `summarise()` has grouped output by 'Index'. You can override using the
## `.groups` argument.
```

```
rdata$Year = as.numeric(rdata$Year)
```

```
library(ggplot2)
ggplot(rdata, aes(x = Year, y = MEDIA, colour = Index)) +
  geom_line()
```



5. Toupper:

Cambiemos las cosas a mayúsculas:

```
a <- "No se duerman"
a
```

```
## [1] "No se duerman"
```

```r
toupper(a)
```

```
## [1] "NO SE DUERMAN"
```

```r
mydata$State1 = toupper(mydata$State)
table(mydata$State1)
```

```
##
##              ALABAMA               ALASKA              ARIZONA
##                    1                    1                    1
##             ARKANSAS           CALIFORNIA             COLORADO
##                    1                    1                    1
##          CONNECTICUT             DELAWARE DISTRICT OF COLUMBIA
##                    1                    1                    1
##              FLORIDA              GEORGIA               HAWAII
##                    1                    1                    1
##                IDAHO             ILLINOIS              INDIANA
##                    1                    1                    1
##                 IOWA               KANSAS             KENTUCKY
##                    1                    1                    1
##            LOUISIANA                MAINE             MARYLAND
##                    1                    1                    1
##        MASSACHUSETTS             MICHIGAN            MINNESOTA
##                    1                    1                    1
##          MISSISSIPPI             MISSOURI              MONTANA
##                    1                    1                    1
##             NEBRASKA               NEVADA        NEW HAMPSHIRE
##                    1                    1                    1
##           NEW JERSEY           NEW MEXICO             NEW YORK
##                    1                    1                    1
##       NORTH CAROLINA         NORTH DAKOTA                 OHIO
##                    1                    1                    1
##             OKLAHOMA               OREGON         PENNSYLVANIA
##                    1                    1                    1
##         RHODE ISLAND       SOUTH CAROLINA         SOUTH DAKOTA
##                    1                    1                    1
##            TENNESSEE                TEXAS                 UTAH
##                    1                    1                    1
##              VERMONT             VIRGINIA           WASHINGTON
##                    1                    1                    1
##        WEST VIRGINIA            WISCONSIN              WYOMING
##                    1                    1                    1
```

Para minúsculas:

```r
mydata$State2 = tolower(mydata$State1)
table(mydata$State2)
```

```
##
##              alabama               alaska              arizona
##                    1                    1                    1
```

```
##             arkansas           california             colorado
##                    1                    1                    1
##          connecticut             delaware district of columbia
##                    1                    1                    1
##              florida              georgia               hawaii
##                    1                    1                    1
##                idaho             illinois              indiana
##                    1                    1                    1
##                 iowa               kansas             kentucky
##                    1                    1                    1
##            louisiana                maine             maryland
##                    1                    1                    1
##        massachusetts             michigan            minnesota
##                    1                    1                    1
##          mississippi             missouri              montana
##                    1                    1                    1
##             nebraska               nevada        new hampshire
##                    1                    1                    1
##           new jersey           new mexico             new york
##                    1                    1                    1
##       north carolina         north dakota                 ohio
##                    1                    1                    1
##             oklahoma               oregon         pennsylvania
##                    1                    1                    1
##         rhode island       south carolina         south dakota
##                    1                    1                    1
##            tennessee                texas                 utah
##                    1                    1                    1
##              vermont             virginia           washington
##                    1                    1                    1
##        west virginia            wisconsin              wyoming
##                    1                    1                    1
```

Si queremos bajar a minúsculas desde el segundo elemento de la línea de texto:

```
mydata$State3=stringr::str_to_title(mydata$State1)
table(mydata$State3)
```

```
##
##              Alabama               Alaska              Arizona
##                    1                    1                    1
##             Arkansas           California             Colorado
##                    1                    1                    1
##          Connecticut             Delaware District Of Columbia
##                    1                    1                    1
##              Florida              Georgia               Hawaii
##                    1                    1                    1
##                Idaho             Illinois              Indiana
##                    1                    1                    1
##                 Iowa               Kansas             Kentucky
##                    1                    1                    1
##            Louisiana                Maine             Maryland
##                    1                    1                    1
```

```
##        Massachusetts             Michigan            Minnesota
##                    1                    1                    1
##          Mississippi             Missouri              Montana
##                    1                    1                    1
##             Nebraska               Nevada        New Hampshire
##                    1                    1                    1
##           New Jersey           New Mexico             New York
##                    1                    1                    1
##       North Carolina         North Dakota                 Ohio
##                    1                    1                    1
##             Oklahoma               Oregon         Pennsylvania
##                    1                    1                    1
##         Rhode Island       South Carolina         South Dakota
##                    1                    1                    1
##            Tennessee                Texas                 Utah
##                    1                    1                    1
##              Vermont             Virginia           Washington
##                    1                    1                    1
##        West Virginia            Wisconsin              Wyoming
##                    1                    1                    1
```

6. Ejercicio de limpieza final:

- Data de ERM a nivel distrital
- Nos quedamos con el ausentismo
- Con el ganador y con el segundo puesto
- Agregamos UBIGEOS
- Cruzamos UBIGEOS de INEI con RENIEC
- Vamos a sacar índice de competitividad: diferencia entre el primer y el segundo
- Cruzamos con variables de CEPLAN

```
ERM = import("https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/DISTRITAL.xlsx")
UBI_JNE = import("https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/UBI.xlsx")
EQUIV = import("https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/Equivalencias.xlsx")
CEPLAN = import("https://github.com/Alexanderbenit7/EAP2_2023-2/raw/main/data/ceplan.xlsx")
```

```
## New names:
## * `` -> `...1`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...17`
```

```
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...23`
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...26`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...29`
## * `` -> `...30`
## * `` -> `...31`
## * `` -> `...32`
## * `` -> `...33`
## * `` -> `...34`
## * `` -> `...35`
## * `` -> `...36`
## * `` -> `...37`
## * `` -> `...38`
## * `` -> `...39`
## * `` -> `...40`
## * `` -> `...41`
## * `` -> `...42`
## * `` -> `...43`
## * `` -> `...44`
## * `` -> `...45`
## * `` -> `...46`
## * `` -> `...47`
## * `` -> `...48`
```

6.1. Ausentismo a nivel distrital: Sacamos el ausentismo a nivel distrital:

```
AUSEN = select(ERM, c(1:3,5,12))
AUSEN$AUSENTISMO = 1-AUSEN$`% Participación`
```

Preparemos los ubigeos de RENIEC y agregamos:

```
UBI_JNE$UNIF = paste0(UBI_JNE$REGION,UBI_JNE$PROVINCIA,UBI_JNE$DISTRITO)
UBI_JNE = select(UBI_JNE, c(1,12))
```

Agregamos vector de unificación en la base de ausentismo:

```
AUSEN$UNIF = paste0(AUSEN$Region, AUSEN$Provincia, AUSEN$Distrito)
AUSEN = select(AUSEN, c(1:3,5:7))
```

Juntamos información:

```
AUSEN = merge(AUSEN, UBI_JNE, by = "UNIF", all.x = T)
AUSEN = select(AUSEN, -c(1))
```

```r
#Lo que perdimos:
sum(is.na(AUSEN$UBIGEO))
```

```
## [1] 629
```

```r
perdidos = AUSEN[is.na(AUSEN$UBIGEO),]
table(perdidos$Distrito)
```

```
##
##            ALEXANDER VON HUMBOLDT                          ANCHIHUAY
##                                12                                 10
##                        ANDAYMARCA ANDRES AVELINO CACERES DORREGARAY
##                                 7                                 27
##                           CANAYRE                  CASTILLO GRANDE
##                                17                                 16
##                             CHACA                       CHANGUILLO
##                                 5                                 22
##                       CONSTITUCION                           COSME
##                                27                                 15
##                             CUENCA                       EL INGENIO
##                                18                                 19
##                        EL PORVENIR                         IHUAYLLO
##                                 6                                 13
##                           INKAWASI               JOSE MARIA ARGUEDAS
##                                 7                                  6
##                         LA MORADA               LA YARADA LOS PALOS
##                                10                                 10
##                        LOS CHANKAS                          MARCONA
##                                 6                                 21
##                          MEGANTONI                          MI PERU
##                                 6                                 13
##                            NESHUYA                         ORONCCOY
##                                14                                  7
##                             PAMPAS                           PICHOS
##                                21                                  5
##                          PUCACOLPA                         PUCAYACU
##                                 9                                 10
##                       PUEBLO NUEVO                         QUICHUAS
##                                12                                  8
##                             ROBLE                         ROCCHACC
##                                 5                                  5
##                       ROSA PANDURO                         SAMUGARI
##                                 8                                 17
##                        SAN MIGUEL               SAN PABLO DE PILLAO
##                                14                                 10
##                SAN PEDRO DE CASTA               SAN PEDRO DE LARAOS
##                                36                                  4
##      SANTA ROSA DE ALTO YANAJANCA                SANTIAGO DE TUCUMA
##                                 9                                  5
##             SANTO DOMINGO DE ANDA           TENIENTE MANUEL CLAVERO
##                                11                                  8
##                       UCHURACCAY             VEINTISEIS DE OCTUBRE
##                                 9                                 39
```

```
##                   VILLA KINTIARINA                       VILLA VIRGEN
##                                  6                                  8
##                      VISTA ALEGRE                   VIZCATAN DEL ENE
##                                 20                                 12
##                             YACUS                             YAGUAS
##                                 17                                  7
```

```
ERM= ERM[complete.cases(ERM$`Tipo Organización Política`),] #Para solo quedarmos con org. políticas
```

6.2. Ganador del Municipio local: Se gana con mayoría simple, así que nos quedamos con el valor más alto:

```
GANADOR = ERM %>%
  filter(YEAR == 2018) %>%
  group_by(Region,Provincia,Distrito) %>%
  summarise(Ganador = max(`% Votos`, na.rm = T))
```

```
## `summarise()` has grouped output by 'Region', 'Provincia'. You can override
## using the `.groups` argument.
```

Agregamos UBIGEO:

```
GANADOR$UNIF = paste0(GANADOR$Region,GANADOR$Provincia, GANADOR$Distrito)
GANADOR = merge(GANADOR, UBI_JNE, by = "UNIF", all.x = T)
GANADOR = select(GANADOR, c(5:6))
```

6.2. El que se queda en segundo puesto: La función *do()* es crucial determinante:

```
SEGUNDO = ERM %>%
  filter(YEAR == 2018) %>%
  group_by(Region,Provincia,Distrito) %>%
   do(arrange(.,desc(`% Votos`))) %>%  slice(2)
```

Seleccionamos las variables de interés:

```
SEGUNDO$UNIF = paste0(SEGUNDO$Region, SEGUNDO$Provincia, SEGUNDO$Distrito)
SEGUNDO = merge(SEGUNDO, UBI_JNE, by = "UNIF", all.x = T)
SEGUNDO= select(SEGUNDO, c(12,14))
colnames(SEGUNDO) = c("Segundo","UBIGEO")
```

6.3. Variables de CEPLAN:

```
AUSEN = filter(AUSEN,YEAR == 2018)
AUSEN=AUSEN[!duplicated(AUSEN), ]
```

```
FINAL_DATA = merge(AUSEN, GANADOR, by = "UBIGEO")
FINAL_DATA = merge(FINAL_DATA, SEGUNDO, by = "UBIGEO")
FINAL_DATA = FINAL_DATA[complete.cases(FINAL_DATA$UBIGEO),]
```

6.4. Juntamos data política: Juntamos UBIGEO INEI:

```
EQUIV = select(EQUIV, c(1,2))
```

```
FINAL_DATA = merge(FINAL_DATA, EQUIV, by.x = "UBIGEO", by.y = "UBIGEO_RENIEC")
```

Agregamos data de CEPLAN:

```
FINAL_DATA = merge(FINAL_DATA, CEPLAN, by.x = "UBIGEO_INEI", by.y = "UBIGEO")
```

Agregamos el nivel de competitividad:

```
FINAL_DATA$COMPETITIVIDAD = FINAL_DATA$Ganador-FINAL_DATA$Segundo
FINAL_DATA$REG=stringr::str_to_title(FINAL_DATA$Region)
FINAL_DATA$PROV=stringr::str_to_title(FINAL_DATA$Provincia)
FINAL_DATA$DIST=stringr::str_to_title(FINAL_DATA$Distrito)
```

Data final:

```
FINAL_DATA = select(FINAL_DATA, c(1,14:16,7,10:13)) %>%
  filter(COMPETITIVIDAD>=0)
head(FINAL_DATA)
```

```
##   UBIGEO_INEI      REG        PROV        DIST AUSENTISMO  IDH_2019 %POBREZA
## 1      010102 Amazonas Chachapoyas    Asuncion 0.07291667 0.4230319 36.51995
## 2      010103 Amazonas Chachapoyas      Balsas 0.21047431 0.3153084 45.73296
## 3      010104 Amazonas Chachapoyas       Cheto 0.12234043 0.3457458 39.16978
## 4      010105 Amazonas Chachapoyas   Chiliquin 0.21617852 0.2750378 53.04566
## 5      010106 Amazonas Chachapoyas Chuquibamba 0.28255034 0.2692697 51.60783
## 6      010107 Amazonas Chachapoyas     Granada 0.14251781 0.3580015 43.86596
##   %POBREZAEXT COMPETITIVIDAD
## 1   15.680750        0.15612
## 2   15.427120        0.12449
## 3   23.678410        0.06222
## 4   36.395370        0.11909
## 5   43.788830        0.09100
## 6    5.943345        0.01776
```

Y ya para irnos:

```
sub_data = select(FINAL_DATA, c(5:9))
```

```
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(sub_data)
```