Dplyr

Rodrigo Negrete Pérez

January 26, 2022

Section 1

Intro al Tidyverse

Paquetes

- Una de las ventajas de R: al ser gratis, hay muchos paquetes con funciones especializadas.
- A menudo, autores publican sus métodos de estimación como paquetes.
- Antes de hacer algo sofisticado, revisa antes si no existe ya un paquete que lo hace.
    - Por ejemplo, muchos loops se pueden hacer con la familia de funciones "apply"

Tidyverse

- Tidyverse es una familia de paquetes estadísticos.
- Vamos a ver “dplyr” -> Data pliers.
- Dedicaremos otras sesión a ““ggplot””.

Instalación de paquetes

- Para todos los paquetes: primero debemos INSTALARLOS UNA SOLA VEZ.
- Encerramos el nombre del paquete entre comillas

```
install.packages('tidyverse')
```

- Antes de usarlos, CARGARLOS CADA VEZ que abramos sesión.

```
library(tidyverse)
```

- Cuando lo cargamos con library() lo escribimos sin comillas.

Section 2

Lectura de bases de datos

Directorio

- Antes de leer bases de datos, necesitamos hablar un poco de los directorios.
- Hay muchas maneras para leer bases de datos:
  - Una para cada formato: excel, stata, csv, etc.
- Pero en todas tenemos que especificar dónde está el archivo: el path

Checar directorio

- podemos verificar el directorio con wd()

```
getwd()
```

```
## [1] "C:/Users/rodri/OneDrive - INSTITUTO TECNOLOGICO AUTONO
```

- Por default:
  - R va a buscar archivos en esta carpeta
  - También va a subir archivos a esta carpeta

Ejemplo

- Usaremos la base de ministros de Jacob Nyrup y Stuart Bramwell (Oxford) presentaron una gran base de datos en su paper Who governs? A New Global Dataset on Members of Cabinets (2020).
- La usarán mucho en MPA con Adrián
- La pueden descargar del Github
  - RodrigoNP/Labs_MIA
- Primero debemos descargar el archivo en nuestra carpeta de elección.

Modificando directorio

- Si elegimos modificar el directorio usamos setwd() y ponemos dentro el path de nuestra carpeta.

```
setwd('C:/Users/rodri/OneDrive - INSTITUTO TECNOLOGICO AUTONOM
```

- La última diagonal es la última carpeta en la que está.
- **Observa que es un forward slash**

- Para abrir un archivo excel necesitamos hacer uso de otro paquete: readxl, dentro del tidyverse.
- Una vez modificado el directorio, solo ponemos el nombre y formato del archivo entre como texto (entre comillas)

```
min<-readxl::read_excel('WhoGov_within_V1.1.xlsx',
guess_max = 99999)
```

- Fíjate en cómo nombro la función y que guardo el df como una variable
- El guess_max es para que no ignore columnas en las que haya muchos NA

Sin modificar directorio

- Si no modificamos el directorio, tenemmos que poner todo el path y el nombre del archivo

```
min<- readxl:: read_excel('C:/Users/rodri/OneDrive - INSTITUTO
```

- Puedes o no poner el readxl::
- Al hacer esto, llamamos la función read_excel() del paquete readxl

NA's

- Todas las bases de datos tiene NA.
- Denotan celdas que están vacías, no tienen dato. NO SON 0.
- Podemos especificar qué valores fungen como NA al descargar la base de datos, añadiendo na=c()

Section 3

Data Pliers

Data Pliers

- El paquete dplyr tiene una sintaxis sencilla.
- Ayuda a tratar filas como una observación.

Pipe (pipa)

- El operador más común es el “pipe”, %>%
- Lo podemos escribir fácilmente con Ctrl+ shift+ m
- Quiere decir: *a los datos de la izquierda (pipa) le voy a aplicar la función de la derecha*.
- no obstante, podemos omitir la pipa si ponemos la varable del df como primer argumento de la función

mutate()

- Hay muchas funciones, mencionemos las más usadas.
- mutate() añade una variable (columna).
- Su atractivo es que podemos utilizar operadores lógicos que relacionen variables dentro de una observación.

- Por ejemplo, añadamos una dummy que denote si un ministro es una mujer.

```
min<- min %>%
  mutate(
    female=ifelse(gender=='Female',1,0)
  )
```

- O podríamos poner

```
min<-mutate(min,gender=ifelse(gender=='Female',1,0))
```

filter()

- filter() ayuda a quedarnos con observaciones que cumplan con ciertos criterios
- Se quedan las observaciones que produzcan un TRUE dentro de los paréntesis.

- Por ejemplo, quedémonos solo con las ministras que estén dentro de core y guardémoslo como un nuevo df

```
min_fem.core<- min %>% filter(gender=='Female'
        & core==1)
```

select()

- select() hace lo mismo que filter(), pero con variables (columnas).

Concatenar funciones

- Podemos concatenar funciones de dplyr añadiendo pipas.
- Por ejemplo, podemos hacer lo mismo que anteriormente, pero en un solo paso.

```
min_fem.core<- min %>%
  mutate(
    female=ifelse(gender=='Female',1,0)
  ) %>% filter(gender=='Female'
                & core==1)
```

Otros verbos

Algunas otras funciones notables son

- **rename()** para renombrar variables
- **arrange()** para reordenar observaciones
- **sample_n()** para seleccionar n observaciones aleatoriamente
- **sample_frac()** para seleccionar un porcentaje de observaciones aleatoriamente.
- **replace_na()** para reemplazar NA's

Section 4

Agrupar datos

group_by()

A veces las observaciones pertenecen a una o más categorías.

- Por ejemplo, los datos panel están agrupados por entidad y por tiempo.

R puede agrupar fácilmente con group_by()

- Después de agrupar podemos añadir variables o comprimir bases de datos.
- Podemos agrupar por más de una categoría separando por una coma.

- Por ejemplo, veamos las proporciones de mujeres en el gabinete por año. Después, añadamos una columna que especifique ese promedio para el año en curso en la base de datos
- Para ello, tenemos que agrupar por año, y luego añadir la columna.

```
min<- min %>%
  group_by(year) %>%
  mutate(
    fem_mean.year=mean(female,
                       na.rm=T)
  )
```

na.rm=T

- ¿Cómo manejamos los NA?
- Hay funciones que ignoran los NA
- Generalmente el argumento que incertamos es **na.rm=T**

summarise()

- Usamos summarise() si queremos comprimir la base de datos.
- La base de datos se modifica totalmente
- Las nuevas observaciones pasan a ser el/los argumento(s) por los que agrupamos.
- Solo tendremos las variables que especifiquemos dentro del summarise()

Veamos la proporción de mujeres por año, pero ahora comprimamos la base de datos.

```
min_grouped<- min %>%
  group_by(year) %>%
  summarise(
    fem_mean.year=mean(female,
                       na.rm=T)
  )
```

```
head(min_grouped)
```

```
##   year fem_mean.year
## 1 1963    0.00000000
## 2 1964    0.00000000
## 3 1965    0.00000000
## 4 1966    0.01383327
## 5 1967    0.01585546
## 6 1968    0.01488744
```

Section 5

Otras funciones

Menciono otras funciones interesantes que pueden llegar a ser útiles al manejar datos y en conjunto con dplyr

case_when()

- Hay veces que queremos hacer un ifelse con varias condiciones
- concatenar un ifelse() se vuelve progresivamente difícil

case_when(condicion_1 ~ qué pone R si se cumple,

condicion_2 ~ ” “,

...

TRUE ~ qué poner R e.o.c.)

- Esta es de las funciones más contraintuitivas, hagamos un ejemplo
- Supongamos que queremos añadir una columna que indique si en un país año hay un porcentaje deficiente, moderado, equitativo o elevado de mujeres, de tal manera que
  - menor a 15%, deficiente
  - entre 15% y menor a 40%, moderado
  - entre 40% y 55%, equitativo
  - más de 55%, elevado
- Primero hay que agrupar

```
min_equitativo<-min %>%
  group_by(year, country_isocode) %>%
  mutate(
    fem_mean.year.country=mean(female,
                               na.rm=T),
    equit=case_when(
      fem_mean.year.country<.15~'deficiente',

      fem_mean.year.country>=.15
      & fem_mean.year.country<.4 ~ 'moderado',

      fem_mean.year.country>=.4
      & fem_mean.year.country<.55 ~ 'equitativo',

      fem_mean.year.country>55 ~ 'elevado',

      T ~ 'no aplica'
    ))
```

recode()

- Hay veces que queremos cambiar el nombre de los factores en un df
- **recode()** preserva las categorizaciones, pero alterando los nombres de los factores.

max()

- Podemos añadir un evento importante para todo un grupo utilizando la función **max()**

factor()

- Podemos fácilmente convertir una variable caracter o numérica en factor con **factor()**
- Hay veces que queremos que sea un factor y no solo texto, especialmente cuando grafiquemos.

Section 6

Merges

Merges

Hay veces que queremos juntas bases de datos con las mismas observaciones.

- Por ejemplo, nuestra base de ministros con datos del PIB.
- Hay muchos tipos de merges y una familia de funciones para todos los casos
- Evidentemente, las categorías deseadas tienen que estar **escritas idénticamente**.

tipo_join(x,y,

by=c('variable1.x'='variable1.y',

'variable2.x'='variable2.y'))

Tipos de merge

- **left_join()** A la base de la izquierda le añade lo de la derecha. Observaciones solo en la base derecha se pierden.
- **right_join()**
- **inner_join()** la intersección de las bases de datos
- **full_join()** la unión

Section 7

Ejercicios

Tomados de Lucardi de MPA

Con la base de datos de ministros

- Crea una variable factor que identifique décadas: del 78 al 89; del 90 al 99, del 2000 al 2009 y del 2010 al 2020. ¿Hay más ministras con el paso de las décadas?
- Crea una variable que indique el año en el que por primea vez una mujer fue líder del gobierno. Para esto:
  - Primero crea una variable que indique el año si el líder es mujer y 0 e.o.c.
  - Agrupa y utiliza la función max()

Merges

Muy a menudo las categorías por las que vamos a hacer un merge van a estar escritas de diferente manera. Con países, puede que cambien de nombre.

- Guarda un duplicado de la base de ministros, guardado con un nombre diferente.
- Cambia el nombre de Czechoslovakia a Czech Republic y United States a USA en esta nueva base. Puedes usar recode().
- Diseña un código que indique el vector de países que está escritos de distinta manera.