# Intro a R

Rodrigo Negrete Pérez

February 8, 2022

1. Idea de los Labs
2. Material de los labs
3. Estructura de los labs
4. ¿Qué es R?
5. Interfaz
6. Creación y tipos de objetos
7. Vectores

# 8 Ejercicios Mauricio Romero

Section 1

# Idea de los Labs

# Teoría y Práctica

El curso de MIA se centra en la pregunta de la causalidad. Han proliferado algunas estrategias de identificación: esenciales para el portafolio de cualquier científico social.

- Parte teórica: intuición de cómo funcionan las estrategias de identificación
- Parte práctica: aplicarlas a un conjunto de datos.

Peeero... tenemos que aprender R.

# Inspiración de los labs.

A grandes razgos, baso estos labs en tres fuentes/profesores:

- Mauricio Romero (Profesor de Economía del ITAM)
  - https://mauricio-romero.com/teaching/microeconometria-aplicada-otono-2021/
- Nick Huntington (Profesor en quien se apoyó Mauricio)
  - https://nickchk.com/econometrics.html#learnR
- Curso de MPA de Adrián Lucardi-> Eventualmente lo llevarán.

Animo a checar estas fuentes ante cualquier inquietud adicional.

Section 2

# Material de los labs

# Github

Todo el material de apoyo de los labs estará en Github.

- rodrigonp/Labs_MIA

Section 3

# Estructura de los labs

# Introducción a R

Una parte de los labs se enfocará en R.

- Nociones básicas del lenguaje: vectores, loops, funciones
- Análisis de datos (Dplyrs)
- Ggplot

Cualquiera que quiera aprender R puede asistir.

# R aplicado a MIA

La otra parte será aplicar R a MIA.

- Simulaciones
- Aplicación de estrategias de identificación
- Algunas nociones básicas para entender papers.

Estos labs se darán cuando se haya cubierto el material del curso pertinente.

Section 4

# ¿Qué es R?

# Lenguaje orientado a objetos

A grandes razgos, con R podemos:

- Crear objetos
- Manipular objetos
- Ver objetos

¿Qué es un objeto? En nuestro caso, todo lo relacionado a datos: variables, observaciones, regresiones, tablas de regresión.

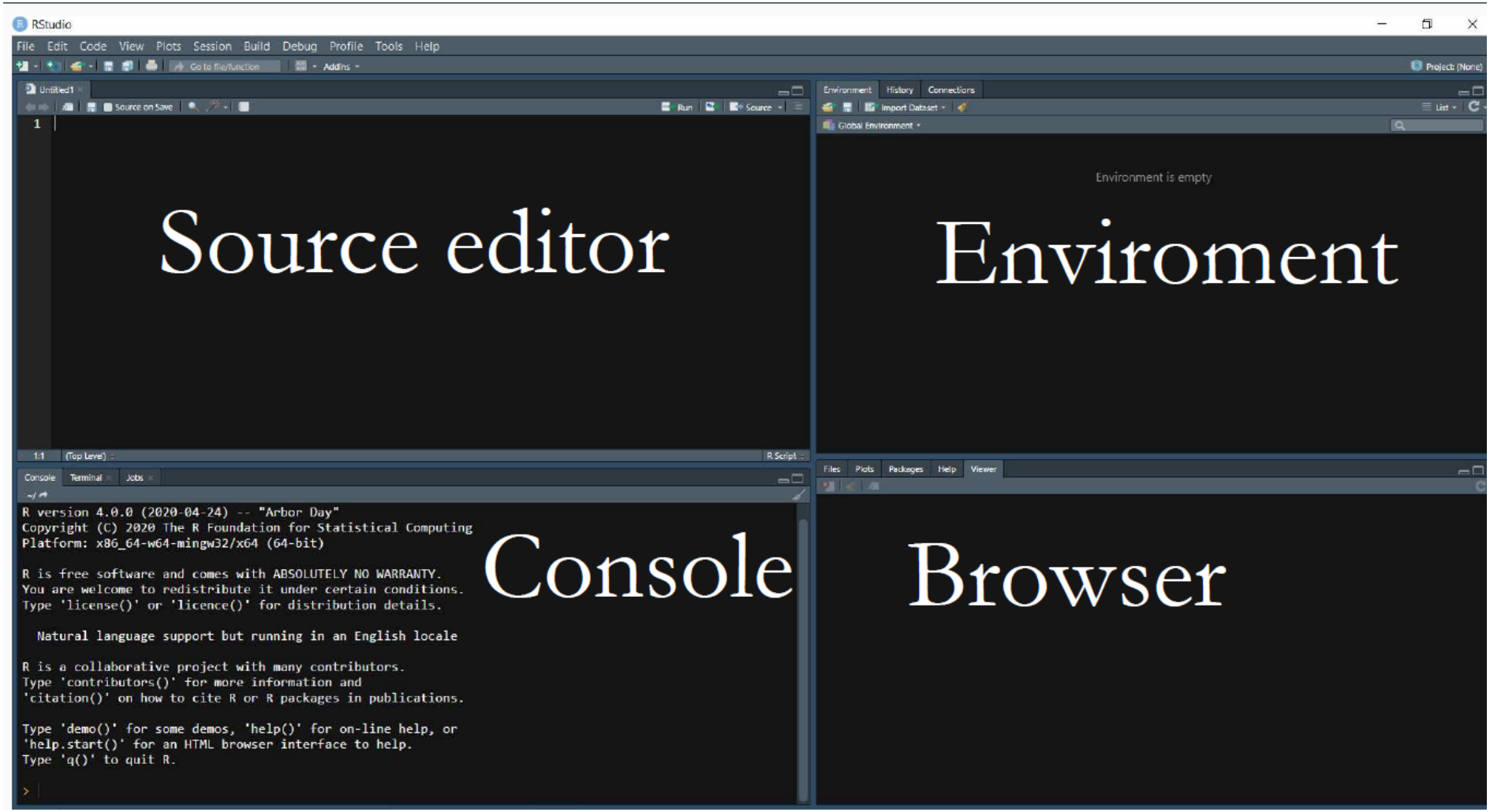
# ¿Por qué R?

- Gratis: todo mundo lo usa, hasta autores.
- Como con la gramática: aprendes uno y entiendes el resto (parecido a Python)
- Replicabilidad

Section 5

# Interfaz

# Interfaz


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source editor
Enviroment
Environment is empty
Console
R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Browser

# Consola

- Podemos ejecutar código aquí, sin embargo, no se guarda
- Muestra los errores en otro color (QUE SIEMPRE HAY QUE LEER)
- Muestras las advertencias (warnings).

# Errores vs Warnigns

- Los errores impiden que R corra el código
  - Dividimos sobre 0
  - Queremos sacar el promedio de una palabra
- Los Warnings son avisos: R pudo correr el código, pero hay algo sospechoso

# Environment

Destacan el environment y y el historial

- El historial es un registro del código corrido
- El environment es un registro de los objetos que hemos creado
- podemos borrarlo con rm(list=ls()) o con la escobita

# Navegador

- Podemos ver archivos y paquetes instalados
- PANEL DE AYUDA
- Gráficas

# Source

A pesar de que el código se puede ejecutar en la consola, aquí deberían trabajar el código. El código aquí se guarda: REPLICABILIDAD.

- Después de escribir, se puede ejecutar con ctrl+ enter, seleccionando previamente
  - o picando los botones arrriba en la pestaña del source, etc.
- R no ejecuta nada despues de #, así que se pueden hacer comentarios después de un #

# Ayuda

- R puede autocompletar nombre de variables, funciones, etc.
- Podemos buscar una funcion en el navegador y saldrá la documentación. O también podemos ejecutar el commando ?función
- GOOGLEEN: R se aprende más en StackOverflow que en cursos.

Section 6

# Creación y tipos de objetos

# Creación de objetos

- Creamos objetos con <- , = o -> Hay muchos tipos de objetos, veamos los más básicos

```
numeric.var<-1
character.var<-'Mexico'
factor.var<-factor(1, labels = 'one')
logic.var<- TRUE #Booleanos # TRUE= T, FALSE= F
```

- Los objetos creados se pueden ver en el environment

- Si los ejecutamos, R los muestra. Si solo quieres ver el objeto, conviene hacerlo en la consola para no tener que borrarlo otra vez

```
numeric.var
```

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

- Podemos preguntarle a R si un objeto es de algun tipo con is.tipo

```
is.numeric(numeric.var)
```

```
## [1] TRUE
```

- y podemos usar as.tipo para cambiar entre tipos de variables

```
as.character(numeric.var)
```

```
## [1] "1"
```

```
as.numeric(numeric.var)
```

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

```
as.factor(character.var)
```

```
## [1] Mexico
## Levels: Mexico
```

Modificar un objeto lo sobreescribe: NO GUARDA DOS VERSIONES, ANULA LA ANTERIOR Y SE QUEDA CON LA NUEVA

```
numeric.var
```

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

```
numeric.var<-3+sqrt(9)+9^4-4+9/7
numeric.var
```

```
## [1] 6564.286
```

- nótese el uso de los operadores algebráicos comunes.
- recordemos que la consola no guarda, así que ahí podemos hacer cálculos rápidos.

# Caracteres o strings

Son un trozo de texto encerrado entre comillas. Podemos usar dos comillas o el apóstrofe (que recomiendo porque es más rápido)

```
name<-'Rodrigo'
```

# Lógicos o Booleanos

Podemos preguntarle algo a R y nos contesta con un TRUE o FALSE

```
a<-59

a>100
```

```
## [1] FALSE
```

Como en casi todos los lenguajes, los operadores son:

- & para 'y', 'intersección de conjuntos'
- | para 'o inclusiva', 'unión de conjuntos'
- == 'igual a'
- >= 'mayor o igual a'

Algo útil es que TRUE=1 y FALSE=0

```
T+3
```

```
## [1] 4
```

# Práctica

```
is.logical(is.numeric(FALSE))
is.numeric(2)+is.character('hola')
T|F
T & F
```

```
## [1] TRUE
```

```
## [1] 2
```

```
## [1] TRUE
```

```
## [1] FALSE
```

# Factores

Son variables categóricas mutuamente excluyentes. Se ven como caracteres, pero tienen niveles, que son el número de categorías.

```
consolas<-as.factor('xbox')
```

podemos añadir niveles de una manera mas sencilla

```
levels(consolas)<-c('xbox','switch','ps5')
consolas
```

```
## [1] xbox
## Levels: xbox switch ps5
```

Section 7

# Vectores

# Vectores o listas

Podemos concatenar objetos usando **c( )**. De preferencia, que sean del mismo tipo. Un vector es una lista de objetos, una colección de objetos. Podemos saber su longitud usando **length()**

```
vector<-c(1,4,5,6)
vector<-c(vector, 1,4,6,8,9)
length(vector)
```

```
## [1] 9
```

# Slices (particiones)

Podemos llamar a partes específicas de los vectores utilizando paréntesis. Dentro de los paréntesis podemos especificar las posiciones deseadas o incluir un operador lógico

```
vector<-c(1,4,5,6)
vector[3]
```

```
## [1] 5
```

```
vector[vector<5]
```

```
## [1] 1 4
```

## :

Un operador habitual es ':' que se interpreta como de x a y.

```
vector<-c(1,4,5,6)
vector[1:3]
```

```
## [1] 1 4 5
```

También podemos hacer series

```
series<-1:10
series
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

# !

- Otro operador es '!' que se interpreta como el complemento

```
vector<-c(1,4,5,6,15,3,20)
vector[vector<5]
```

```
## [1] 1 4 3
```

```
vector[! vector<5]
```

```
## [1]  5  6 15 20
```

Muchas funciones utilizan vectores

```
mean(vector); sd(vector); prod(vector)
```

```
## [1] 7.714286
```

```
## [1] 7.016986
```

```
## [1] 108000
```

o puedes operar con los vectores mismos, y funciona como un vector matemático

```
r<-c(1,4,6,4,2,5,9)
r*2
```

```
## [1]  2  8 12  8  4 10 18
```

```
r+r
```

```
## [1]  2  8 12  8  4 10 18
```

```
r>=4
```

```
## [1] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

Como vimos, podemos utilizar factores para ver cuantas observaciones pertenecen a una categoría

```
carreras<-as.factor(c('eco','eco','cpol','ri','ri'))
table(carreras)
```

```
## carreras
## cpol  eco   ri
##    1    2    2
```

# %in%

Usamos el operador '%in%' para ver si un objeto pertenece a un vector. Denota si un objeto está dentro de un vector.

```
carreras<-c('eco','eco','cpol','ri','ri')
'mat' %in% carreras
```

```
## [1] FALSE
```

# Otras funciones

Para simulaciones seguramente usaremos rep() y sample (). rep() copia un vector y lo repite algunas veces.

- rep(*objeto a repetir*, *número de veces a repetir* )

```
rep(4,4)
```

```
## [1] 4 4 4 4
```

```
a<-c(1,7,9)
rep(a,3)
```

```
## [1] 1 7 9 1 7 9 1 7 9
```

Podemos hacer vectores de ceros o de texto vacío

```
numeric(5) #vector de 0 el numero de veces indicada
```

```
## [1] 0 0 0 0 0
```

```
character(6) # lo mismo, pero para characters
```

```
## [1] "" "" "" "" "" ""
```

sample() toma una muestra aleatoria. Se puede especificar el reemplazo

- sample( *objeto para tomar la muestra*, *tamaño de muestra*, replace= FALSE )

```
sample(1:10,3)
```

```
## [1]  2 10  9
```

```
a<-c(2,4,5,6,1,3,12,45,56)
sample(a,4)
```

```
## [1]  2  6  5 45
```

```
sample(a, 2, replace = F)
```

```
## [1]  2 12
```

# Menciones honorables

- sort() acomoda las entradas del vector según se especifique
- unique() da las entradas únicas en un vector con posibles repeticiones
- max()
- min()
- length() da la longitud del vector

# Particiones con vectores independientes

Podemos hacer particiones lógicas, incluso usando otro vector.

```
coin_toss<-sample(
  rep(c('aguila', 'sol')), 10 ,
  replace = T)

y<-1:10

y[coin_toss=='aguila']
```

```
## [1] 1 3 5 6 7 8 9
```

Section 8

# Ejercicios Mauricio Romero

# Ejericios tomados de Mauricio Romero

```
f<-c(2,7,5,1)
f^2
f + c(1,2,3,4)
c(f,6)
is.numeric(f)
mean(f >= 4)
f*c(1,2,3)
length(f)
length(rep(1:4,3))
f/2 == 2 | f < 3
as.character(f)
f[1]+f[4]
c(f,f,f,f)
f[f[1]]
f[c(1,3)]
f %in% (1:4*2)
```

```
## [1]  4 49 25  1
```

```
## [1] 3 9 8 5
```

```
## [1] 2 7 5 1 6
```

```
## [1] TRUE
```

```
## [1] 0.5
```

```
## Warning in f * c(1, 2, 3): longitud de objeto mayor no es m
## longitud de uno menor
```

```
## [1]  2 14 15  1
```

```
## [1] 4
```

```
## [1] 12
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

```
## [1] "2" "7" "5" "1"
```

```
## [1] 3
```

```
##  [1] 2 7 5 1 2 7 5 1 2 7 5 1 2 7 5 1
```

```
## [1] 7
```

```
## [1] 2 5
```

```
## [1]  TRUE FALSE FALSE FALSE
```

- Crea un factor que seleccione aleatoriamente entre seis personas que pueden ser hombre, mujer u otro género
- Averigua la suma acumulada entre 45 y 987 y súmale la media de todos estos números. (averigua la función para suma acumulada)
- Reacomoda h<-c(1,3,5,23,-4) de mayor a menor
- ¿Cuántos múltiplos de 4 hay entre 344 y 899? (averigua cómo sacar residuos no econométricos)