

NIDO DEL BÚHO PRESENTA:

# Estadística y Econometría Espacial con R, Módulo I

Clase 2: Herramientas básicas de R para la manipulación de datos espaciales



**PRESENTADO POR**  
Alex Bajaña | Pablo Sarango

---

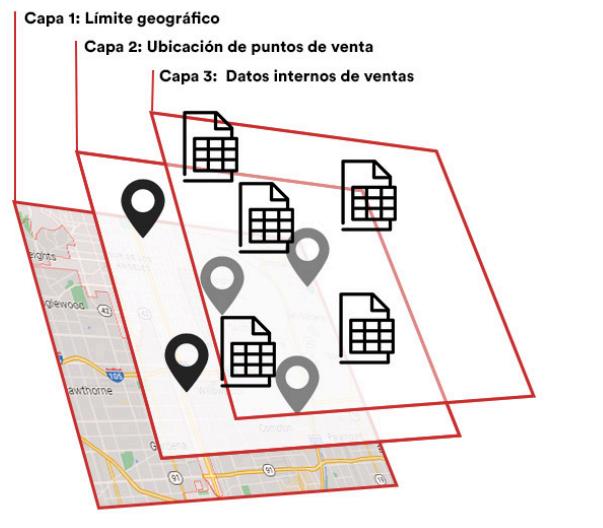
## Sesgos y datos



- Empleamos la modelación matemática para minimizar los *sesgos concientes o inconcientes*
- Sin embargo la interpretación de los datos y los resultados recaen en el ser humano
  - En un mundo ideal buscamos ser **objetivos**.
  - Sin embargo, la influencia del ser humano puede perpetuar desigualdades.
- Para responder la pregunta de investigación debemos aplicar **supuestos**, esto puede modificar de forma implícita la manera que recolectamos e interpretamos los datos.

# Importancia de los datos espaciales en la ciencia de datos

Los datos espaciales, también conocidos como **datos geoespaciales**, hacen referencia a la información que identifica la **ubicación geográfica** y las **características** de los elementos y límites naturales o construidos en la Tierra.



# Datos Espaciales

Estos datos suelen representarse en términos de **coordenadas cartesianas (x,y)** para mapas bidimensionales, pero también pueden incluir la **altitud (z)** para una representación tridimensional.

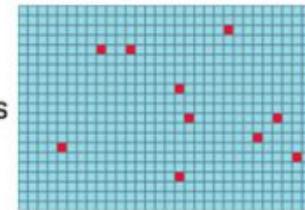
Los datos espaciales pueden adoptar diversas formas como:

- Puntos (GPS)
- Líneas (carreteras o ríos)
- Polígonos (fronteras, zonas de uso del suelo).

Datos Vectoriales

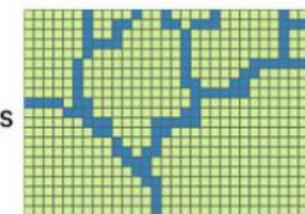


Datos Ráster



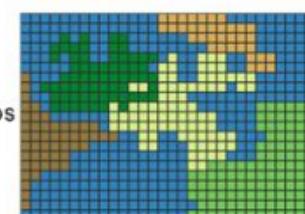
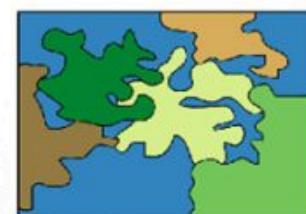
Puntos

Líneas



Líneas

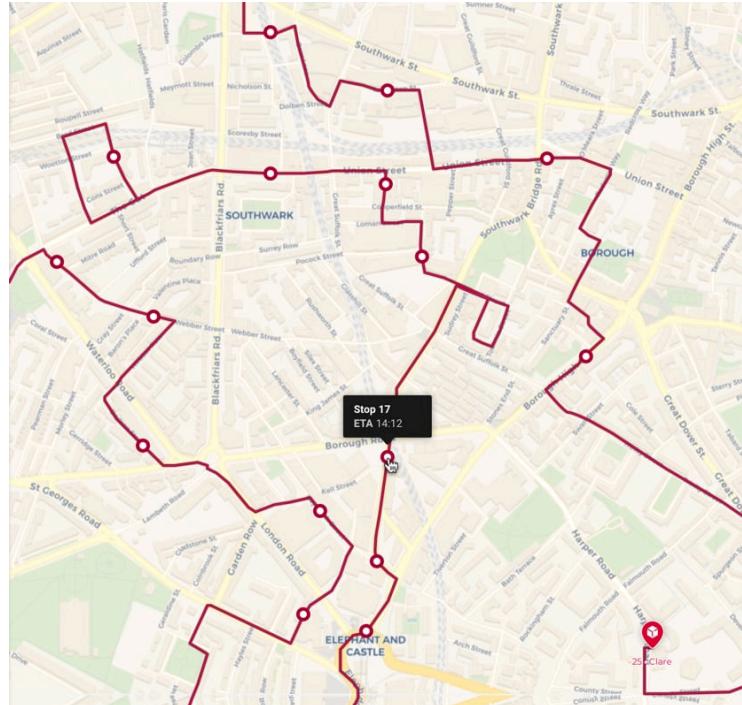
Polígonos



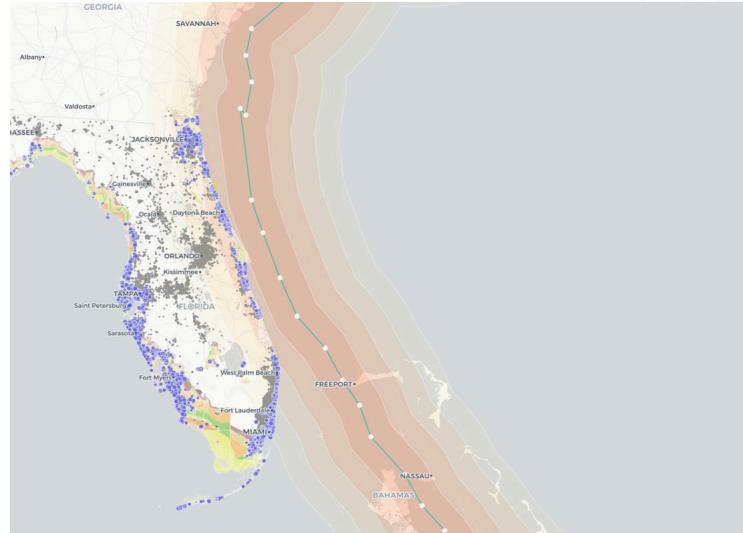
# Casos de estudio

Los datos espaciales son fundamentales en diversas áreas de la ciencia de datos debido a su *capacidad* para proporcionar información georreferenciada que puede revelar *patrones, tendencias y relaciones* que no son evidentes con datos no espaciales.

# Optimización de la cadena de suministros



# Modelización de catástrofes





## Análisis del mercado inmobiliario

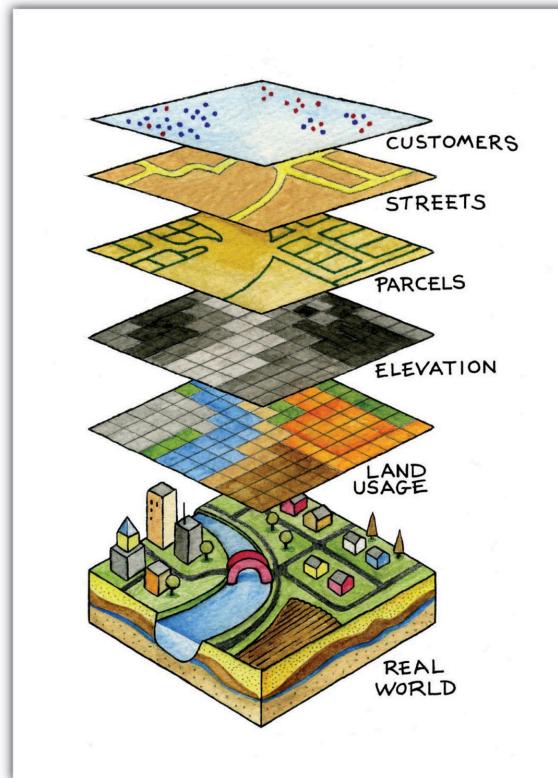


## Análisis sanitario



# ¿Cómo se ven los datos espaciales?

Los dos modelos de datos más utilizados para almacenar datos geoespaciales son los **vectores** y **ráster**.



# Vectores

- Los datos vectoriales se componen de lugares geométricos discretos (valores x,y) conocidos como vértices que definen la **forma** del objeto espacial.
- La organización de los vértices, determina el tipo de vector con el que estamos trabajando: punto, línea o polígono.

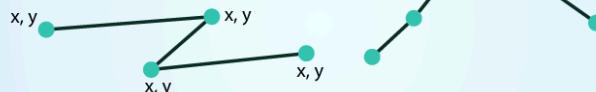
**POINTS:** Individual **x, y** locations.

ex: Center point of plot locations, tower locations, sampling locations.



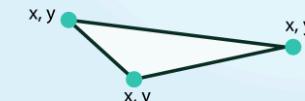
**LINES:** Composed of many (at least 2) vertices, or points, that are connected.

ex: Roads and streams.



**POLYGONS:** 3 or more vertices that are connected and **closed**.

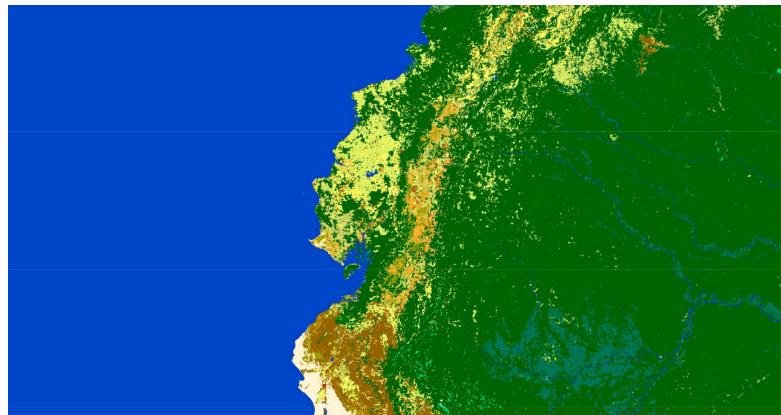
ex: Building boundaries and lakes.



neon

# Raster

- Los datos ráster o "cuadriculados" son datos que se guardan en **píxeles**.
- En el mundo espacial, cada píxel representa un área de la superficie terrestre. Por ejemplo, en el ráster que se muestra a continuación, cada píxel representa una clase concreta de cubierta terrestre que se encontraría en ese lugar del mundo real.

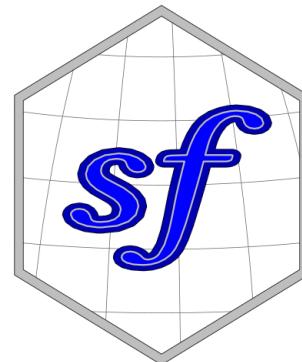


# ¿Cómo manipular estos datos?

Para este curso, emplearemos principalmente los siguientes paquetes: `sf`, `terra` y `tidyverse`.

## `sf`

- **Simple features** es un modelo de datos jerárquico desarrollado y aprobado por el Open Geospatial Consortium (OGC) que representa una amplia gama de tipos de geometría.
- El paquete `sf` puede representar todos los tipos comunes de geometría vectorial: puntos, líneas, polígonos y sus respectivas versiones multi (que agrupan características del mismo tipo en una única característica).



## terra

El paquete terra soporta objetos raster en R. Proporciona un amplio conjunto de funciones para crear, leer, exportar, manipular y procesar conjuntos de datos raster.



# tidyverse



El paquete tidyverse es una colección de paquetes orientados a la manipulación, importación, exploración y visualización de datos y que se utiliza exhaustivamente en ciencia de datos.

El uso de este paquete permite facilitar el trabajo estadístico y la generación de trabajos reproducibles. Está compuesto de los siguientes paquetes:

- readr
- dplyr
- ggplot2
- tibble
- tidyr
- purr
- stringr
- forcats

**Antes de pasar a la parte  
práctica**

# La *pipe* o escribir código como una oración

Pensemos en el juego del florón:

El *florón* **está** en mis manos  
de mis manos ya **paso**  
*las monjitas carmelitas*  
**se fueron** a Popayan  
a **buscar** lo que han perdido  
debajo del arrayan

Vamos a identificar dos sujetos (en cursiva): *el florón*, y *las monjitas*. Escribamos este canción como si se tratara de código de R. Primero para el florón:

```
## Objeto:  
floron <- florontito()  
  
floron <- está(qué = florón, en = mis_manos)  
  
floron <- pasó(qué = florón, de_donde = mis_manos, cuando = ya)
```

# La *pipe* o escribir código como una oración

Ahora para las monjitas carmelitas:

```
## Objeto:  
monjitas <- carmelitas()  
  
monitas <- fueron(quienes = monjitas, donde = Popayan)  
  
monitas <- buscar(quienes = monjitas, qué = lo_se_a_perdido, donde = debajo_arrayan)
```

Tanto para el florón como para las monjitas hemos descrito **verbos y sujetos**. Lo que en R se traduce como **funciones y objetos/argumentos**.

Podemos reescribir el caso de las monjitas:

```
buscar(  
  quienes = fueron(quienes = monjitas,  
                    donde = Popayan), # Despues de irse, buscan  
  qué = lo_se_a_perdido, donde = debajo_arrayan  
)
```

# La *pipe* o escribir código como una oración

En el primer caso, se entiende que es una **secuencia de pasos** que alteran el estado del **sujeto/objeto**. En el segundo caso se complica la lectura, pero el resultado es igual, R va a evaluar las funciones **de adentro hacia afuera**. Sin embargo podemos hacer más legible el código con una **pipe** o `%>%`:

**Floroncito:**

```
floron <- floroncito() %>%
  está(en = mis_manos) %>%
  pasó(de_donde = mis_manos, cuando = ya)
```

**Monjitas:**

```
monjitas <- carmelitas() %>%
  fueron(donde = Popayan) %>%
  buscar(qué = lo_se_a_perdido, donde = debajo_arrayan)
```

# *La pipe o escribir código como una oración*

Entonces la pipe nos permite:

- Componer funciones o secuencia de funciones
- Crear un código más legible

Pero debemos conocer sus alcances:

- La empleamos en una **secuencia de pasos** en las que se modifica un **objeto principal**
- No es recomendado unir más de 10 pasos
- Si necesitamos los resultados intermedios, es mejor no usarla

Para incluirla en nuestro código de forma sencilla usamos la combinación

**Ctrl + Shift + M**

# La *pipe* o escribir código como una oración

Empleemos la pipe para determinar los momentos de la distribución del *Número de empleados* en las empresas medianas tipo A, por sector de actividad en la ENESEM.

Primero descompongamos el ejercicio en pasos:

1. **Leemos** los *datos/tabla* del archivo
2. **Filtrar** los *datos/tabla* para las empresas medianas tipo A
3. **Calcular** con los *datos/tabla* el número total de empleados
4. **Agrupar** los *dato/tablas* por tamaño
5. **Calcular** con los *datos/tabla* los momentos de la distribución

Primero filtramos para que las operaciones se realicen con menos observaciones

Fijemonos que se cumplen las condiciones para usar la pipe 😊

# La *pipe* o escribir código como una oración

```
library(haven)
library(tidyverse)

resumen <- read_sav("data/2020_ENESEM_BDD_TIC.sav") %>%
  filter(des_tamano == "Mediana Empresa A") %>%
  mutate(total_empleados = tic2_5_pers_ocup_h + tic2_5_pers_ocup_m) %>%
  group_by(des_sector) %>%
  summarise(
    media = mean(total_empleados, na.rm = TRUE),
    desviacion = sd(total_empleados, na.rm = TRUE),
    mediana = median(total_empleados, na.rm = TRUE),
    q25 = quantile(total_empleados, probs = 0.25, na.rm = TRUE),
    q75 = quantile(total_empleados, probs = 0.75, na.rm = TRUE),
    empresas = n()
)
```

# *La pipe o escribir código como una oración*

```
## # A tibble: 5 × 7
##   des_sector    media desviacion mediana    q25    q75 empresas
##   <chr>        <dbl>      <dbl>     <dbl> <dbl>    <dbl>    <int>
## 1 Comercio      18.0       28.8      10     5.25   19.5      54
## 2 Construcción  26.4       25.0      16      8      40       35
## 3 Manufactura    42.1       28.0      42     18.2     60       24
## 4 Minería       44.0       28.4      32.5   24.8     69.8      24
## 5 Servicios     48.2       60.3      35     13.5     62.5     215
```

# Intervalos de confianza para la media

En el ejemplo anterior empleamos la función `mutate` para crear la variable `total_empleados`. Para poder hacer inferencia estadística sobre **la diferencia de las medias** entre grupos podemos emplear los intervalos de confianza para las medias.

$$\bar{x} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

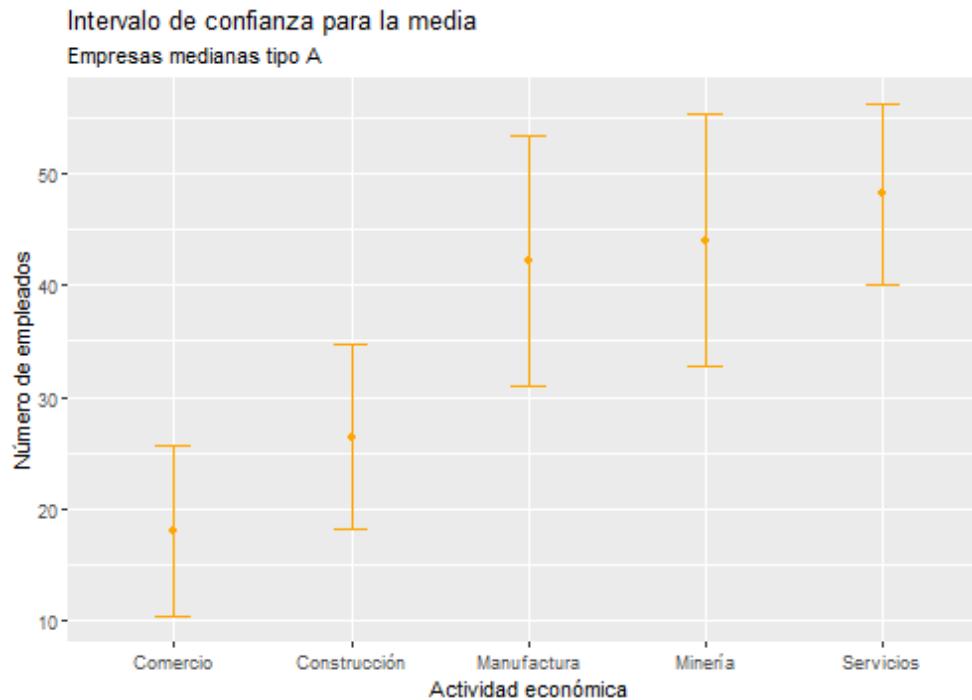
Con  $\alpha = 0.05$  y  $z \sim N(0, 1)$ :

```
intervalo <- resumen %>%
  # Creamos nuevas variables a partir del resultado anterior
  mutate(error_estandar = (qnorm(0.975)*desviacion)/sqrt(empresas),
         inferior = media - error_estandar,
         superior = media + error_estandar) %>%
  # Elegimos las variables de interes
  select(des_sector,inferior,media,superior)
```

Vamos a ver estos valores de forma gráfica:

```
plot <- intervalo %>%
  # La pipe nos permitira llamar a las variables de tabla dentro
  # de las capas de ggplot:
  ggplot() +
  # Barras de error para el intervalo:
  geom_errorbar(mapping = aes(x = des_sector,
                               ymin = inferior,
                               ymax = superior),
                color = "orange", width = 0.2) +
  # Ploteamos la media
  geom_point(mapping = aes(x = des_sector,
                           y = media),
             color = "orange")
```

# Gráfico de los intervalos de confianza



Podemos decir estadísticamente que las actividades de manufactura, minería y servicios contratan más empleados con respecto a las actividades de comercio. No podemos decir lo mismo entre el comercio y la construcción.

# Práctica: Comparando los sectores por tamaño

Vamos a replicar este análisis incluyendo en el análisis el tamaño y el sector. Recuerden abrir el proyecto y en un script de R vamos a ampliar el análisis.

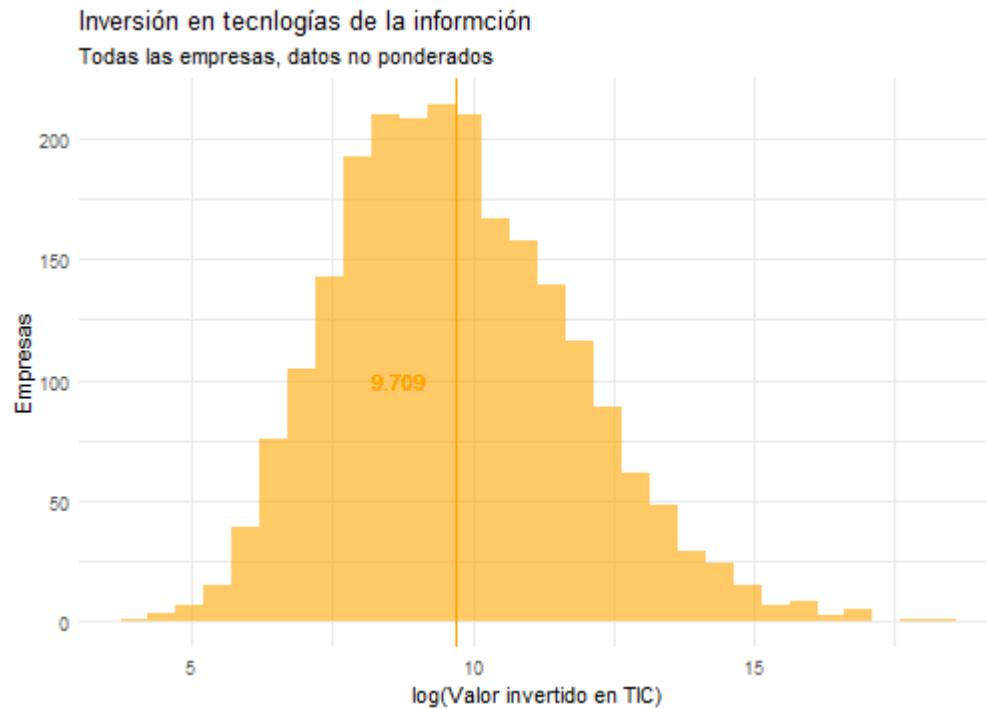
# Condiciones

Ya sea para **filtrar** o para crear variables **lógicas** tenemos algunos operadores:

Operador	Descripción
<	Menor a
<=	Menor o igual a
>	Mayor a
>=	Mayor o igual a
==	Igual
!=	No igual
!x	No x
x   y	x ó y (elemento a elemento)
x    y	x ó y (de vectores)
x & y	'x' y 'y'
is.na(x)	¿Son vacíos o missing?

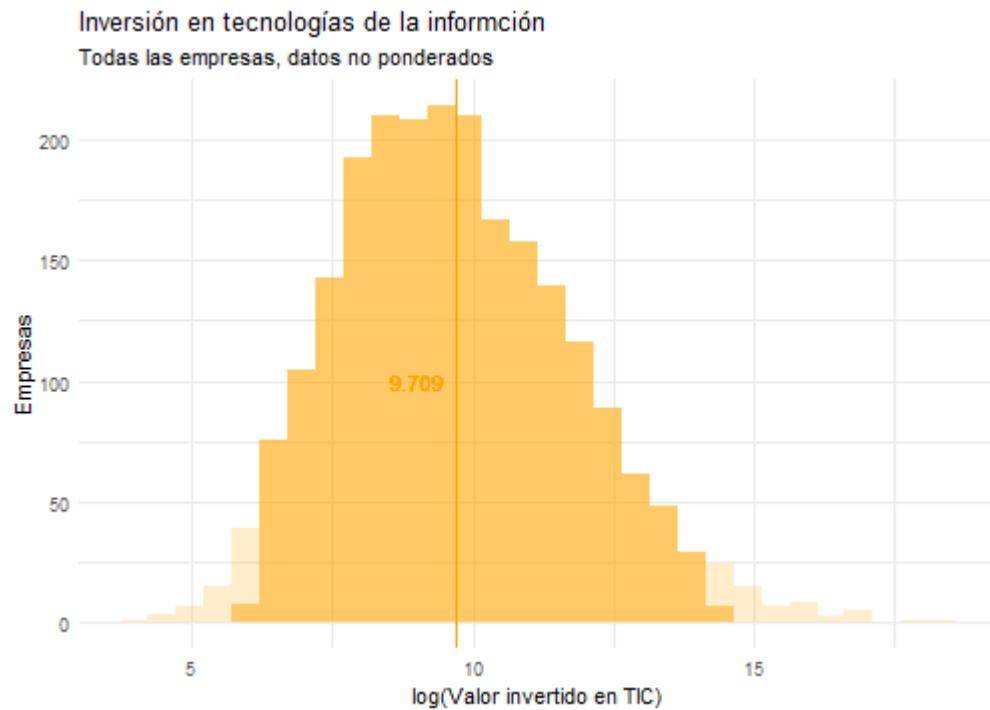
# Más reflexiones sobre los valores atípicos (outliers)

Recordemos de la distribución de la variable **logaritmo de inversión en TICS**:



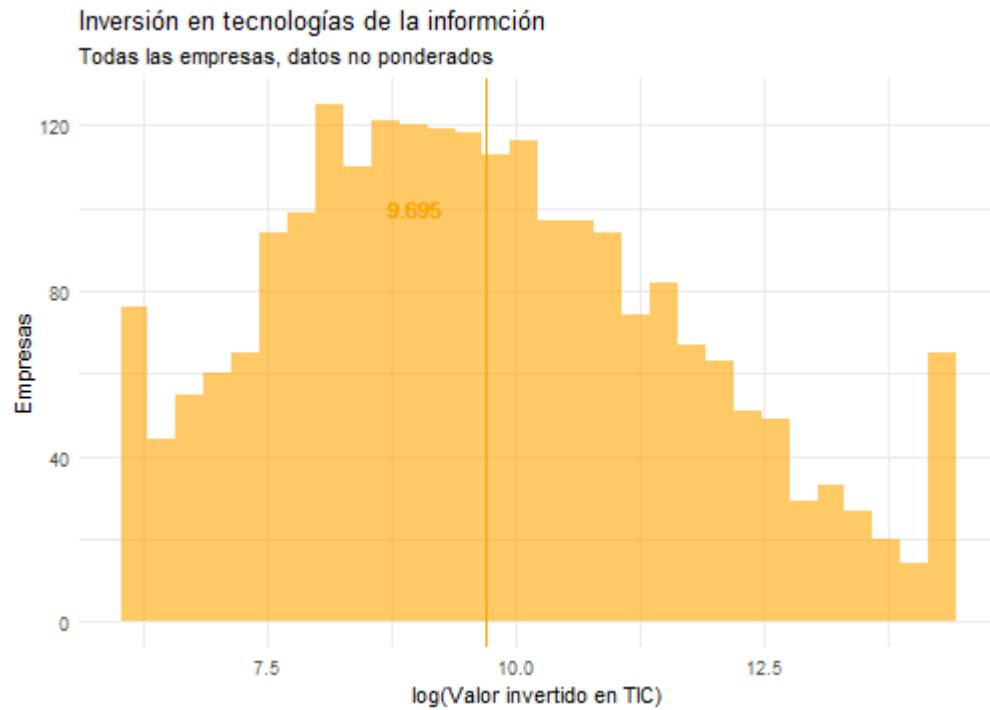
# Más reflexiones sobre los valores atípicos: alternativas

La media truncada al 5% (2.5% en cada cola):



# Más reflexiones sobre los valores atípicos: alternativas

La transformación **winsorize** al 95%:



**Gracias por la atención**