Artículo 1: Redes Bayesianas Multinomiales

Jose Angel Govea Garcia Diego Vértiz Padilla
Daniel Sánchez Fortiz Augusto Ley Rodriguez

2025-08-31

Table of contents

1 Abstract

En este documento se analizó la base de datos de la encuesta Origen-Destino en Hogares del Instituto Nacional de Estadística y Geografía (INEGI). Construimos redes bayesianas (DAGs) y realizamos selección de modelo comparando BIC para elegir la mejor estructura. Evaluamos relaciones de dependencia mediante pruebas con p-values. Con la DAG final, aprendida por hill climbing, inferimos probabilidades de consultas (queries) sobre elección modal y duración de viaje, proporcionando un marco interpretable para análisis y escenarios de política.

1.1 Keywords

Selección de modelo, redes bayesianas, BIC, dependencia, relaciones, p-values, hill climbing, queries

2 Introducción

La movilidad cotidiana en la Zona Metropolitana del Valle de México (ZMVM) concentra tensiones clave de la vida urbana: congestión, tiempos de traslado prolongados y efectos en el ambiente y la salud pública. Para recopilar información y entender estos fenómenos con evidencia robusta, el Instituto Nacional de Estadística y Geografía realizó la encuesta Origen-Destino en Hogares de la ZMVM (EOD) en 2017, con el principal objetivo de obtener las características de los viajes diarios, sus motivos, duraciones, horarios y medios de transporte.

Una red bayesiana es un modelo probabilístico que representa un conjunto de variables aleatorias y sus dependencias mediante un grafo dirigido acíclico (DAG). Cada nodo del DAG es una variable y cada arco indica dependencia directa. Cuando se interpreta un DAG, se pueden hacer preguntas de los casos que suceden en la base de datos para poder obtener información relevante de manera sencilla. Es por eso que es una herramienta útil para analizar los datos de la encuesta de INEGI.

La EOD tiene capas que una red bayesiana permite modelar al relacionar características de las personas del Valle de México y agruparlas para estudiar sus hábitos de traslado y explicarlos estadísticamente. Con esta herramienta también se pueden realizar escenarios para la toma de decisiones. Por ejemplo, se pueden estimar efectos de intervención, es decir, los cambios en las probabilidades de los eventos al cambiar algunas variables.

3 Metodología

3.1 EDA

La primera parte del trabajo fue realizar un análisis exploratorio de los datos, esto basado en las queries que buscábamos resolver. Analizamos los 5 archivos con la información referente a los hogares, aspectos demográficos, métodos de transporte, información sobre los viajes y aspectos sobre la vivienda de las personas. Buscamos qué preguntas se realizaron en el censo del INEGI que nos ayudarían a responder nuestras propias preguntas. Una vez encontradas estas preguntas, seleccionamos la columna dentro del conjunto de datos perteneciente a esta pregunta.

3.2 Limpieza y Preparacion de Datos

3.2.1 Limpieza 1

Para llevar a cabo el proceso de limpieza y transformación de la base de datos se instalaron e implementaron distintos paquetes de R especializados en la manipulación de datos. En particular, se utilizaron: readr, que permite importar y leer archivos de manera eficiente; dplyr, empleado para la manipulación de tablas mediante funciones que simplifican operaciones como filtrado, selección y agrupamiento; y tidyr, el cual facilita la reestructuración de los datos y la gestión de valores faltantes.

```
##install.packages("readr")
##install.packages("dplyr")
##install.packages("tidyr")
```

Una vez instalados, se cargaron los paquetes en el entorno de trabajo para habilitar sus funciones y garantizar la correcta ejecución de las operaciones de limpieza y transformación de los datos.

```
library(readr)
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
```

```
library(tidyr)
```

En esta etapa se definieron los subconjuntos de variables relevantes para cada tabla de la base de datos. Para ello, se crearon vectores que almacenan únicamente las columnas necesarias de acuerdo con el tipo de información: vivienda (cols_viv), hogar (cols_hog), características sociodemográficas (cols_sdem), viajes (cols_vje) y transporte (cols_tra). Esta selección inicial permite reducir la complejidad del conjunto de datos, mantener únicamente las variables pertinentes para el análisis y establecer una estructura organizada que facilita el cruce posterior entre tablas.

```r
cols_viv <- c("id_viv", "p1_1")                                    # tvivienda
cols_hog <- c("id_hog", "id_viv")                                  # thogar
cols_sdem <- c("id_soc", "id_hog", "n_ren", "sexo", "niv")         # tsdem
cols_vje <- c(                                                      # tviaje
  "id_via", "id_soc", "p5_6", "p5_11a", "p5_11_a",
  "p5_14_01", "p5_14_05", "p5_14_08",
  "p5_9_1", "p5_9_2", "p5_10_1", "p5_10_2",
  "estrato"
)
cols_tra <- c("id_via", "p5_14", "p5_3", "n_via") # ttransporte
```

Se importaron los archivos en formato .csv correspondientes a las tablas de vivienda, hogar, sociodemográficas, viajes y transporte. Durante este proceso también se realizaron algunos ajustes en los nombres de las variables para facilitar su lectura. Por ejemplo, en la tabla de vivienda la variable p1_1 se renombró como n_personas_vivienda, y en la tabla sociodemográfica la variable niv pasó a llamarse nivel_estudios.

En la tabla de viajes, además de seleccionar las columnas necesarias, se crearon nuevas variables que describen mejor los desplazamientos, como el origen y destino, así como indicadores del uso de automóvil, metro y autobús. También se desagregaron los tiempos de inicio y fin en horas y minutos para poder calcular con precisión la duración de cada trayecto. Por último, en la tabla de transporte la variable p5_14 se renombró como transporte_publico, con el fin de hacer la información más clara y comprensible.

```r
viv <- read_csv("tvivienda.csv", col_types = cols(), col_select = any_of(cols_viv)) %>%
  rename(n_personas_vivienda = p1_1)

hog <- read_csv("thogar.csv", col_types = cols(), col_select = any_of(cols_hog))

sdem <- read_csv("tsdem.csv", col_types = cols(), col_select = any_of(cols_sdem)) %>%
  rename(nivel_estudios = niv)

vje <- read_csv("tviaje.csv", col_types = cols(), col_select = any_of(cols_vje)) %>%
  mutate(
    # origen / destino (acepta p5_11a o p5_11_a)
    origen  = .data[["p5_6"]],
    destino = coalesce(.data[["p5_11a"]], .data[["p5_11a"]]),
    uso_auto    = .data[["p5_14_01"]],
    uso_metro   = .data[["p5_14_05"]],
    uso_autobus = .data[["p5_14_08"]],
    dur_horas_i = .data[["p5_9_1"]],
    dur_min_i   = .data[["p5_9_2"]],
```

```
    dur_horas_v = .data[["p5_10_1"]],
    dur_min_v   = .data[["p5_10_2"]]
  ) %>%
  select(id_via, id_soc, origen, destino, uso_auto, uso_metro, uso_autobus,
         dur_horas_i, dur_min_i, dur_horas_v, dur_min_v, estrato)

tra <- read_csv("ttransporte.csv", col_types = cols(), col_select = any_of(cols_tra)) %>%
  rename(transporte_publico = p5_14)
```

Una vez preparadas las tablas individuales, se integraron en un único conjunto de datos. La tabla de viajes se tomó como base, ya que constituye el eje central del análisis, y sobre ella se unieron progresivamente las demás mediante operaciones de left join. Primero se incorporó la información sociodemográfica a través del identificador de persona (id_soc), después los datos de hogar con el identificador de hogar (id_hog), luego la información de vivienda con el identificador de vivienda (id_viv), y por último los datos de transporte mediante el identificador de viaje (id_via).

```
df <- vje %>%
  left_join(sdem, by = "id_soc") %>%
  left_join(hog,  by = "id_hog") %>%
  left_join(viv,  by = "id_viv") %>%
  left_join(tra,  by = "id_via")
#Partimos desde viaje porque será el más usado en nuestro análisis.
```

Finalmente, se creó un conjunto de datos con las variables necesarias para el análisis. Se mantuvieron los identificadores principales de viaje, persona, hogar y vivienda, además de variables como el número de personas en la vivienda, los medios de transporte usados, el origen y destino de los viajes, la duración de los trayectos, el sexo, el estrato, el nivel de estudios y el uso de transporte público. De esta forma se obtuvo un dataset claro y enfocado, que después fue exportado a formato .csv para poder trabajar con él en las siguientes etapas del estudio.

```
df_final <- df %>%
  select(
    id_via, id_soc, id_hog, id_viv,
    # Query 1 - Vertiz
    n_personas_vivienda, uso_metro, uso_autobus,
    # Query 2 - Sanchez
    origen, destino, uso_auto, dur_horas_i, dur_min_i, dur_horas_v, dur_min_v,
    # Query 3 - Augusto
    sexo, estrato,
    # Query 4 - Govea
    nivel_estudios, transporte_publico
```

```
  )
glimpse(df_final)
```

```
Rows: 890,748
Columns: 18
$ id_via              <dbl> 2936, 2936, 2937, 2937, 2938, 2939, 2940, 2940, 29~
$ id_soc              <dbl> 1268, 1268, 1268, 1268, 1268, 1268, 1269, 1269, 12~
$ id_hog              <dbl> 434, 434, 434, 434, 434, 434, 434, 434, 434, 434, ~
$ id_viv              <dbl> 418, 418, 418, 418, 418, 418, 418, 418, 418, 418, ~
$ n_personas_vivienda <chr> "03", "03", "03", "03", "03", "03", "03", "03", "0~
$ uso_metro           <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ uso_autobus         <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ origen              <chr> "01", "01", "03", "03", "01", "07", "01", "01", "0~
$ destino             <chr> "03", "03", "01", "01", "07", "01", "03", "03", "0~
$ uso_auto            <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,~
$ dur_horas_i         <chr> "08", "08", "14", "14", "17", "23", "08", "08", "1~
$ dur_min_i           <chr> "00", "00", "00", "00", "00", "00", "00", "00", "0~
$ dur_horas_v         <chr> "09", "09", "15", "15", "17", "23", "09", "09", "1~
$ dur_min_v           <chr> "00", "00", "30", "30", "05", "05", "00", "00", "3~
$ sexo                <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1,~
$ estrato             <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
$ nivel_estudios      <chr> "07", "07", "07", "07", "07", "07", "06", "06", "0~
$ transporte_publico  <chr> "11", "14", "11", "14", "14", "04", "11", "14", "1~
```

```
readr::write_csv(df_final, "df_final_r.csv")
```

Posteriormente, se continuó con la limpieza en Python para calcular la duración total de los viajes en minutos. Para ello se combinaron las variables de horas y minutos de inicio y fin, y a partir de esa diferencia se creó una nueva columna denominada duracion_viaje_min. Con este paso ya no fue necesario conservar las variables originales de horas y minutos, por lo que se eliminaron para simplificar la base de datos. Finalmente, el dataset actualizado se exportó a un nuevo archivo .csv, listo para ser utilizado en el análisis.

```
import pandas as pd
df = pd.read_csv("df_final_r.csv")

df["duracion_viaje_min"] = (df["dur_horas_v"]*60 + df["dur_min_v"]) -(df["dur_horas_i"]*60 + di

df = df.drop(columns=["dur_horas_i","dur_min_i","dur_horas_v","dur_min_v"])
df.to_csv("df_final_py.csv", index=False, encoding="utf-8")
```

Finalmente, se reimportó el archivo consolidado en R para su verificación. Se leyó el dataset df_final_py.csv y, mediante glimpse(), se inspeccionó su estructura (tipos de dato, número de columnas y primeras observaciones) para confirmar que la transformación en Python se reflejara correctamente antes de continuar con el análisis.

```
data <- readr::read_csv("df_final_py.csv")
```

```
Rows: 890748 Columns: 15
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (15): id_via, id_soc, id_hog, id_viv, n_personas_vivienda, uso_metro, us...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(data)
```

```
Rows: 890,748
Columns: 15
$ id_via              <dbl> 2936, 2936, 2937, 2937, 2938, 2939, 2940, 2940, 29~
$ id_soc              <dbl> 1268, 1268, 1268, 1268, 1268, 1268, 1269, 1269, 12~
$ id_hog              <dbl> 434, 434, 434, 434, 434, 434, 434, 434, 434, 434, ~
$ id_viv              <dbl> 418, 418, 418, 418, 418, 418, 418, 418, 418, 418, ~
$ n_personas_vivienda <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
$ uso_metro           <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ uso_autobus         <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ origen              <dbl> 1, 1, 3, 3, 1, 7, 1, 1, 3, 3, 1, 1, 9, 9, 1, 2, 1,~
$ destino             <dbl> 3, 3, 1, 1, 7, 1, 3, 3, 1, 1, 9, 9, 1, 1, 2, 1, 6,~
$ uso_auto            <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,~
$ sexo                <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1,~
$ estrato             <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
$ nivel_estudios      <dbl> 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 8, 8, 8,~
$ transporte_publico  <dbl> 11, 14, 11, 14, 14, 4, 11, 14, 11, 14, 11, 14, 11,~
$ duracion_viaje_min  <dbl> 60, 60, 90, 90, 5, 5, 60, 60, 90, 90, 45, 45, 45, ~
```

3.2.2 Limpieza 2

Es necesario preparar el entorno de trabajo para el análisis. Primero, se incluyen las instrucciones para instalar los paquetes bnlearn y dplyr (comentadas como recordatorio, solo deben usarse si los paquetes no están previamente instalados). Después, se cargan ambas librerías en memoria para poder utilizarlas en el resto del estudio.

```
##install.packages("bnlearn")
##install.packages("dplyr")
```

```
library("bnlearn")
library(dplyr)
```

El conjunto de datos con el que contamos después de la primera limpieza es el siguiente:

```
data = read.csv("df_final_py.csv", stringsAsFactors = TRUE)
head(data)
```

```
  id_via id_soc id_hog id_viv n_personas_vivienda uso_metro uso_autobus origen
1   2936   1268    434    418                   3         2           2      1
2   2936   1268    434    418                   3         2           2      1
3   2937   1268    434    418                   3         2           2      3
4   2937   1268    434    418                   3         2           2      3
5   2938   1268    434    418                   3         2           2      1
6   2939   1268    434    418                   3         2           2      7
  destino uso_auto sexo estrato nivel_estudios transporte_publico
1       3        2    2       3              7                11
2       3        2    2       3              7                14
3       1        2    2       3              7                11
4       1        2    2       3              7                14
5       7        2    2       3              7                14
6       1        2    2       3              7                 4
  duracion_viaje_min
1                 60
2                 60
3                 90
4                 90
5                  5
6                  5
```

En la segunda limpieza de datos lo primero que hicimos fue eliminar 3 columnas referentes al uso de transporte (uso de metro, uso de carro, uso de autobús). Esto porque otra columna nos indicaba si el viaje se realizó en metro, carro, autobús, entre otros medios de transporte. También eliminamos del conjunto de datos los números identificadores (ID) de los diferentes conjuntos de datos, estos son el ID del viaje, social, vivienda y hogar. El código utiliza dplyr::select(-columna) para descartar las variables innecesarias y conservar solo la información útil para construir la red bayesiana.

```
data <- data %>%
  select(-uso_autobus, -uso_metro, -uso_auto)
```

```
data <- data %>%
  select(-id_via, -id_soc, -id_hog, -id_viv )
```

El siguiente paso fue renombrar los nombres de las columnas, esto para facilitar la creación de las DAG. Las columnas renombradas quedaron de la siguiente manera:

- NP = Número de personas en la vivienda
- O = Origen
- D = Destino
- S = Sexo
- E = Estrato
- NE = Nivel de Estudios
- T = Transporte público (también hay transporte no público, pero es el nombre de la variable)
- DV = Duración de viaje en minutos

El comando names(data) reemplaza los nombres originales de las variables por las abreviaciones que previamente designamos:

```
names(data) <- c("NP", "O","D", "S","E", "NE", "T", "DV")
```

Por último, discretizamos la columna DV, referente a la duración de viaje en minutos. Para esto creamos una nueva columna llamada DV60 con los valores discretizados de DV y eliminamos la columna DV, y renombramos DV60 como DV, ya que no sería de utilidad para nuestro análisis. Tras realizar esa limpieza, nuestro conjunto de datos nos quedó de la siguiente manera:

Crea una variable binaria a partir de DV (>60 → “1”, si no “0”) y la convierte en factor fijando el orden de niveles (“0” primero).

```
data$DV60 <- factor(ifelse(data$DV > 60, "1", "0"), levels = c("0","1"))
```

Elimina la DV original (numérica) para evitar duplicados y quedarnos solo con la versión discretizada.

```
data <- data %>%
  select(-DV )
```

Renombra las columnas por posición; la última pasa a llamarse DV (sustituyendo a DV60) para mantener el nombre esperado en el resto del análisis.

```
names(data) <- c("NP", "O","D", "S","E", "NE", "T", "DV")
```

```
head(data)
```

```
  NP O D S E NE  T DV
1  3 1 3 2 3  7 11  0
2  3 1 3 2 3  7 14  0
3  3 3 1 2 3  7 11  1
4  3 3 1 2 3  7 14  1
5  3 1 7 2 3  7 14  0
6  3 7 1 2 3  7  4  0
```

3.3 Propuestas DAGs

Para modelar las relaciones de dependencia entre variables, se construyeron grafos acíclicos dirigidos (DAGs), los cuales permiten representar de manera visual y formal la posible estructura probabilística entre los diferentes datos.

En una primera etapa, se construyeron tres DAGs de manera manual, basados en nuestro conocimiento contextual de las variables seleccionadas. Este proceso consistió en identificar las dependencias teóricas más plausibles entre las variables y representarlas en forma de arcos dirigidos. La construcción manual permitió explorar diferentes configuraciones estructurales sin la restricción de un algoritmo, asegurando además que cada grafo fuese acíclico y conceptualmente coherente.

Para la creación de una DAG primero se crean los nodos vacíos, sin ningún tipo de relación. Después, creamos la matriz de relaciones, dependiendo de cuáles hayamos establecido. En este proceso podemos ver la relación de dependencia entre los nodos padre y los nodos hijo. Una vez creada la matriz de relaciones, la incluimos dentro de nuestro DAG. Este proceso se realiza con las 3 DAG que diseñamos de acuerdo a nuestro criterio.

Este bloque de código está comentado, ya que únicamente debe ejecutarse en caso de que las librerías no se encuentren instaladas en el entorno. En primer lugar, se instala el paquete BiocManager, que actúa como gestor de librerías del repositorio Bioconductor. Posteriormente, a través de este gestor se instala Rgraphviz, el cual es necesario para la visualización gráfica de redes bayesianas (DAGs) generadas con bnlearn.

```
##install.packages("BiocManager")
##BiocManager::install("Rgraphviz")
```

3.3.1 DAG 1

Este bloque inicializa un grafo vacío con el conjunto de nodos que representan las variables de nuestro dataset: número de personas (NP), origen (O), destino (D), sexo (S), edad (E), nivel educativo (NE), tipo de transporte (T) y duración del viaje (DV). El objeto resultante (dag1) no contiene aún arcos, únicamente define la estructura base sobre la cual se construirá la red bayesiana.

```
dag1 = empty.graph(nodes = c("NP", "O", "D", "S", "E", "NE", "T", "DV"))
```

Este bloque construye una matriz de arcos dirigida (arc_set) que define las relaciones de dependencia entre las variables del modelo. Cada fila de la matriz representa un arco, donde la primera columna corresponde al nodo de origen (from) y la segunda al nodo de destino (to). De esta forma, se especifica la dirección de la influencia entre variables (por ejemplo, S → NE indica que el sexo influye en el nivel educativo). Esta matriz será utilizada posteriormente para asignar la estructura al grafo bayesiano.

```
arc_set = matrix(c("S", "NE",
                   "E", "NE",
                   "E", "T",
                   "E", "NP",
                   "T", "D",
                   "D", "O",
                   "D", "DV",
                   "O", "DV"), byrow = TRUE, ncol = 2,
                 dimnames = list(NULL, c("from", "to")))
```

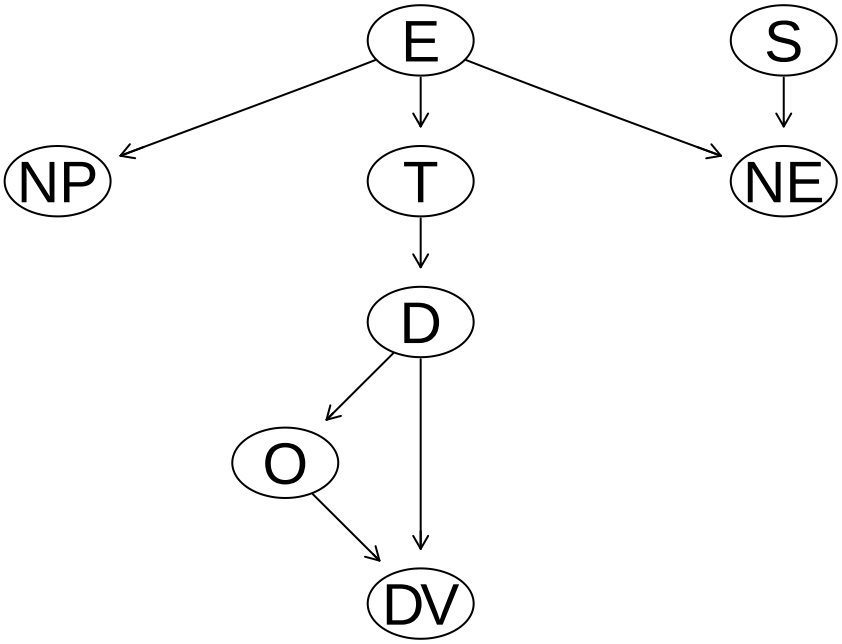
Este bloque asigna la matriz de arcos previamente definida (arc_set) al grafo vacío dag1. Con esto, el grafo pasa de ser una estructura sin conexiones a un grafo bayesiano dirigido con las relaciones especificadas entre las variables.

```
arcs(dag1) = arc_set
```

La DAG 1 la visualizamos gráficamente de la siguiente manera:

```
graphviz.plot(dag1, shape = "ellipse")
```

```
Loading required namespace: Rgraphviz
```



3.3.2 DAG 2

Este bloque crea un nuevo grafo vacío con los mismos nodos que el primero, pero ahora destinado a la construcción del segundo DAG que se empleará más adelante en el análisis.

```
dag2 = empty.graph(nodes = c("NP", "O", "D", "S", "E", "NE", "T", "DV"))
```

Este bloque define la matriz de arcos para el segundo DAG (dag2). Cada fila representa una relación dirigida de un nodo origen (from) a un nodo destino (to). En este caso se incluyen tanto las relaciones ya presentes en el primer grafo como la adición de la conexión S → T, lo que permite contrastar estructuras alternativas en el análisis.

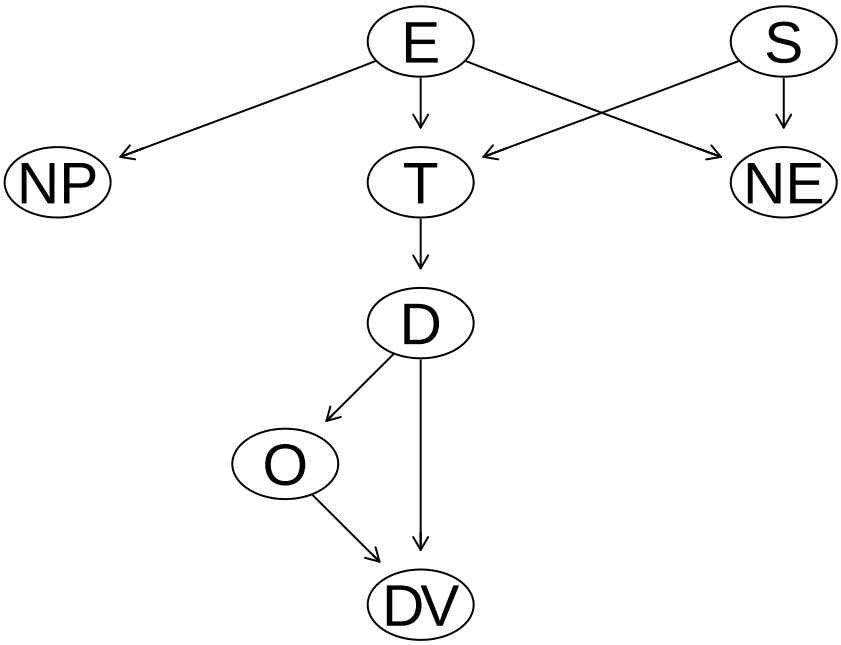
```
arc_set2 = matrix(c("S", "NE",
                    "S", "T",
                   "E", "NE",
                   "E", "T",
                   "E", "NP",
                   "T", "D",
                   "D", "O",
                   "D", "DV",
                   "O", "DV"), byrow = TRUE, ncol = 2,
                 dimnames = list(NULL, c("from", "to")))
```

Ese bloque asigna el conjunto de arcos definido en arc_set2 al grafo vacío dag2. En otras palabras, se está construyendo formalmente la estructura del segundo DAG a partir de las relaciones especificadas previamente en la matriz.

```
arcs(dag2) = arc_set2
```

La DAG 2 la visualizamos gráficamente de la siguiente manera:

```
graphviz.plot(dag2, shape = "ellipse")
```



3.3.3 DAG 3

Ese bloque crea un tercer grafo vacío (dag3) que contiene los mismos nodos que los DAG anteriores, pero sin arcos iniciales. Es la base sobre la cual se definirán las relaciones para la tercera estructura de red.

```
dag3 = empty.graph(nodes = c("NP", "O", "D", "S", "E", "NE", "T", "DV"))
```

Ese bloque define la matriz de arcos para el tercer DAG. En ella se especifican las relaciones de dependencia entre los nodos, donde cada par (from, to) representa una dirección de influencia. Por ejemplo, “E” → “T” indica que la variable E influye directamente sobre T.

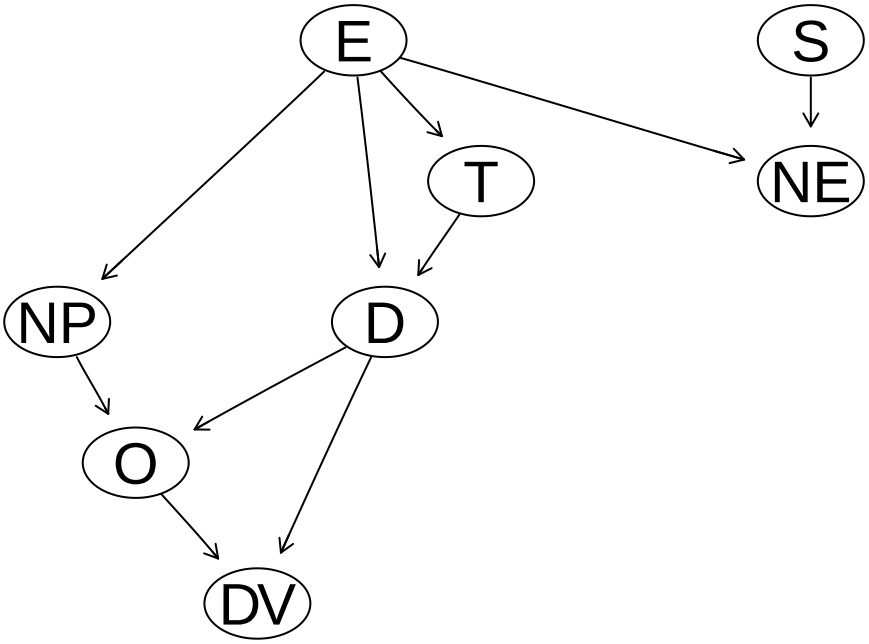
```
arc_set3 = matrix(c("S", "NE",
                    "E", "NE",
                    "E", "T",
                    "E", "NP",
                    "E", "D",
                    "NP", "O",
                    "T", "D",
                    "D", "O",
                    "D", "DV",
                    "O", "DV"), byrow = TRUE, ncol = 2,
                  dimnames = list(NULL, c("from", "to")))
```

Ese bloque asigna la matriz de arcos previamente definida (arc_set3) al objeto dag3, completando así la estructura del tercer grafo dirigido acíclico.

```
arcs(dag3) = arc_set3
```

La DAG 3 la visualizamos gráficamente de la siguiente manera:

```
graphviz.plot(dag3, shape = "ellipse")
```



3.4 Comparacion DAGs

Para poder comparar las tres estructuras propuestas de DAG mediante el criterio de información bayesiano (BIC), fue necesario transformar las variables de tipo entero a factores. Esto se debe a

que, en el contexto de redes bayesianas discretas, los valores numéricos que representan categorías (origen, destino, sexo, estrato, nivel de estudios o transporte) no deben ser tratados como cantidades continuas, sino como variables categóricas. Al convertirlas en factores, la librería bnlearn reconoce correctamente sus niveles y puede construir las tablas de probabilidad condicional asociadas a cada nodo. De esta manera, se garantizó que el cálculo del BIC fuera válido y se pudieran comparar de forma consistente las distintas configuraciones de los grafos.

```
data[] <- lapply(data, factor) #columnas a factor
str(data)
```

```
'data.frame':   890748 obs. of  8 variables:
 $ NP: Factor w/ 21 levels "1","2","3","4",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ O : Factor w/ 17 levels "1","2","3","4",..: 1 1 3 3 1 7 1 1 3 3 ...
 $ D : Factor w/ 17 levels "1","2","3","4",..: 3 3 1 1 7 1 3 3 1 1 ...
 $ S : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
 $ E : Factor w/ 4 levels "1","2","3","4": 3 3 3 3 3 3 3 3 3 3 ...
 $ NE: Factor w/ 11 levels "0","1","2","3",..: 8 8 8 8 8 8 7 7 7 7 ...
 $ T : Factor w/ 20 levels "1","2","3","4",..: 11 14 11 14 14 4 11 14 11 14 ...
 $ DV: Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 2 2 ...
```

Como se acaba de mencionar, utilizaremos el BIC para determinar cuál de los 3 DAG es una mejor representación de las relaciones entre los nodos.

El score de la DAG 1 usando BIC es de:

```
score(dag1, data = data, type = "bic")
```

```
[1] -9333963
```

El score de la DAG 2 usando BIC es de:

```
score(dag2, data = data, type = "bic")
```

```
[1] -9326168
```

El score de la DAG 3 usando BIC es de:

```
score(dag3, data = data, type = "bic")
```

```
[1] -9359756
```

Con estos resultados podemos determinar que la DAG que mejor representa las relaciones entre nodos es la DAG número 2. Esto porque su score es el más grande entre las 3. Un apunte importante y necesario sobre el BIC es que, en la práctica, el mejor score es el más pequeño. Debido a la programación de la librería bnlearn, se selecciona el número más grande.

Hill Climbing

Tras construir de manera manual las primeras 3 DAG, se aplicó el algoritmo Hill Climbing para poder determinar la mejor estructura de relaciones para una DAG referente a los datos con los que contamos. Este método corresponde a una técnica de aprendizaje que explora el espacio de grafos posibles siguiendo un proceso iterativo. En este, parte de un grafo inicial y evalúa los posibles cambios que se pueden realizar en los arcos. Para decidir qué combinación utilizar, se basa en un criterio de puntuación. De la misma manera que en los grafos manuales, utiliza el BIC.

El bloque ejecuta el algoritmo Hill Climbing para el aprendizaje automático de la estructura de la red bayesiana a partir de los datos.

```
best_dag = hc(data)
```

Este código convierte el grafo obtenido mediante el algoritmo Hill-Climbing en su representación textual compacta (model string). Esta notación muestra los nodos y las relaciones de dependencia entre ellos en una sola línea, lo que permite verificar rápidamente la estructura aprendida de la red bayesiana sin necesidad de graficarla.

```
modelstring(best_dag)
```

```
[1] "[E][NE|E][NP|E:NE][T|E:NE][DV|NE:T][O|NE:DV][D|O:DV][S|O:D:DV]"
```

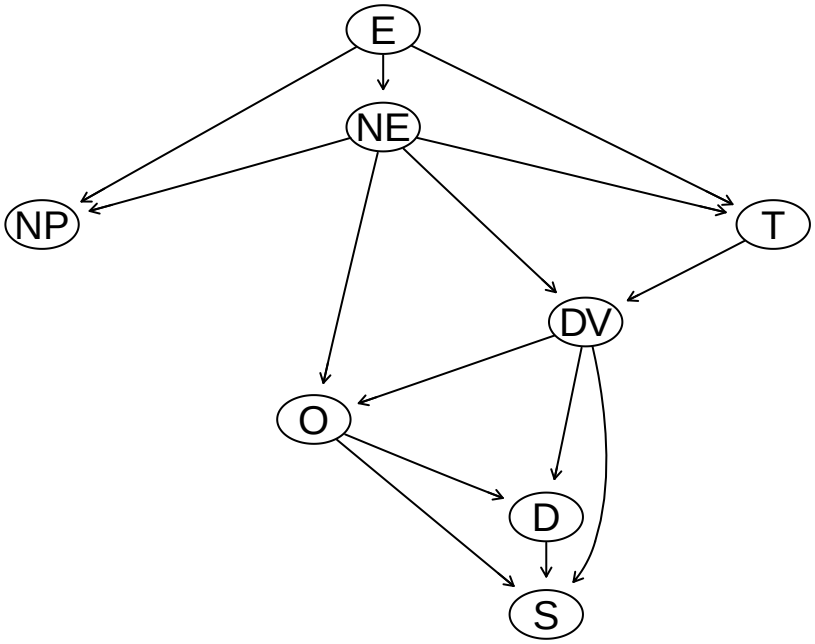
Una vez realizado el DAG, comprobaremos su score utilizando el BIC:

```
score(best_dag, data = data, type = "bic")
```

```
[1] -9211699
```

Este resultado nos muestra una clara mejora respecto a los 3 DAG creados de manera manual por nosotros. Para ver las relaciones entre los nodos padre y los nodos hijos lo visualizamos gráficamente:

```
graphviz.plot(best_dag, shape = "ellipse")
```



4 Aplicación

El siguiente código realiza el ajuste de parámetros de la red bayesiana. En este paso, se toma la estructura obtenida previamente con el algoritmo Hill-Climbing (best_dag) y, utilizando los datos originales (data), se estiman las tablas de probabilidad condicional para cada nodo dado sus padres. Con esto, la red deja de ser únicamente una representación estructural y se convierte en un modelo probabilístico.

```
bn = bn.fit(best_dag, data = data)
```

Una vez realizado el análisis exploratorio de datos, la correspondiente limpieza y la creación y evaluación de grafos acíclicos dirigidos (DAG), se determinó cuál de las estructuras representaba mejor las relaciones de dependencia entre los nodos. Con esta red seleccionada como la de mejor ajuste, procedimos a utilizarla como base para responder preguntas específicas acerca de la información disponible, las cuales denominaremos queries.

Cada query consiste en calcular la probabilidad de un evento dado cierto conjunto de evidencias. Esta función se basa en simulaciones de Monte Carlo, generando un gran número de muestras para aproximar las distribuciones condicionales.

Las preguntas de interés fueron las siguientes:

1. ¿Cuál es la probabilidad de que una persona que vive en una vivienda en la que habitan de 2 a 5 personas use transporte público, específicamente metro o autobús, como principal medio de viaje?

2. ¿Cuál es la probabilidad de que el viaje del hogar al trabajo dure más de 60 minutos si la persona viaja en coche?
3. ¿Cuál es la probabilidad de que una mujer use el colectivo/micro dado que es de estrato alto?
4. ¿Cuál es la probabilidad de que una persona con Licenciatura, Maestría o Doctorado utilice transporte público como medio principal de viaje para ir al trabajo?

La función cpquery recibe tres parámetros principales: el modelo ya ajustado, el evento de interés cuya probabilidad se quiere calcular y la evidencia que representa las condiciones bajo las cuales se evalúa dicho evento, además de un número de simulaciones que determina la precisión de la estimación.

El resultado del primer query es:

```
cpquery(bn, event = (T=="5" | T == "8"), evidence = ((NP == "2")|(NP == "3")|(NP == "4")|(NP ==
```

```
[1] 0.09259673
```

El resultado del segundo query es:

```
cpquery(bn, event = (O =="2" & D == "3" & DV == "1"), evidence = (T== "1") , n = 10^6)
```

```
[1] 0.0002545906
```

El resultado del tercer query es:

```
cpquery(bn, event = (S =="1" & T == "2" ), evidence = (E== "4") , n = 10^6)
```

```
[1] 0.0880232
```

El resultado del cuarto y ultimo query es:

```
cpquery(bn, event = (T=="2" | T == "5" | T == "6" | T== "8" | T== "10" | T == "11" | T == "12"
```

```
[1] 0.3709257
```

Como puede observarse, los primeros 3 queries muestran probabilidades muy bajas, lo que indica que, con la evidencia que se tiene en los datos, es poco común que suceda dicho evento. El segundo query, que pregunta la probabilidad de que el viaje del hogar al trabajo dure más de 60 minutos si la persona viaja en coche, presenta una probabilidad casi nula (0.017%), lo que sugiere que es un evento sumamente excepcional.

El cuarto query, aunque la probabilidad de dicho evento es menor al 50%, es considerablemente más probable (37.04%) en comparación con los 3 queries previos. Esto indica que, dadas las evidencias previas, hay una mayor probabilidad de que una persona con Licenciatura, Maestría o Doctorado utilice transporte público como medio principal de viaje para ir al trabajo, en comparación con el resto de queries.

5 Conclusiones

Los hallazgos principales tras la investigacion y los metodos aplicados durante el proyecto son los siguientes:

- Una apropiada limpieza y preparación de datos representa una parte fundamental para poder trabajar de manera correcta con la información y garantizar que los modelos construidos se basen en variables consistentes y confiables.
- Como se demostró, utilizar métodos de aprendizaje estructural como Hill Climbing permite encontrar relaciones más óptimas entre los nodos de acuerdo con el conjunto de datos, superando las aproximaciones manuales.
- El uso de grafos acíclicos dirigidos (DAGs) no solo facilita la representación de dependencias probabilísticas, sino que además habilita la posibilidad de responder queries que conectan el modelo con preguntas relevantes.

La relevancia de este trabajo radica en que proporciona un marco probabilístico interpretable que puede servir como apoyo en el diseño de políticas públicas orientadas a la movilidad. Por ejemplo, permite identificar qué condiciones hacen más probable el uso de transporte público, o qué factores están asociados a viajes prolongados.

Sin embargo, el análisis también presenta limitaciones:

- Los resultados dependen de la calidad y representatividad de la encuesta utilizada (EOD 2017), la cual puede no reflejar cambios recientes en los patrones de movilidad.
- La discretización de variables continuas, como la duración de viaje, implica una simplificación que puede afectar la precisión de las inferencias.

6 Referencias

Koller, D. & Friedman, N. Probabilistic Graphical Models: Principles and Techniques. MIT Press, 2009. (definición de BN/DAG, factorización e independencias). Pearl, J. “The Do-Calculus Revisited.” UAI Keynote, 2012. (bases para inferencia causal con DAGs). Ma, T.Y . et al. “Bayesian Networks for Multimodal Mode Choice Behavior.” Transportation Research Procedia, 2015 (aplicación a elección modal).