

# Proyecto integrador

## Predicción del índice de precios de la vivienda nueva - IPVN en Medellín y el área metropolitana

Maestría en ciencia de datos y analítica

Universidad Eafit.

Catalina Piedrahita Jaramillo, Susana Londoño Muñoz, David Betancur Londoño, Diego Valderrama Laverde, Diego Jaramillo Zapata

### 1. Clarificación del problema



En el caso colombiano, el acceso a la vivienda propia, por parte de las familias, es un proyecto de mediano y largo plazo, que es de gran relevancia en la construcción de patrimonio familiar y en el mejoramiento de condiciones de vida. En este proyecto, normalmente familiar, interactúan dos fuerzas: La oferta de viviendas, nuevas y usadas, según la zona de interés. La cual depende a su vez de la disponibilidad de suelos, costos de los insumos, cumplimiento normativos, entre otros.

La demanda, dada principalmente por la disponibilidad de recursos de las familias o inversores para comprar las viviendas. Acá desempeña un rol clave tanto los ahorros que puedan tener las personas, como también, el acceso efectivo a crédito. Este último, está fuertemente influenciado

por el cumplimiento de las condiciones de acceso y el valor de la tasa de interés. Destacando que, en los últimos años, la concurrencia de subsidios estatales también ha desempeñado un papel preponderante, al permitir un aumento de la disponibilidad de recursos.

La interacción de ambas fuerzas, que se conoce tradicionalmente como mercado, es lo que genera la formación de precios. Esta variable es de suma importancia, puesto que se convierte en la mayoría de los casos, en el factor decisivo para el acceso efectivo de las familias a una vivienda digna o deseada.

En los últimos años, en las principales ciudades y áreas metropolitanas del país, entre ellas Medellín y su área metropolitana, se ha presentado que, el precio promedio de la vivienda nueva, medido a través del índice de precios de la vivienda nueva - IPVN que estima trimestralmente el Departamento Administrativo Nacional de Estadística - DANE, ha mostrado un crecimiento continuo, inclusive por encima de la variación del índice de precios del consumidor (inflación). Lo cual, si bien, no ha detenido la construcción, y más importante, el acceso a vivienda nueva, plantea preguntas relacionadas sobre su comportamiento futuro y las variables de oferta y demanda que más influyen en su comportamiento.

El Índice de precios de vivienda - IPVN permite medir la variación porcentual promedio de los precios de venta de la vivienda nueva en proceso de construcción y/o hasta la última unidad vendida.

Entender cómo evolucionará el IPVN, desempeñaría un insumo clave en términos de política económica y social, con la cual se podrían diseñar instrumentos que faciliten el acceso, especialmente, de la población de menores ingresos, además, como un posibilitador para dinamizar el sector de la construcción, de importancia en la dinamización productiva de los territorios. Asimismo, podría ofrecer nueva información, en términos de rentabilidad de inversión futura, tanto para compradores como para constructores.

## Objetivo general

Predecir el comportamiento del índice de precios de la vivienda nueva (IPNV) en Medellín y su área metropolitana, empleando modelos estadísticos y técnicas de aprendizaje de máquina.

## Objetivos específicos

- Encontrar variables con mayor influencia en el IPVN.
- Medir la capacidad de predicción de modelos estadísticos y de aprendizaje de máquina.
- Evaluar la precisión de los modelos implementados.

## ▼ 2. Comprensión de los Datos

### ▼ - Descripción de los datos

Los datos usados para este trabajo provienen del Departamento Administrativo Nacional de Estadística -DANE-. Los datos específicos del sector de la construcción, tomados del DANE, se encuentran agregados en un informe de la Cámara colombiana de la construcción -CAMACOL- llamado "Colombia construcción en cifras." El dataset construido contiene las siguientes 42 variables macroeconómicas y del sector de la construcción:

- **d\_TD\_MDE\_Porc:** Tasa de desempleo de Medellín | Porcentaje
- **d\_IPVN\_MDE\_Num:** Índice de Precio de Vivienda Nueva Medellín Valor Total | Valor
- **d\_TO\_MDE\_Porc:** Tasa de ocupación de Medellín | Porcentaje
- **d\_PT\_MDE\_Num:** Población total de Medellín | Millones
- **d\_PET\_MDE\_Num\_Millon:** Población en edad de trabajar en Medellín | Millones
- **d\_PEA\_MDE\_Nun\_Millon:** Población económicamente activa en Medellín | Millones
- **d\_IPC\_MDE\_Num:** Índice de precio al consumidor | Valor
- **d\_IPC\_NAL\_Porc:** Índice de precio al consumidor | Valor
- **d\_TOSC\_Nal\_Porc:** Tasa de ocupación del sector de la construcción nacional | Porcentaje
- **d\_TDSC\_NAL\_Porc:** Tasa de desocupación del sector de la construcción nacional | Porcentaje
- **d\_UVAPCSL\_Total\_NAL\_Num:** Unidades de vivienda aprobadas para construcción según licencias | Valor
- **d\_UIVIS\_MDEVA\_Num:** Unidades iniciadas de VIS Valle de Aburrá | Valor
- **d\_UINOVIS\_MDEVA\_Num:** Unidades iniciadas de NO VIS Valle de Aburrá | Valor
- **d\_UIVISNOVIS\_MDEVA\_Num:** Unidades iniciadas VIS y NO VIS Valle de Aburrá | Valor
- **d\_PIB\_VPCB2015\_NAL\_Num\_Mil\_million:** PIB Valor a precio corriente base 2015 en miles de millones (Datos corregidos de efectos estacionales y de calendario) | Miles de millones
- **d\_PIB\_VPKB2015\_NAL\_Num\_Mil\_million:** PIB Valor a precio Constante base 2015 en miles de millones (Datos corregidos de efectos estacionales y de calendario) | Miles de millones
- **d\_PIB\_VPCB2015\_SC\_Num\_Mil\_Millon:** PIB sector de la construcción valor a precio corriente base 2015 en miles de millones | Miles de millones
- **d\_PIB\_VPKB2015\_SC\_Num\_Mil\_Millon:** PIB sector de la construcción valor a precio Constante base 2015 en miles de millones | Miles de millones
- **d\_TIPPBR\_NAL\_Porc:** Tasa de interés promedio ponderado (Promedio trimestral de la tasa de interés de colocación) del Banco de la Republica %1 | Porcentaje
- **d\_TIPPST\_NAL\_Porc:** Tasa de interés promedio ponderado (Promedio trimestral de la tasa de interés de colocación) sin tesorería % 3 | Porcentaje
- **d\_OC\_MDE\_MT2:** Obras culminadas en el área metropolitana de Medellín, metros cuadrados. | Metros cuadrados
- **d\_UVAPCSL\_Total\_ANT\_Num:** Unidades de vivienda aprobadas para construcción según licencias | Valor
- **d\_UVAPCSL\_VIS\_ANT\_Num:** Unidades de vivienda aprobadas para construcción según licencias, Vivienda de interés social, Antioquia. Número de unidades | Valor

- **d\_UVAPCSL\_NoVIS\_Ant\_Num:** Unidades de vivienda aprobadas para construcción según licencias, NO vivienda de interés social, Antioquia. Número de unidades | Valor
- **d\_ALCSD\_ANT\_VIS\_mt2:** ANTIOQUIA: Área (m<sup>2</sup>) licenciada para construcción según destino, vivienda de interés social | Metros cuadrados
- **d\_ALCSD\_NoVis\_ANT\_mt2:** ANTIOQUIA: Área (m<sup>2</sup>) licenciada para construcción según destino, NO vivienda de interés social | Metros cuadrados
- **d\_ICCV\_NAL\_Num:** Índice de costos de construcción de vivienda (ICCV) - | Valor
- **d\_ICCV\_Mat\_NAL\_Num:** Índice de costos de construcción de vivienda (ICCV) - Material | Valor
- **d\_ICCV\_MO\_NAL\_Num:** Índice de costos de construcción de vivienda (ICCV) - Mano de Obra | Valor
- **d\_ICCV\_MaqEq\_NAL\_Num:** Índice de costos de construcción de vivienda (ICCV) - Maquinaria y Equipo | Valor
- **d\_ICCV\_MDE\_Num:** Índice de Costos de Construcción de Vivienda (ICCV) | Valor
- **d\_IPPTotal\_NAL\_Num:** Índice de Precios al Productor (IPP) - Total | Valor
- **d\_IPPMat\_NAL\_Num:** Índice de Precios al Productor (IPP) - Materiales de Construcción | Valor
- **d\_IPVUNom\_NAL\_Num:** Índice de precios de vivienda usada Nominal | Valor
- **d\_IPVUReal\_NAL\_Num:** Índice de precios de vivienda usada Real | Valor
- **d\_AFC\_NAL\_Millon:** Cuentas de Ahorro para el Fomento de la Construcción - AFC, saldo y número | Millones
- **d\_AFC\_NAL\_Numero:** Cuentas de Ahorro para el Fomento de la Construcción - AFC, saldo y número | Valor
- **d\_CAPVis\_NAL\_Millon:** Cuentas de Ahorro Programado - CAP para VIS, saldo y número | Millones
- **d\_CAPVis\_NAL\_Num:** Cuentas de Ahorro Programado - CAP para VIS, saldo y número | Valor
- **d\_PDCG\_NAL\_Ton:** Producción de Cemento Gris en Toneladas - Nacional | Toneladas
- **d\_DDCG\_NAL\_Ton:** Despacho de Cemento Gris en Toneladas - Nacional | Toneladas
- **d\_DDCG\_ANT\_Ton:** Despacho de Cemento Gris en Toneladas - Antioquia | Toneladas

Para cada una de las variables se hizo el cálculo de las variaciones trimestrales anuales, de forma que el conjunto de datos final contiene 84 variables.

Fuentes:

Cámara Colombiana de la Construcción - CAMACOL- (2020). Colombia construcción en cifras septiembre 2020. Disponible en: <https://bit.ly/39rfMiY>

Departamento Nacional de Estadística -DANE-(2020). Cuentas nacionales. Disponible en: <https://bit.ly/33uYxcK>

Instalar e importar paquetes y bibliotecas

```

1 # instalar bibliotecas
2 !pip install pandas-profiling
3 !pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
4 !pip install featuretools
5 !pip install chart_studio
6
7 # importar bibliotecas
8 from google.colab import files
9 from pathlib import Path
10 import requests
11 import pandas_profiling
12 from pandas_profiling import ProfileReport
13 from pandas_profiling.utils.cache import cache_file
14 import numpy as np
15 import pandas as pd
16 import matplotlib.pyplot as plt
17 import seaborn as sns
18 sns.set()
19
20 from plotly.subplots import make_subplots
21 import plotly.graph_objects as go
22 import math
23 import warnings
24 warnings.filterwarnings('ignore')

```

## ▼ - Recolección de datos

```

1 #Para cargar desde google drive:
2 from google.colab import drive
3 drive.mount('/content/drive')
4

```

Mounted at /content/drive

```

1 #Rutas de los archivos en google drive:
2 jara = "/content/drive/MyDrive/Proyecto Integrador Semestre 2/dataset_vf_deci_coma.xlsx"
3

```

## ▼ Conjunto de datos completo (índices y variaciones)

```

1 ruta = jara

1 data1 = pd.read_excel(ruta)
2 data1.head()

```

ID	Periodo	Año	Trim	d_IPVN_MDE_Num	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_I
0	1	2005-01	2005	I	41.72	15.050086	50.983674
1	2	2005-02	2005	II	42.85	14.461586	50.981004
2	3	2005-03	2005	III	41.61	14.811176	50.987551
3	4	2006-01	2006	I	45.94	14.317852	51.264581
4	5	2006-02	2006	II	45.94	14.317852	51.264581

```
1 data1.shape
```

(61, 88)

```
1 data1.info()
```

31	a_PPPMat_NAL_Num	61	non-null	float64
32	d_IPVUNom_NAL_Num	61	non-null	float64
33	d_IPVUReal_NAL_Num	61	non-null	float64
34	d_AFC_NAL_Millon	61	non-null	float64
35	d_AFC_NAL_Num	61	non-null	float64
36	d_CAPVIS_NAL_Millon	61	non-null	float64
37	d_CAPVIS_NAL_Num	61	non-null	float64
38	d_PDCG_NAL_ton	61	non-null	float64
39	d_DDCG_NAL_Ton	61	non-null	float64
40	d_DCG_Ant_Ton	61	non-null	int64
41	d_UIVIS_MDEVA_Num	61	non-null	int64
42	d_UINOVIS_MDEVA_Num	61	non-null	int64
43	d_UIVISNOVIS_MDEVA_Num	61	non-null	int64
44	d_PIB_VPKB2015_NAL_Num_Mil_million	61	non-null	float64
45	d_PIB_VPKB2015_SC_Num_Mil_million	61	non-null	float64
46	v_IPVN_MDE_Num	57	non-null	float64
47	v_TD_MDE_Porc	57	non-null	float64
48	v_TO_MDE_Porc	57	non-null	float64
49	v_PT_MDE_Num_Millon	57	non-null	float64
50	v_PET_MDE_Num_Millon	57	non-null	float64
51	v_PEA_MDE_Num_Millon	57	non-null	float64
52	v_IPC_MDE_Num	57	non-null	float64
53	v_IPC_NAL_Porc	57	non-null	float64
54	v_TOSC_NAL_Porc	57	non-null	float64
55	v_TDSC_NAL_Porc	57	non-null	float64
56	v_UVAPCSL_Total_NAL_Num	57	non-null	float64
57	v_PIB_VPCB2015_NAL_Num_Mil_million	57	non-null	float64
58	v_PIB_VPCB2015_SC_Num_Mil_Millon	57	non-null	float64
59	v_TIPPBR_NAL_Porc	57	non-null	float64
60	v_TIPPST_NAL_Porc	57	non-null	float64
61	v_OC_MDE_mt2	57	non-null	float64
62	v_UVAPCSL_Total_ANT_Num	57	non-null	float64
63	v_UVAPCSL_VIS_ANT_Num	57	non-null	float64
64	v_UVAPCSL_NoVIS_Ant_Num	57	non-null	float64
65	v_ALCSD_ANT_VIS_mt2	57	non-null	float64
66	v_ALCSD_NoVis_ANT_mt2	57	non-null	float64
67	v_ICCV_NAL_Num	57	non-null	float64
68	v_ICCV_Mat_NAL_Num	57	non-null	float64
69	v_ICCV_MO_NAL_Num	57	non-null	float64

```

70 v_ICCV_MaqEq_NAL_Num      57 non-null   float64
71 v_ICCV_MED_Num           57 non-null   float64
72 v_IPPTotal_NAL_Num       57 non-null   float64
73 v_PPPMat_NAL_Num         57 non-null   float64
74 v_IPVUNom_NAL_Num        57 non-null   float64
75 v_IPVUReal_NAL_Num       57 non-null   float64
76 v_AFC_NAL_Millon         57 non-null   float64
77 v_AFC_NAL_Num            57 non-null   float64
78 v_CAPVIS_NAL_Millon      57 non-null   float64
79 v_CAPVIS_NAL_Num          57 non-null   float64
80 v_PDCG_NAL_ton           57 non-null   float64
81 v_DDCG_NAL_Ton           57 non-null   float64
82 v_DCG_Ant_Ton            57 non-null   float64
83 v_UIVIS_MDEVA_Num        57 non-null   float64
84 v_UINOVIS_MDEVA_Num      57 non-null   float64
85 v_UIVISNOVIS_MDEVA_Num   57 non-null   float64
86 v_PIB_VPKB2015_NAL_Num_Mil_million 57 non-null   float64
87 v_PIB_VPKB2015_SC_Num_Mil_million 57 non-null   float64
dtypes: float64(73), int64(13), object(2)
memory usage: 42.1+ KB

```

Exceptuando las etiquetas de periodo y trimestre, todos los datos son numéricos.

## ▼ - Datos nulos

```

1 nulls = data1.isnull().sum()
2 nulls = nulls[nulls > 0]
3 nulls

v_IPVN_MDE_Num             4
v_TD_MDE_Porc               4
v_TO_MDE_Porc               4
v_PT_MDE_Num_Millon         4
v_PET_MDE_Num_Millon        4
v_PEA_MDE_Num_Millon        4
v_IPC_MDE_Num               4
v_IPC_NAL_Porc              4
v_TOSC_NAL_Porc             4
v_TDSC_NAL_Porc             4
v_UVAPCSL_Total_NAL_Num     4
v_PIB_VPCB2015_NAL_Num_Mil_million 4
v_PIB_VPCB2015_SC_Num_Mil_Millon 4
v_TIPPBR_NAL_Porc            4
v_TIPPST_NAL_Porc            4
v_OC_MDE_mt2                 4
v_UVAPCSL_Total_ANT_Num      4
v_UVAPCSL_VIS_ANT_Num        4
v_UVAPCSL_NoVIS_Ant_Num      4
v_ALCSD_ANT_VIS_mt2          4
v_ALCSD_NoVis_ANT_mt2        4
v_ICCV_NAL_Num                4
v_ICCV_Mat_NAL_Num            4
v_ICCV_MO_NAL_Num              4

```

```
v_ICCV_MaqEq_NAL_Num          4
v_ICCV_MED_Num                4
v_IPPTotal_NAL_Num           4
v_PPPMat_NAL_Num             4
v_IPVUNom_NAL_Num            4
v_IPVUReal_NAL_Num           4
v_AFC_NAL_Millon             4
v_AFC_NAL_Num                 4
v_CAPVIS_NAL_Millon          4
v_CAPVIS_NAL_Num              4
v_PDCG_NAL_ton               4
v_DDCG_NAL_Ton               4
v_DCG_Ant_Ton                4
v_UIVIS_MDEVA_Num            4
v_UINOVIS_MDEVA_Num          4
v_UIVISNOVIS_MDEVA_Num       4
v_PIB_VPKB2015_NAL_Num_Mil_million 4
v_PIB_VPKB2015_SC_Num_Mil_million 4
dtype: int64
```

No se eliminan las filas con los nulos. Estas corresponden a columnas de variaciones las cuales, como son anuales, empiezan desde el 2006.

## ▼ - Datos duplicados

```
1 duplicateRowsDF = data1[data1.duplicated()]
2 duplicateRowsDF
```

ID	Periodo	Año	Trim	d_IPVN_MDE_Num	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_Mi
0 rows × 88 columns							

No se identifican datos duplicados.

## ▼ - Exploración de variables (Profiling)

Para la iniciar la exploración de variables se utiliza *Pandas profiling*

```
1 profile = ProfileReport(data1, minimal=True)
2 #profile.to_file(output_file="output.html")
3 profile
4
```

Summarize dataset:	97/97 [00:05<00:00, 17.28it/s,
100%	Completed]
Generate report structure: 100%	1/1 [03:23<00:00, 203.05s/it]
Render HTML: 100%	1/1 [00:02<00:00, 2.47s/it]

# Overview

## Dataset statistics

<b>Number of variables</b>	88
<b>Number of observations</b>	61
<b>Missing cells</b>	168
<b>Missing cells (%)</b>	3.1%
<b>Duplicate rows</b>	0
<b>Duplicate rows (%)</b>	0.0%
<b>Total size in memory</b>	42.1 KiB
<b>Average record size in memory</b>	706.1 B

## Variable types

<b>NUM</b>	86
<b>CAT</b>	2

## Warnings

v_IPVN_MDE_Num has 4 (6.6%) missing values	Missing
v_TD_MDE_Porc has 4 (6.6%) missing values	Missing

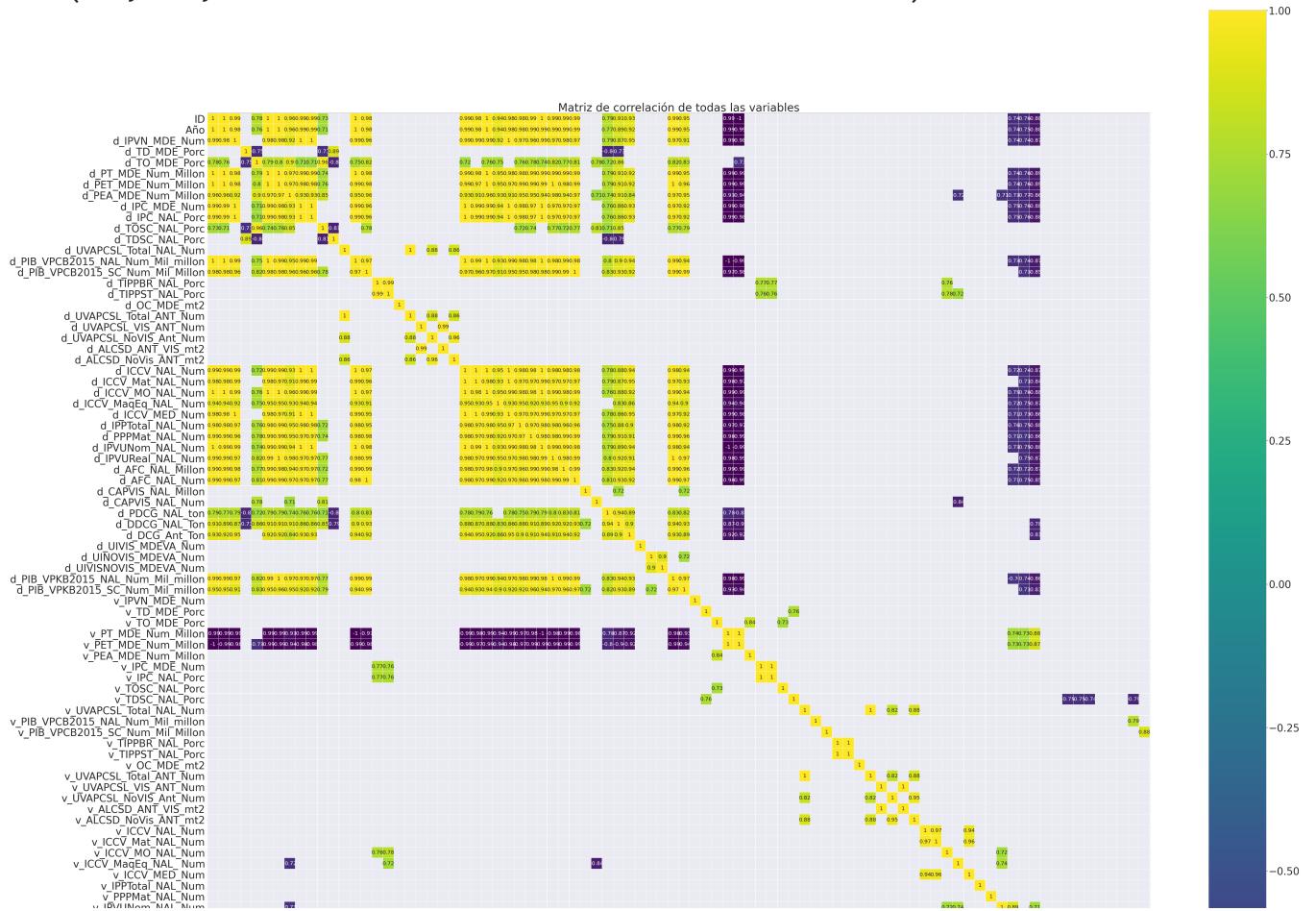
De la exploración de variables se identifican algunas que tienen variaciones que se consideran

## ▼ Matriz de correlación de todas las variables

---

```
1 corr = data1.corr()
2 plt.figure(figsize=(80, 80))
3
4 sns.set(font_scale=3.5)
5 ax = sns.heatmap(corr[(corr >= 0.7) | (corr <= -0.7)], # 70% de correlación
6                  cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
7                  annot=True, annot_kws={"size": 20}, square=True);
8 plt.title("Matriz de correlación de todas las variables")
```

Text(0.5, 1.0, 'Matriz de correlación de todas las variables')



De la matriz anterior se puede ver como la variable respuesta [d\_IPVN\_MDE\_Num] está correlacionada con otros índices y variaciones. A continuación se hará una exploración más detallada de la variable respuesta

## ▼ - Exploración de la variable respuesta: d\_IPVN\_MDE\_Num y su variación

```
1 var_res = data1.iloc[:,0:5]
2 var_res['d_IPVN_MDE_Num'] = data1.d_IPVN_MDE_Num
3 var_res['v_IPVN_MDE_Num'] = data1.v_IPVN_MDE_Num
4 var_res.describe()
```

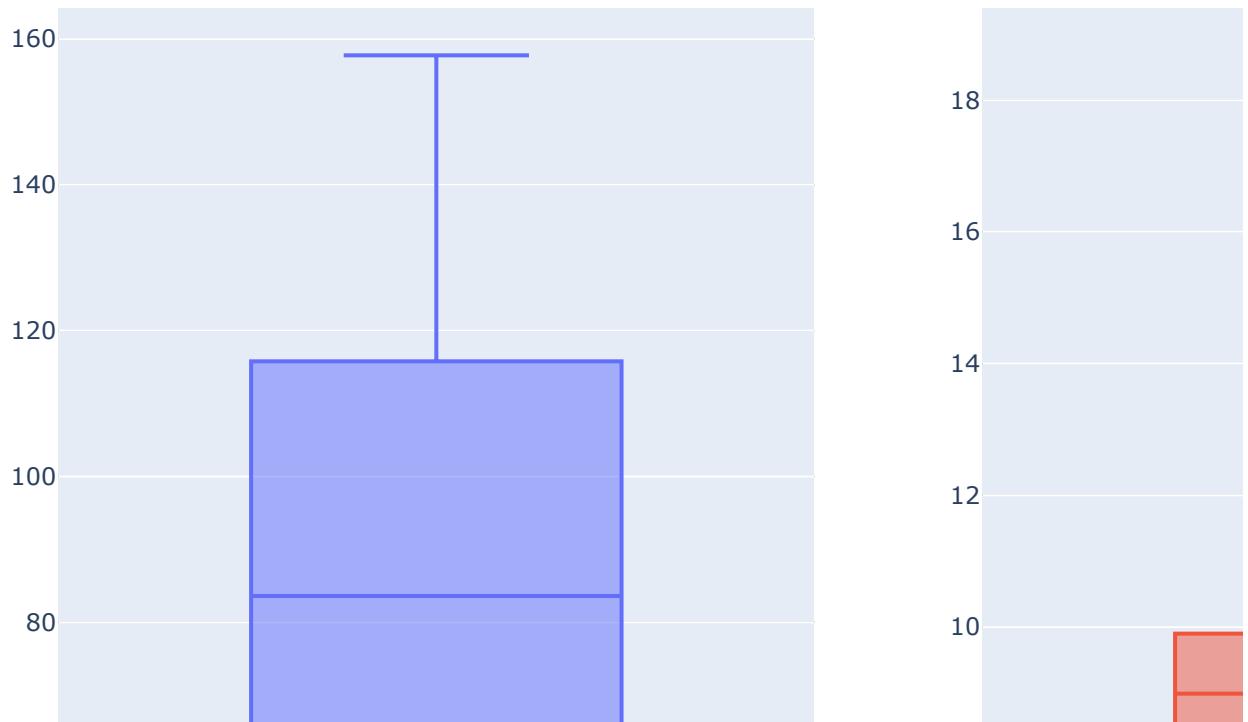
	ID	Año	d_IPVN_MDE_Num	v_IPVN_MDE_Num
<b>count</b>	61.000000	61.000000	61.000000	57.000000

```
1 var_res.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61 entries, 0 to 60
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   ID               61 non-null      int64  
 1   Periodo          61 non-null      object  
 2   Año              61 non-null      int64  
 3   Trim             61 non-null      object  
 4   d_IPVN_MDE_Num  61 non-null      float64 
 5   v_IPVN_MDE_Num  57 non-null      float64 
dtypes: float64(2), int64(2), object(2)
memory usage: 3.0+ KB
```

## Cajas y bigotes

```
1 df_i_vs = var_res.iloc[:,4:6]
2 total_items = len(df_i_vs.columns)
3 items_per_row = 2
4 total_rows = math.ceil(total_items / items_per_row)
5
6 fig = make_subplots(rows=total_rows, cols=items_per_row)
7
8 cur_row = 1
9 cur_col = 1
10
11 for index, column in enumerate(df_i_vs.columns):
12     fig.add_trace(go.Box(y=df_i_vs[column], name=column), row=cur_row, col=cur_col)
13
14     if cur_col % items_per_row == 0:
15         cur_col = 1
16         cur_row = cur_row + 1
17     else:
18         cur_col = cur_col + 1
19
20
21 fig.update_layout(height=650, width=1000, showlegend=False)
22 fig.show()
```



## Histogramas

```
1 sns.set(font_scale=1.2)
2 plt.figure(figsize=(9, 8))
3 var_res.hist(['d_IPVN_MDE_Num', 'v_IPVN_MDE_Num'], bins = 20, figsize=(10, 6));
4 plt.show()
```

```
<Figure size 648x576 with 0 Axes>
```

d\_IPVN\_MDE\_Num

v\_IPVN\_MDE\_Num

Serie de tiempo del índice



A continuación se muestran dos gráficas del comportamiento del índice y su variación en el tiempo



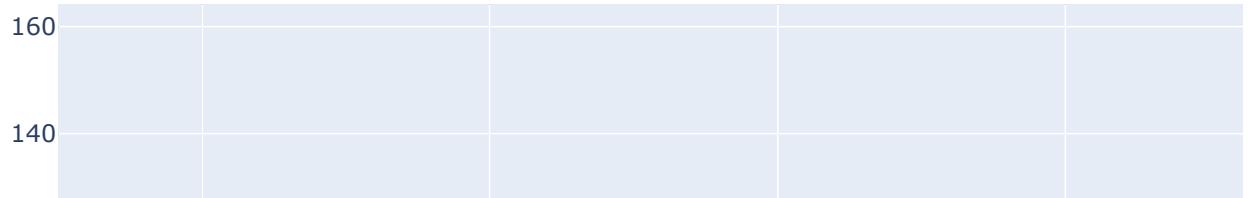
```
1 indices_data = go.Scatter(x=var_res.Periodo,y=var_res.d_IPVN_MDE_Num)
2 layout = go.Layout (title = 'Serie de Tiempo - Indices IPVN MDE', xaxis = dict (title = 'Trimestre'),
3                     yaxis = dict (title = 'Indice'))
4
```



```
1 def configure_plotly_browser_state():
2     import IPython
3     display(IPython.core.display.HTML('''
4         <script src="/static/components/requirejs/require.js"></script>
5         <script>
6             requirejs.config({
7                 paths: {
8                     base: '/static/base',
9                     plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
10                },
11            });
12         </script>
13     '''))
```

```
1 import chart_studio.plotly as py
2 import numpy as np
3 from plotly.offline import init_notebook_mode, iplot
4 import plotly.graph_objs as go
5 #from plotly.graph_objs import Contours, Histogram2dContour, Marker, Scatter
6 configure_plotly_browser_state()
7 init_notebook_mode(connected=False)
8 fig = go.Figure (data = [indices_data], layout = layout)
9 iplot(fig)
```

## Serie de Tiempo - Indices IPVN MDE



Serie de tiempo de la variación del índice

100

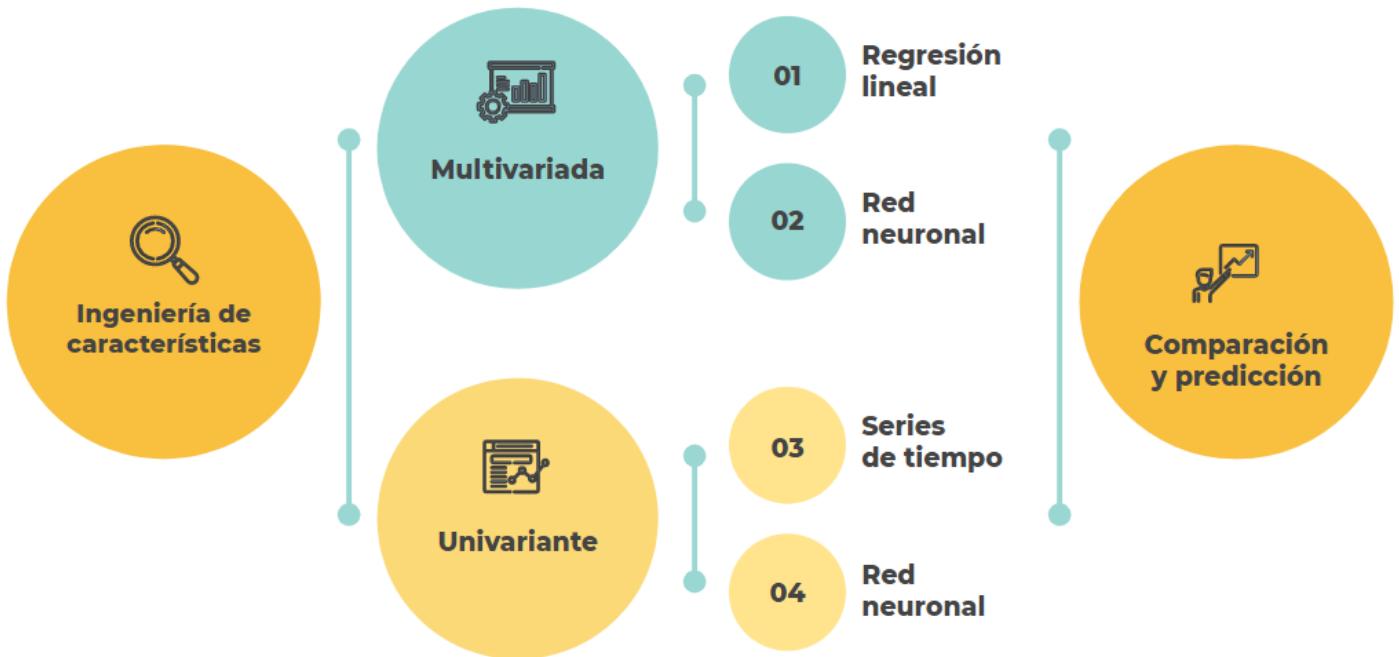
```
1 variaciones_data = go.Scatter(x=var_res.Periodo,y=var_res.v_IPVN_MDE_Num)
2 layout = go.Layout (title = 'Serie de Tiempo - Variaciones IPVN MDE', xaxis = dict (title = 'Trimestre'),
3                     yaxis = dict (title = 'Variación'))
```

```
1 import chart_studio.plotly as py
2 import numpy as np
3 from plotly.offline import init_notebook_mode, iplot
4 import plotly.graph_objs as go
5 #from plotly.graph_objs import Contours, Histogram2dContour, Marker, Scatter
6 configure_plotly_browser_state()
7 init_notebook_mode(connected=False)
8 fig = go.Figure (data = [variaciones_data], layout = layout)
9 iplot(fig)
```

## Serie de Tiempo - Variaciones IPVN MDE

### ▼ Análisis propuesto

Luego de la exploración realizada se plantea explorar por un lado la predicción del índice a partir de las otras variables y por otro la predicción a partir de su propio comportamiento histórico. En ambos casos, se trabajan dos posibilidades, un método estadístico y uno de aprendizaje de máquina. Para el análisis multivariante se tomarán las variables y se hará la ingeniería de características para identificar aquellas que ofrecen mejor resultado en una regresión lineal, como método estadístico. Contrastado con una red neuronal como método de aprendizaje de máquina. Para el análisis de la variable respuesta se usarán series de tiempo como método estadístico y una red neuronal como método de aprendizaje de máquina. Finalmente, se podrán comparar los resultados para determinar cuál sería el modelo con mejores resultados y se hará una predicción.



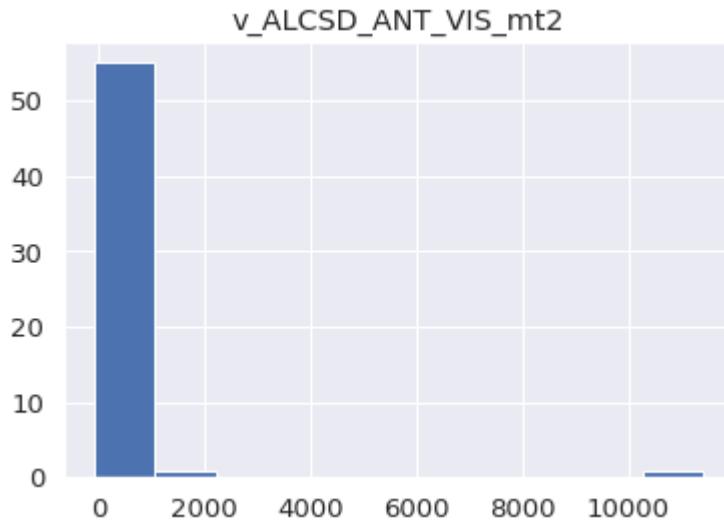
### ▼ 3. Preparación de los datos

#### ▼ - Limpieza de datos

De la exploración de variables en el profiling se identifican algunos índices que tienen variaciones que se consideran datos atípicos y se hace la eliminación del conjunto de datos.

```
1 #ejemplo de variación atípica
2 data1.hist(column='v_ALCSD_ANT_VIS_mt2')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f52e5ca9be0>]],  
      dtype=object)
```



```
1 data2 = data1.iloc[:,4:]
```

```
1 data1 = data1.drop(['d_UIVIS_MDEVA_Num', 'd_UINOVIS_MDEVA_Num', 'd_UIVISNOVIS_MDEVA_Num', 'd_UVAPCSL_VIS_ANT_Num', 'd_UVAPCSL_Nc  
2                 'v_UIVIS_MDEVA_Num', 'v_UINOVIS_MDEVA_Num', 'v_UIVISNOVIS_MDEVA_Num', 'v_UVAPCSL_VIS_ANT_Num', 'v_UVAPCSL_Nc  
3                 axis = 1  
4                 )  
5  
6 data1.head()
```

	ID	Periodo	Año	Trim	d_IPVN_MDE_Num	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_I
0	1	2005-01	2005	I	41.72	15.050086	50.983674	3107.1
1	2	2005-02	2005	II	42.85	14.461586	50.981004	3119.0
2	3	2005-03	2005	III	44.81	14.811179	50.667551	3131.1
3	4	2005-04	2005	IV	45.80	10.760881	53.078909	3143.1
4	5	2006-01	2006	I	45.94	14.317852	51.264581	3155.1

## Detección de outliers

Se determinó el porcentaje de datos atípicos extremos, se consideraron valores atípicos aquellos que:

$$Q_1 - 1.5 \cdot IQR > q > Q_3 + 1.5 \cdot IQR$$

Dónde  $Q_1$  y  $Q_3$  son el primer y tercer cuartil, y  $IQR$  el rango intercuartil ( $Q_3 - Q_1$ ).

```

1 #Para identificar el porcentaje de outlier por variable
2 for k, v in data2.items():
3     q1 = v.quantile(0.25)
4     q3 = v.quantile(0.75)
5     irq = q3 - q1
6     v_col = v[(v <= q1 - 1.5 * irq) | (v >= q3 + 1.5 * irq)]
7     perc = np.shape(v_col)[0] * 100.0 / np.shape(data2)[0]
8     print("Column %s outliers = %.2f%%" % (k, perc))

Column d_ICCV_MED_Num outliers = 0.00%
Column d_IPPTotal_NAL_Num outliers = 0.00%
Column d_PPPMat_NAL_Num outliers = 0.00%
Column d_IPVUNom_NAL_Num outliers = 0.00%
Column d_IPVUReal_NAL_Num outliers = 0.00%
Column d_AFC_NAL_Millon outliers = 0.00%
Column d_AFC_NAL_Num outliers = 0.00%
Column d_CAPVIS_NAL_Millon outliers = 11.48%
Column d_CAPVIS_NAL_Num outliers = 0.00%
Column d_PDCG_NAL_ton outliers = 0.00%
Column d_DDCG_NAL_Ton outliers = 0.00%
Column d_DCG_Ant_Ton outliers = 0.00%
Column d_UIVIS_MDEVA_Num outliers = 3.28%
Column d_UINOVIS_MDEVA_Num outliers = 0.00%
Column d_UIVISNOVIS_MDEVA_Num outliers = 0.00%
Column d_PIB_VPKB2015_NAL_Num_Mil_millon outliers = 0.00%
Column d_PIB_VPKB2015_SC_Num_Mil_millon outliers = 0.00%
Column v_IPVN_MDE_Num outliers = 6.56%
Column v_TD_MDE_Porc outliers = 0.00%
Column v_TO_MDE_Porc outliers = 1.64%

Column v_PT_MDE_Num_Millon outliers = 0.00%
Column v_PET_MDE_Num_Millon outliers = 0.00%
Column v_PEA_MDE_Num_Millon outliers = 3.28%
Column v_IPC_MDE_Num outliers = 0.00%
Column v_IPC_NAL_Porc outliers = 0.00%
Column v_TOSC_NAL_Porc outliers = 1.64%
Column v_TDSC_NAL_Porc outliers = 0.00%
Column v_UVAPCSL_Total_NAL_Num outliers = 1.64%
Column v_PIB_VPCB2015_NAL_Num_Mil_millon outliers = 0.00%
Column v_PIB_VPCB2015_SC_Num_Mil_Millon outliers = 3.28%
Column v_TIPPBR_NAL_Porc outliers = 0.00%
Column v_TIPPST_NAL_Porc outliers = 0.00%
Column v_OC_MDE_mt2 outliers = 4.92%
Column v_UVAPCSL_Total_ANT_Num outliers = 1.64%
Column v_UVAPCSL_VIS_ANT_Num outliers = 8.20%
Column v_UVAPCSL_NoVIS_Ant_Num outliers = 6.56%
Column v_ALCSD_ANT_VIS_mt2 outliers = 6.56%
Column v_ALCSD_NoVis_ANT_mt2 outliers = 8.20%
Column v_ICCV_NAL_Num outliers = 0.00%
Column v_ICCV_Mat_NAL_Num outliers = 6.56%
Column v_ICCV_MO_NAL_Num outliers = 1.64%
Column v_ICCV_MaqEq_NAL_Num outliers = 18.03%
Column v_ICCV_MED_Num outliers = 0.00%
Column v_IPPTotal_NAL_Num outliers = 3.28%
Column v_PPPMat_NAL_Num outliers = 0.00%
Column v_TDVIINom_NAL_Num outliers = 1.61%

```

```

Column v_IPVUReal_NAL_Num outliers = 0.00%
Column v_AFC_NAL_Millon outliers = 9.84%
Column v_AFC_NAL_Num outliers = 0.00%
Column v_CAPVIS_NAL_Millon outliers = 4.92%
Column v_CAPVIS_NAL_Num outliers = 22.95%
Column v_PDCG_NAL_ton outliers = 3.28%
Column v_DDCG_NAL_Ton outliers = 0.00%
Column v_DCG_Ant_Ton outliers = 1.64%
Column v_UIVIS_MDEVA_Num outliers = 1.64%
Column v_UINOVIS_MDEVA_Num outliers = 3.28%
Column v_UIVISNOVIS_MDEVA_Num outliers = 4.92%
Column v_PIB_VPKB2015_NAL_Num_Mil_million outliers = 0.00%
Column v_PIB_VPKB2015_SC_Num_Mil_million outliers = 3.28%

```

Se observó que la cantidad de datos atípicos no es mucha y la disponibilidad de datos es poca, por lo que si se retiran quedan pocos datos, por lo que no se realizó ningún proceso de eliminación de *outliers*.

### Camino de los índices

La variación de la variable respuesta muestra muy poca correlación con las demás variables, por lo que el enfoque se dara usando el indice que muestra mayor relación con las demás.

```

1 # dividiendo dataset en indices y variaciones
2 df_encabezado = data1.iloc[:,1:4]
3 df_indices = data1.filter(like='d_', axis=1)
4 df_variaciones = data1.filter(like='v_', axis=1)

1 df_indices = pd.concat([df_encabezado, df_indices ], axis=1)
2 df_variaciones = pd.concat([df_encabezado, df_variaciones], axis=1)

1 v_iccv = df_variaciones.loc[:, 'v_ICCV_MED_Num']

1 df_indices = pd.concat([df_indices, v_iccv], axis=1)

1 df_indices.shape
(61, 46)

1 df_indices.head()

```

	Periodo	Año	Trim	d_IPVN_MDE_Num	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_Mill
0	2005-01	2005	I	41.72	15.050086	50.983674	3107.2550
1	2005-02	2005	II	42.85	14.461586	50.981004	3119.6576

## ▼ - Ingeniería de características

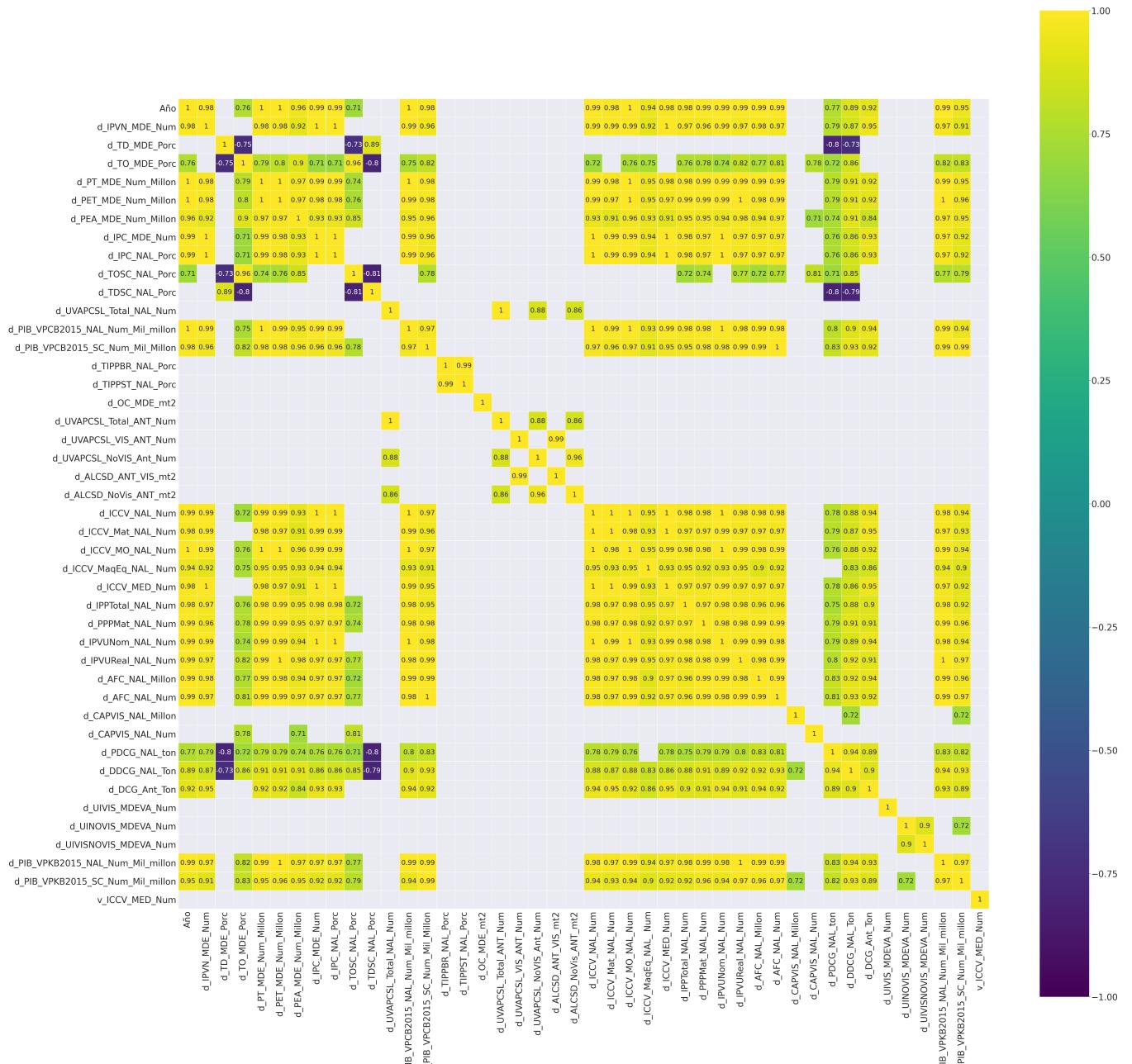
Se hace ingeniería de características sobre las variables a usar en la regresión lineal.

## ▼ - Correlación de variables

```

1 corr_i = df_indices.corr(method="pearson")
2 plt.figure(figsize=(80, 80))
3 sns.set(font_scale=3.5)
4 ax = sns.heatmap(corr_i[(corr_i >= 0.7) | (corr_i <= -0.7)], # 70% de correlación
5                  cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
6                  annot=True, annot_kws={"size": 30}, square=True);

```



```
1 sns.set(font_scale=1.2)
```

```
1 # quitamos las variables de identificación (Periodo, año y trimestre)
2 data = df_indices.iloc[:, 3:]
3 data = data.dropna()
4 data.head()
```

d\_IPVN\_MDE\_Num d\_TD\_MDE\_Porc d\_TO\_MDE\_Porc d\_PT\_MDE\_Num\_Millon d\_PET\_MDE\_Num\_Mil

## ▼ - Escalamiento de datos

	5	46.93	13.864797	49.343463	3166.966666	2589.015
--	---	-------	-----------	-----------	-------------	----------

### Normalización

"sklearn.preprocessing.scale: Standardize a dataset along any axis. Center to the mean and component wise scale to unit variance."

#### Fuente

```

1 columns=data.columns
2 from sklearn.preprocessing import scale
3 data = scale(data)

1 data=pd.DataFrame(data=data,columns=columns)
2 data.head()

```

d\_IPVN\_MDE\_Num d\_TD\_MDE\_Porc d\_TO\_MDE\_Porc d\_PT\_MDE\_Num\_Millon d\_PET\_MDE\_Num\_Mil

0	-1.469822	1.123081	-1.577980	-1.732920	-1.789
1	-1.438816	0.880355	-2.209479	-1.671385	-1.723
2	-1.418146	0.433449	-1.973457	-1.609062	-1.656
3	-1.344546	0.201549	-2.187225	-1.545795	-1.588
4	-1.275331	1.207592	-2.170807	-1.481486	-1.518

## ▼ - Selección de características

Se ejecutan varios métodos de selección de características para usar en el modelo.

#### Referente

## ▼ Regresión lineal con penalización de LASSO

```

1 import statsmodels.api as sm
2 from sklearn.linear_model import LinearRegression
3 from sklearn.feature_selection import RFE
4 from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso

1 #Para ajustar el modelo hay que partir el dataset en variable respuesta y explicativas
2 y1 = data['d_IPVN_MDE_Num']
3 X=data.drop(['d_IPVN_MDE_Num'], axis = 1)
4 X.shape

```

(57, 42)

## Estimación de regresión Lasso con penalización (Método Integrado)

```

1 #Estimación de regresión Lasso con penalización (Método Integrado)
2 reg = LassoCV()
3 reg.fit (X, y1)
4 print ("Mejor alfa usando LassoCV integrado:% f"% reg.alpha_)
5 print ("Mejor puntaje usando LassoCV integrado:% f"% reg.score (X, y1))
6 coef = pd.Series(reg.coef_, index = X.columns, name = "coeficientes")

```

Mejor alfa usando LassoCV integrado: 0.000995  
 Mejor puntaje usando LassoCV integrado: 0.999085

```

1 print ("Lasso recogió " + str(sum(coef != 0)) + " variables y eliminó las otras " + str(sum(coef == 0)) + " variables\n")
2 coef

```

Lasso recogió 18 variables y eliminó las otras 24 variables

d_TD_MDE_Porc	-7.374420e-03
d_TO_MDE_Porc	-1.132905e-02
d_PT_MDE_Num_Millon	1.315097e-01
d_PET_MDE_Num_Millon	0.000000e+00
d_PEA_MDE_Num_Millon	-0.000000e+00
d_IPC_MDE_Num	5.287804e-01
d_IPC_NAL_Porc	3.043375e-18
d_TOSC_NAL_Porc	-4.963915e-02
d_TDSC_NAL_Porc	-0.000000e+00
d_UVAPCSL_Total_NAL_Num	8.905553e-03
d_PIB_VPCB2015_NAL_Num_Mil_million	1.703957e-01
d_PIB_VPCB2015_SC_Num_Mil_Millon	-0.000000e+00
d_TIPPBR_NAL_Porc	-4.056203e-02
d_TIPPST_NAL_Porc	-0.000000e+00
d_OC_MDE_mt2	0.000000e+00
d_UVAPCSL_Total_ANT_Num	0.000000e+00
d_UVAPCSL_VIS_ANT_Num	0.000000e+00
d_UVAPCSL_NoVIS_Ant_Num	0.000000e+00
d_ALCSD_ANT_VIS_mt2	1.376220e-03
d_ALCSD_NoVis_ANT_mt2	-6.835294e-03
d_ICCV_NAL_Num	0.000000e+00
d_ICCV_Mat_NAL_Num	0.000000e+00
d_ICCV_MO_NAL_Num	0.000000e+00
d_ICCV_MaqEq_NAL_Num	-0.000000e+00
d_ICCV_MED_Num	1.456488e-01
d_IPPTotal_NAL_Num	0.000000e+00
d_PPPMat_NAL_Num	0.000000e+00
d_IPVUNom_NAL_Num	0.000000e+00
d_IPVUReal_NAL_Num	-0.000000e+00
d_AFC_NAL_Millon	0.000000e+00
d_AFC_NAL_Num	-0.000000e+00
d_CAPVIS_NAL_Millon	-1.111232e-03
d_CAPVIS_NAL_Num	-4.190502e-02
d_PDCG_NAL_ton	8.377857e-02
d_DDCG_NAL_Ton	-0.000000e+00

```

d_DCG_Ant_Ton          0.000000e+00
d_UIVVIS_MDEVA_Num     -1.574631e-02
d_UINOVIS_MDEVA_Num    -0.000000e+00
d_UIVVISNOVIS_MDEVA_Num -6.744439e-04
d_PIB_VPKB2015_NAL_Num_Mil_million 0.000000e+00
d_PIB_VPKB2015_SC_Num_Mil_million   -0.000000e+00
v_ICCV_MED_Num         -3.440370e-02
Name: coeficientes, dtype: float64

```

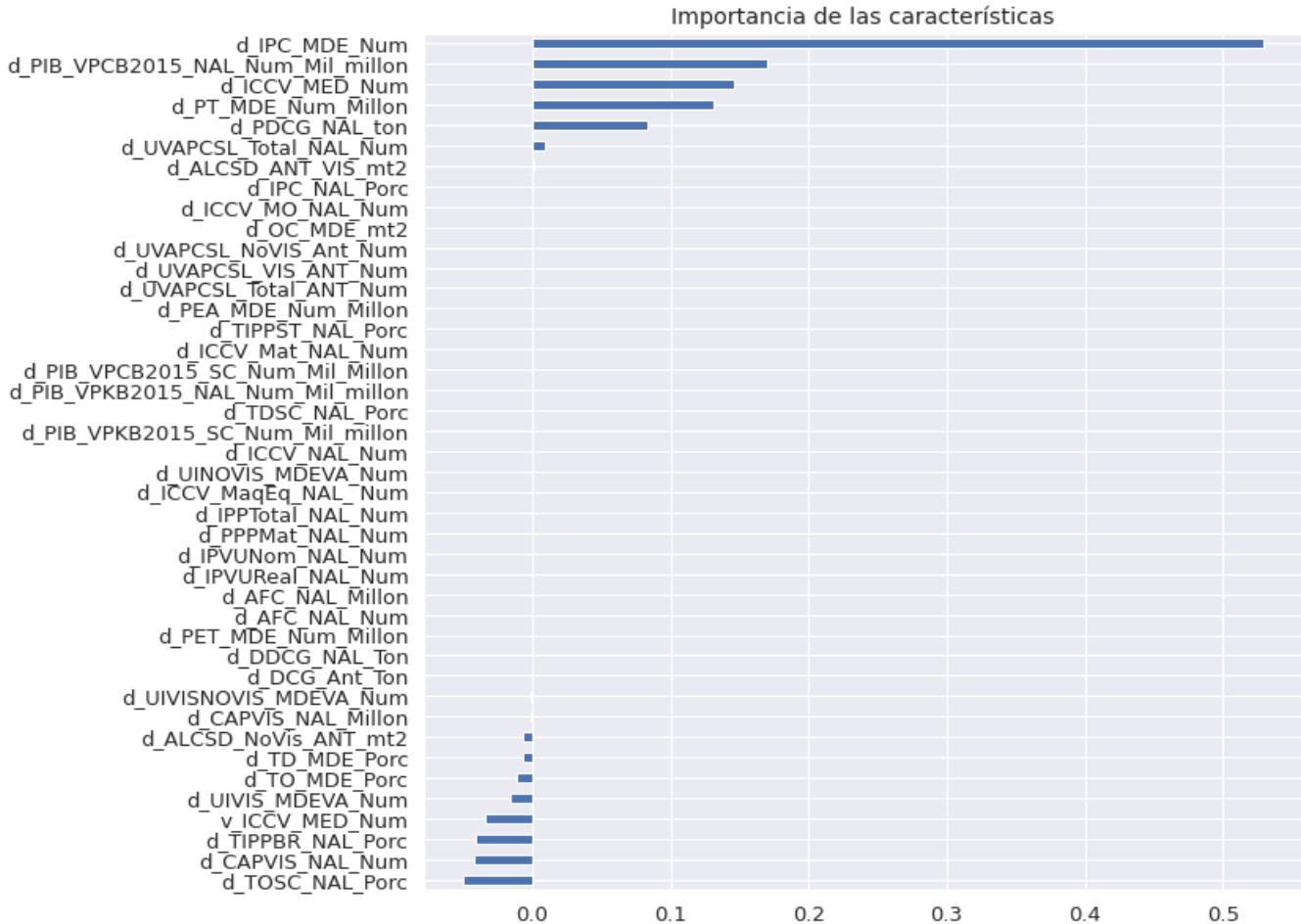
## Importancia de las características usando modelo Lasso

```

1 imp_coef = coef.sort_values()
2 import matplotlib
3 matplotlib.rcParams['figure.figsize'] = (10.0, 10.0)
4 imp_coef.plot(kind = "barh")
5 plt.title("Importancia de las características")

```

Text(0.5, 1.0, 'Importancia de las características')



```
1 coef_sel = coef[coef != 0.0]
```

```

2 coef_sel = list(coef_sel.index)
3 coef_sel

['d_TD_MDE_Porc',
 'd_TO_MDE_Porc',
 'd_PT_MDE_Num_Millon',
 'd_IPC_MDE_Num',
 'd_IPC_NAL_Porc',
 'd_TOSC_NAL_Porc',
 'd_UVAPCSL_Total_NAL_Num',
 'd_PIB_VPCB2015_NAL_Num_Mil_million',
 'd_TIPPBR_NAL_Porc',
 'd_ALCSD_ANT_VIS_mt2',
 'd_ALCSD_NoVis_ANT_mt2',
 'd_ICCV_MED_Num',
 'd_CAPVIS_NAL_Millon',
 'd_CAPVIS_NAL_Num',
 'd_PDCG_NAL_ton',
 'd_UIVIS_MDEVA_Num',
 'd_UIVISNOVIS_MDEVA_Num',
 'v_ICCV_MED_Num']

```

## Dataset con las caracteristicas seleccionadas

```

1 #Dataset con las caracteristicas seleccionadas
2 datam = df_indices.loc[:, coef_sel]
3 datam.head(5)

```

	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_Millon	d_IPC_MDE_Num	d_IPC_NAL_Porc	d_
<b>0</b>	15.050086	50.983674	3107.255000	56.38	56.38	
<b>1</b>	14.461586	50.981004	3119.657667	56.93	56.93	
<b>2</b>	14.811179	50.667551	3131.743000	57.23	57.23	
<b>3</b>	10.760881	53.078909	3143.574667	57.25	57.25	
<b>4</b>	14.317852	51.264581	3155.263667	58.48	58.48	

```

1 a_y_vr = df_indices.loc[:,["Año","Periodo", "d_IPVN_MDE_Num"]]
2 datam = pd.concat([a_y_vr, datam], axis = 1)
3 datam.head()

```

```
Año Periodo d_IPVN_MDE_Num d_TD_MDE_Porc d_TO_MDE_Porc d_PT_MDE_Num_Millon d_
```

```
1 datam.columns
```

```
Index(['Año', 'Periodo', 'd_IPVN_MDE_Num', 'd_TD_MDE_Porc', 'd_TO_MDE_Porc',
       'd_PT_MDE_Num_Millon', 'd_IPC_MDE_Num', 'd_IPC_NAL_Porc',
       'd_TOSC_NAL_Porc', 'd_UVAPCSL_Total_NAL_Num',
       'd_PIB_VPCB2015_NAL_Num_Mil_million', 'd_TIPPBR_NAL_Porc',
       'd_ALCSD_ANT_VIS_mt2', 'd_ALCSD_NoVis_ANT_mt2', 'd_ICCV_MED_Num',
       'd_CAPVIS_NAL_Millon', 'd_CAPVIS_NAL_Num', 'd_PDCG_NAL_ton',
       'd_UIVIS_MDEVA_Num', 'd_UIVISNOVIS_MDEVA_Num', 'v_ICCV_MED_Num'],
      dtype='object')
```

```
1 datam.to_excel ('/content/drive/MyDrive/Proyecto Integrador Semestre 2/Data_primera_seleccion_Lasso.xlsx', index = False)
```

## ▼ 4. Implementación de modelos

### ▼ 4.1 Multivariados

#### ▼ 4.1.1 Regresión lineal

Esta parte del código se realizó en R.

```
1 # activate R magic
2 %load_ext rpy2.ipython
```

```
1 %%R
2 install.packages('dslabs')
3 install.packages('dplyr')
4 install.packages('tidyverse')
5 install.packages('ggplot2')
6 install.packages('magrittr')
7 install.packages('scales')
8 install.packages('gridExtra')
9 install.packages('lmtest')
10 install.packages('corrplot')
11
```

```
1
2 %%R
3 library('dslabs')
4 library('dplyr')
5 library('tidyverse')
6 library('ggplot2')
7 library('scales')
8 library('gridExtra')
9 library('lmtest')
10 library('corrplot')
11 library('magrittr')
12 library('readxl')
```

```

1 %%R
2 dataSet <- read_excel("/content/drive/MyDrive/Proyecto Integrador Semestre 2/datam.xlsx")
3 dataSet

# A tibble: 57 x 24
  Año d_IPVN_MDE_Num d_ICCV_MED_Num d_PT_MDE_Num_Mi... d_PDCG_NAL_ton
  <dbl>      <dbl>      <dbl>          <dbl>          <dbl>
1 2006        45.9       155.         3155.        2357082
2 2006        46.9       158.         3167.        2424563
3 2006        47.6       162.         3179.        2637129
4 2006        49.9       162.         3191.        2619357.
5 2007        52.2       166.         3203.        2552605
6 2007        54.3       167.         3215.        2758299
7 2007        56.5       167.         3227.        2788005
8 2007        57.7       168.         3240.        2968851
9 2008        58.8       174.         3252.        2635273
10 2008       60.3       176.         3264.        2668874
# ... with 47 more rows, and 19 more variables: d_UVAPCSL_Total_NAL_Num <dbl>,
#   d_ALCSD_ANT_VIS_mt2 <dbl>, d_IPC_NAL_Porc <dbl>,
#   d_UIVISNOVIS_MDEVA_Num <dbl>, d_CAPVIS_NAL_Millon <dbl>,
#   d_ALCSD_NoVis_ANT_mt2 <dbl>, d_TD_MDE_Porc <dbl>, d_TO_MDE_Porc <dbl>,
#   d_UIVIS_MDEVA_Num <dbl>, d_TOSC_NAL_Porc <dbl>, d_PEA_MDE_Num_Millon <dbl>,
#   d_IPC_MDE_Num <dbl>, d_PIB_VPCB2015_NAL_Num_Mil_million <dbl>,
#   d_TIPPBR_NAL_Porc <dbl>, d_ICCV_MO_NAL_Num <dbl>, `d_ICCV_MaqEq_NAL_`  

#   Num` <dbl>, d_IPVUNom_NAL_Num <dbl>, d_CAPVIS_NAL_Num <dbl>,
#   v_ICCV_MED_Num <dbl>

```

```

1 %%R
2 #data<-filter(dataSet, Año>=2006 & Año<2020)
3 data<-filter(dataSet, Año>=2006)
4 data <- as.data.frame(data)
5 #data <- subset(data, select = -Periodo)
6 colnames(data)

[1] "Año"
[3] "d_ICCV_MED_Num"
[5] "d_PDCG_NAL_ton"
[7] "d_ALCSD_ANT_VIS_mt2"
[9] "d_UIVISNOVIS_MDEVA_Num"
[11] "d_ALCSD_NoVis_ANT_mt2"
[13] "d_TO_MDE_Porc"
[15] "d_TOSC_NAL_Porc"
[17] "d_IPC_MDE_Num"
[19] "d_TIPPBR_NAL_Porc"
[21] "d_ICCV_MaqEq_NAL_`Num`"
[23] "d_CAPVIS_NAL_Num"

[1] "d_IPVN_MDE_Num"
[3] "d_PT_MDE_Num_Millon"
[5] "d_UVAPCSL_Total_NAL_Num"
[7] "d_IPC_NAL_Porc"
[9] "d_CAPVIS_NAL_Millon"
[11] "d_TD_MDE_Porc"
[13] "d_UIVIS_MDEVA_Num"
[15] "d_PEA_MDE_Num_Millon"
[17] "d_PIB_VPCB2015_NAL_Num_Mil_million"
[19] "d_ICCV_MO_NAL_Num"
[21] "d_IPVUNom_NAL_Num"
[23] "v_ICCV_MED_Num"

```

```

1 %%R
2 #Se normalizan las variables y se guarda de nuevo como un dataframe
3 dataN <- scale(data, center=TRUE, scale=TRUE)
4 dataN1 <- as.data.frame(dataN)

```

```

1 %%R
2 #En el dataset dataN2 se guarda un dataset normalizado con los datos inferiores al año 2018, ya que estos son los que se usarán
3 dataN2<-filter(dataN1, Año<1.1)#Aquí estoy dejando por fuera los años 2018, 2019 y 2020 trimestre I

```

4 #dataN2

## Estimación regresión lineal con las fecaracterísticas escogidas por la regresión con penalización de Lasso

```

1 %%R
2 colnames(dataN2)
3 # colnames(dataN2) <- c("Año", "d_IPVN_MDE_Num", "d_TD_MDE_Porc", "d_TO_MDE_Porc",
4 #                         "d_PT_MDE_Num_Millon", "d_IPC_MDE_Num", "d_IPC_NAL_Porc",
5 #                         "d_TOSC_NAL_Porc", "d_TDSC_NAL_Porc", "d_UVAPCSL_Total_NAL_Num",
6 #                         "d_PIB_VPCB2015_NAL_Num_Mil_million", "d_TIPPBR_NAL_Porc",
7 #                         "d_OC_MDE_mt2", "d_ALCSD_NoVis_ANT_mt2", "d_ICCV_MaqEq_NAL_Num",
8 #                         "d_ICCV_MED_Num", "d_CAPVIS_NAL_Num", "d_PDCG_NAL_ton", "v_ICCV_MED_Num")
9 # colnames(dataN2)
10 colnames(dataN2)[21] <- "d_ICCV_MaqEq_NAL_Num"
11 colnames(dataN2)

```

[1]	"Año"	"d_IPVN_MDE_Num"
[3]	"d_ICCV_MED_Num"	"d_PT_MDE_Num_Millon"
[5]	"d_PDCG_NAL_ton"	"d_UVAPCSL_Total_NAL_Num"
[7]	"d_ALCSD_ANT_VIS_mt2"	"d_IPC_NAL_Porc"
[9]	"d_UIVISNOVIS_MDEVA_Num"	"d_CAPVIS_NAL_Millon"
[11]	"d_ALCSD_NoVis_ANT_mt2"	"d_TD_MDE_Porc"
[13]	"d_TO_MDE_Porc"	"d_UIVIS_MDEVA_Num"
[15]	"d_TOSC_NAL_Porc"	"d_PEA_MDE_Num_Millon"
[17]	"d_IPC_MDE_Num"	"d_PIB_VPCB2015_NAL_Num_Mil_million"
[19]	"d_TIPPBR_NAL_Porc"	"d_ICCV_MO_NAL_Num"
[21]	"d_ICCV_MaqEq_NAL_Num"	"d_IPVUNom_NAL_Num"
[23]	"d_CAPVIS_NAL_Num"	"v_ICCV_MED_Num"

```

1 %%R
2 modeloLasso <- lm(d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
3                         d_ICCV_MED_Num + d_PT_MDE_Num_Millon + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
4                         d_ALCSD_ANT_VIS_mt2 + d_UIVISNOVIS_MDEVA_Num + d_CAPVIS_NAL_Millon + d_ALCSD_NoVis_ANT_mt2 +
5                         d_TD_MDE_Porc + d_TO_MDE_Porc + d_UIVIS_MDEVA_Num + v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num + d_
6
7 summary(modeloLasso)

```

Call:

```

lm(formula = d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
  d_ICCV_MED_Num + d_PT_MDE_Num_Millon + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
  d_ALCSD_ANT_VIS_mt2 + d_UIVISNOVIS_MDEVA_Num + d_CAPVIS_NAL_Millon +
  d_ALCSD_NoVis_ANT_mt2 + d_TD_MDE_Porc + d_TO_MDE_Porc + d_UIVIS_MDEVA_Num +
  v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num + d_TOSC_NAL_Porc,
  data = dataN2)

```

Residuals:

Min	1Q	Median	3Q	Max
-0.041045	-0.014944	-0.000722	0.010403	0.058613

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-0.015476	0.007355	-2.104	0.043861 *
d_IPC_MDE_Num	0.378677	0.079164	4.783	4.29e-05 ***

```

d_PIB_VPCB2015_NAL_Num_Mil_million 0.301106 0.165771 1.816 0.079315 .
d_ICCV_MED_Num                      0.334393 0.110887 3.016 0.005182 **
d_PT_MDE_Num_Millon                 -0.073447 0.168850 -0.435 0.666685
d_PDCG_NAL_ton                      0.047512 0.021151 2.246 0.032198 *
d_UVAPCSL_Total_NAL_Num              0.032352 0.019536 1.656 0.108142
d_ALCSD_ANT_VIS_mt2                 -0.012985 0.009742 -1.333 0.192594
d_UIVISNOVIS_MDEVA_Num               -0.000441 0.007878 -0.056 0.955728
d_CAPVIS_NAL_Millon                 0.021981 0.010256 2.143 0.040316 *
d_ALCSD_NoVis_ANT_mt2                -0.027417 0.016607 -1.651 0.109193
d_TD_MDE_Porc                        -0.014062 0.011688 -1.203 0.238312
d_TO_MDE_Porc                         0.009704 0.021725 -0.447 0.658322
d_UIVIS_MDEVA_Num                   -0.005197 0.008091 -0.642 0.525565
v_ICCV_MED_Num                       -0.037412 0.007912 -4.729 5.01e-05 ***
d_TIPPBR_NAL_Porc                    -0.031091 0.007664 -4.057 0.000326 ***
d_CAPVIS_NAL_Num                     -0.031894 0.013094 -2.436 0.021015 *
d_TOSC_NAL_Porc                      -0.007938 0.022342 -0.355 0.724844
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```

Residual standard error: 0.02611 on 30 degrees of freedom  
 Multiple R-squared: 0.9992, Adjusted R-squared: 0.9988  
 F-statistic: 2252 on 17 and 30 DF, p-value: < 2.2e-16

Con el resultado anterior puede verse que algunas variables no son significativas, por lo que se procedera a realizar una estrategia de stepwise mixto usando la función step() de R. El valor matemático empleado para determinar la calidad del modelo va a ser Akaike(AIC).

```

1 %%%R
2 step(object = modeloLasso, direction = "both", trace = 1)

- d_UVAPCSL_Total_NAL_Num          1 0.0021470 0.023432 -339.99
- d_PDCG_NAL_ton                  1 0.0027047 0.023989 -338.86
- d_PIB_VPCB2015_NAL_Num_Mil_million 1 0.0050900 0.026375 -334.31
- d_CAPVIS_NAL_Millon             1 0.0062736 0.027558 -332.21
- d_CAPVIS_NAL_Num                1 0.0099861 0.031271 -326.14
- d_ICCV_MED_Num                  1 0.0119706 0.033255 -323.19
- d_TIPPBR_NAL_Porc                1 0.0138121 0.035097 -320.60
- d_IPC_MDE_Num                   1 0.0174460 0.038731 -315.87
- v_ICCV_MED_Num                  1 0.0215779 0.042863 -311.01

Step: AIC=-343.27

d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
  d_ICCV_MED_Num + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
  d_ALCSD_ANT_VIS_mt2 + d_CAPVIS_NAL_Millon + d_ALCSD_NoVis_ANT_mt2 +
  d_TD_MDE_Porc + v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num

```

	Df	Sum of Sq	RSS	AIC
<none>		0.021886	-343.27	
- d_TD_MDE_Porc	1	0.0010199	0.022906	-343.08
+ d_UIVIS_MDEVA_Num	1	0.0006017	0.021285	-342.61
+ d_PT_MDE_Num_Millon	1	0.0004449	0.021441	-342.25
+ d_TO_MDE_Porc	1	0.0004301	0.021456	-342.22
+ d_UIVISNOVIS_MDEVA_Num	1	0.0004273	0.021459	-342.21

```
+ d_TOSC_NAL_Porc          1 0.0004091 0.021477 -342.17
- d_ALCSD_ANT_VIS_mt2      1 0.0020816 0.023968 -340.91
- d_ALCSD_NoVis_ANT_mt2    1 0.0023146 0.024201 -340.44
- d_UVAPCSL_Total_NAL_Num  1 0.0025957 0.024482 -339.89
- d_PDCG_NAL_ton           1 0.0030043 0.024891 -339.09
- d_PIB_VPCB2015_NAL_Num_Mil_million 1 0.0048622 0.026749 -335.64
- d_CAPVIS_NAL_Millon      1 0.0078199 0.029706 -330.60
- d_CAPVIS_NAL_Num          1 0.0105876 0.032474 -326.33
- d_ICCV_MED_Num            1 0.0118136 0.033700 -324.55
- d_TIPPBR_NAL_Porc         1 0.0132154 0.035102 -322.59
- d_IPC_MDE_Num             1 0.0181684 0.040055 -316.26
- v_ICCV_MED_Num            1 0.0213176 0.043204 -312.63
```

Call:

```
lm(formula = d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
d_ICCV_MED_Num + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
d_ALCSD_ANT_VIS_mt2 + d_CAPVIS_NAL_Millon + d_ALCSD_NoVis_ANT_mt2 +
d_TD_MDE_Porc + v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num,
data = dataN2)
```

Coefficients:

	(Intercept)	d_IPC_MDE_Num
	-0.02138	0.35396
d_PIB_VPCB2015_NAL_Num_Mil_million	0.19279	0.38113
d_PDCG_NAL_ton	0.04028	d_UVAPCSL_Total_NAL_Num
d_ALCSD_ANT_VIS_mt2	-0.01607	d_CAPVIS_NAL_Millon
d_ALCSD_NoVis_ANT_mt2	-0.02977	d_TD_MDE_Porc
v_ICCV_MED_Num	-0.03396	d_TIPPBR_NAL_Porc
d_CAPVIS_NAL_Num	-0.04131	-0.03126

```
1 %%%R
2 modeloLasso <- lm(d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
3                         d_ICCV_MED_Num + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
4                         d_ALCSD_ANT_VIS_mt2 + d_CAPVIS_NAL_Millon +
5                         d_ALCSD_NoVis_ANT_mt2 + d_TD_MDE_Porc + v_ICCV_MED_Num +
6                         d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num, data=dataN2)
7 summary(modeloLasso)
8
```

Call:

```
lm(formula = d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
d_ICCV_MED_Num + d_PDCG_NAL_ton + d_UVAPCSL_Total_NAL_Num +
d_ALCSD_ANT_VIS_mt2 + d_CAPVIS_NAL_Millon + d_ALCSD_NoVis_ANT_mt2 +
d_TD_MDE_Porc + v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num,
data = dataN2)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.044797	-0.015460	0.000165	0.011190	0.067908

## Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	-0.021380	0.005112	-4.182	0.000184	***
d_IPC_MDE_Num	0.353956	0.065667	5.390	4.92e-06	***
d_PIB_VPCB2015_NAL_Num_Mil_million	0.192786	0.069138	2.788	0.008506	**
d_ICCV_MED_Num	0.381126	0.087686	4.346	0.000113	***
d_PDCG_NAL_ton	0.040283	0.018378	2.192	0.035131	*
d_UVAPCSL_Total_NAL_Num	0.036624	0.017976	2.037	0.049229	*
d_ALCSD_ANT_VIS_mt2	-0.016070	0.008808	-1.825	0.076621	.
d_CAPVIS_NAL_Millon	0.027560	0.007793	3.536	0.001165	**
d_ALCSD_NoVis_ANT_mt2	-0.029767	0.015472	-1.924	0.062528	.
d_TD_MDE_Porc	-0.012236	0.009581	-1.277	0.209982	
v_ICCV_MED_Num	-0.033955	0.005816	-5.839	1.26e-06	***
d_TIPPBR_NAL_Porc	-0.031257	0.006799	-4.597	5.38e-05	***
d_CAPVIS_NAL_Num	-0.041305	0.010038	-4.115	0.000224	***
---					
Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1				

Residual standard error: 0.02501 on 35 degrees of freedom

Multiple R-squared: 0.9992, Adjusted R-squared: 0.9989

F-statistic: 3478 on 12 and 35 DF, p-value: &lt; 2.2e-16

Dado que despues de ajustar el modelo obtenido en steps algunas variables no eran significativas y el ajuste del modelo es elevado, se ha decidido eliminar aquellas variables no significativas, el proceso se realizó de forma iterativa eliminando de a una y estimando de nuevo. El resultado se muestra a continuación.

```

1 %%R
2 modeloLasso <- lm(d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
3                     d_ICCV_MED_Num + d_PDCG_NAL_ton + d_CAPVIS_NAL_Millon + v_ICCV_MED_Num +
4                     d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num, data=dataN2)
5 summary(modeloLasso)

```

## Call:

```
lm(formula = d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PIB_VPCB2015_NAL_Num_Mil_million +
  d_ICCV_MED_Num + d_PDCG_NAL_ton + d_CAPVIS_NAL_Millon + v_ICCV_MED_Num +
  d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num, data = dataN2)
```

## Residuals:

Min	1Q	Median	3Q	Max
-0.056035	-0.016898	-0.000751	0.011252	0.065715

## Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	-0.020043	0.004868	-4.117	0.000193	***
d_IPC_MDE_Num	0.418312	0.062532	6.690	5.73e-08	***
d_PIB_VPCB2015_NAL_Num_Mil_million	0.212772	0.070162	3.033	0.004296	**
d_ICCV_MED_Num	0.293930	0.082039	3.583	0.000932	***
d_PDCG_NAL_ton	0.055273	0.013492	4.097	0.000205	***
d_CAPVIS_NAL_Millon	0.025385	0.007309	3.473	0.001274	**

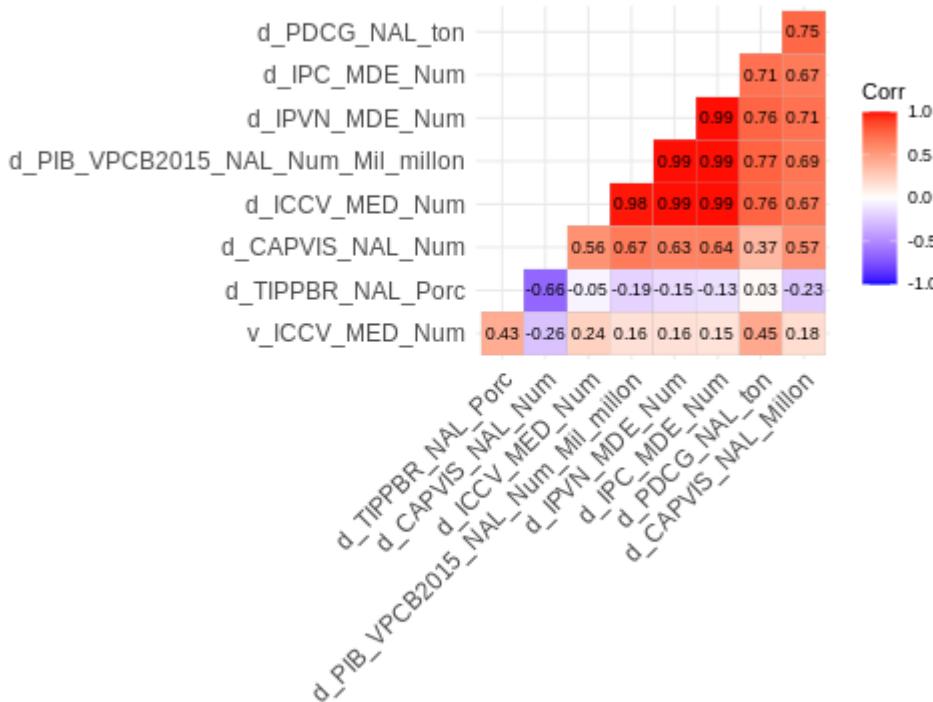
```
v_ICCV_MED_Num      -0.034473  0.005345 -6.449 1.23e-07 ***
d_TIPPBR_NAL_Porc  -0.032088  0.006949 -4.617 4.15e-05 ***
d_CAPVIS_NAL_Num   -0.045930  0.009584 -4.793 2.40e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 0.02607 on 39 degrees of freedom  
 Multiple R-squared: 0.999, Adjusted R-squared: 0.9988  
 F-statistic: 4801 on 8 and 39 DF, p-value: < 2.2e-16

```
1 %%R
2 #dataLM <- data.frame()
3 dataLM <- (subset(dataN2, select = c(d_IPVN_MDE_Num, d_IPC_MDE_Num, d_PIB_VPCB2015_NAL_Num_Mil_million,
4                                     d_ICCV_MED_Num, d_PDCG_NAL_ton, d_CAPVIS_NAL_Million, v_ICCV_MED_Num,
5                                     d_TIPPBR_NAL_Porc, d_CAPVIS_NAL_Num)))
```

```
1 %%R
2 install.packages("ggcorrplot")
3
```

```
1 %%R
2 library("ggcorrplot")
3 ggcorrplot(cor(dataLM), hc.order=TRUE, type='lower', lab = TRUE, digits = 2, lab_size =3)
```



Dada la alta correlación entre algunas de las variables, se han retirado del modelo, a continuación el **modelo finalmente estimado**

```
1 %%%R
2 modeloLasso <- lm(d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PDCG_NAL_ton +
3                      v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num, data=dataN2)
4 summary(modeloLasso)
```

Call:

```
lm(formula = d_IPVN_MDE_Num ~ d_IPC_MDE_Num + d_PDCG_NAL_ton +
  v_ICCV_MED_Num + d_TIPPBR_NAL_Porc + d_CAPVIS_NAL_Num, data = dataN2)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.096679	-0.026502	-0.009114	0.022474	0.083055

Coefficients:

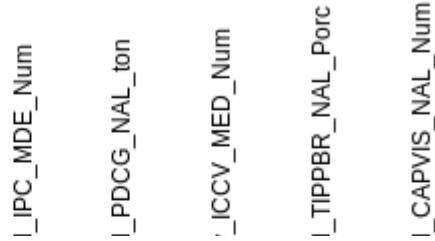
	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-0.019098	0.007264	-2.629	0.011907 *
d_IPC_MDE_Num	0.886701	0.013943	63.596	< 2e-16 ***
d_PDCG_NAL_ton	0.121539	0.010105	12.027	3.44e-15 ***
v_ICCV_MED_Num	-0.028010	0.007330	-3.821	0.000432 ***
d_TIPPBR_NAL_Porc	-0.044660	0.008611	-5.186	5.80e-06 ***
d_CAPVIS_NAL_Num	-0.054548	0.011063	-4.931	1.33e-05 ***
---				
Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .
	1			

Residual standard error: 0.0408 on 42 degrees of freedom

Multiple R-squared: 0.9973, Adjusted R-squared: 0.997

F-statistic: 3130 on 5 and 42 DF, p-value: < 2.2e-16

```
1 %%%R
2 #Matriz de correlación de los nuevos predictores
3 corrplot(corr(dplyr::select(dataN2, d_IPC_MDE_Num, d_PDCG_NAL_ton, v_ICCV_MED_Num, d_TIPPBR_NAL_Porc, d_CAPVIS_NAL_Num)),
4           method = "number", tl.col = "black")
```



## ▼ Validación de supuestos

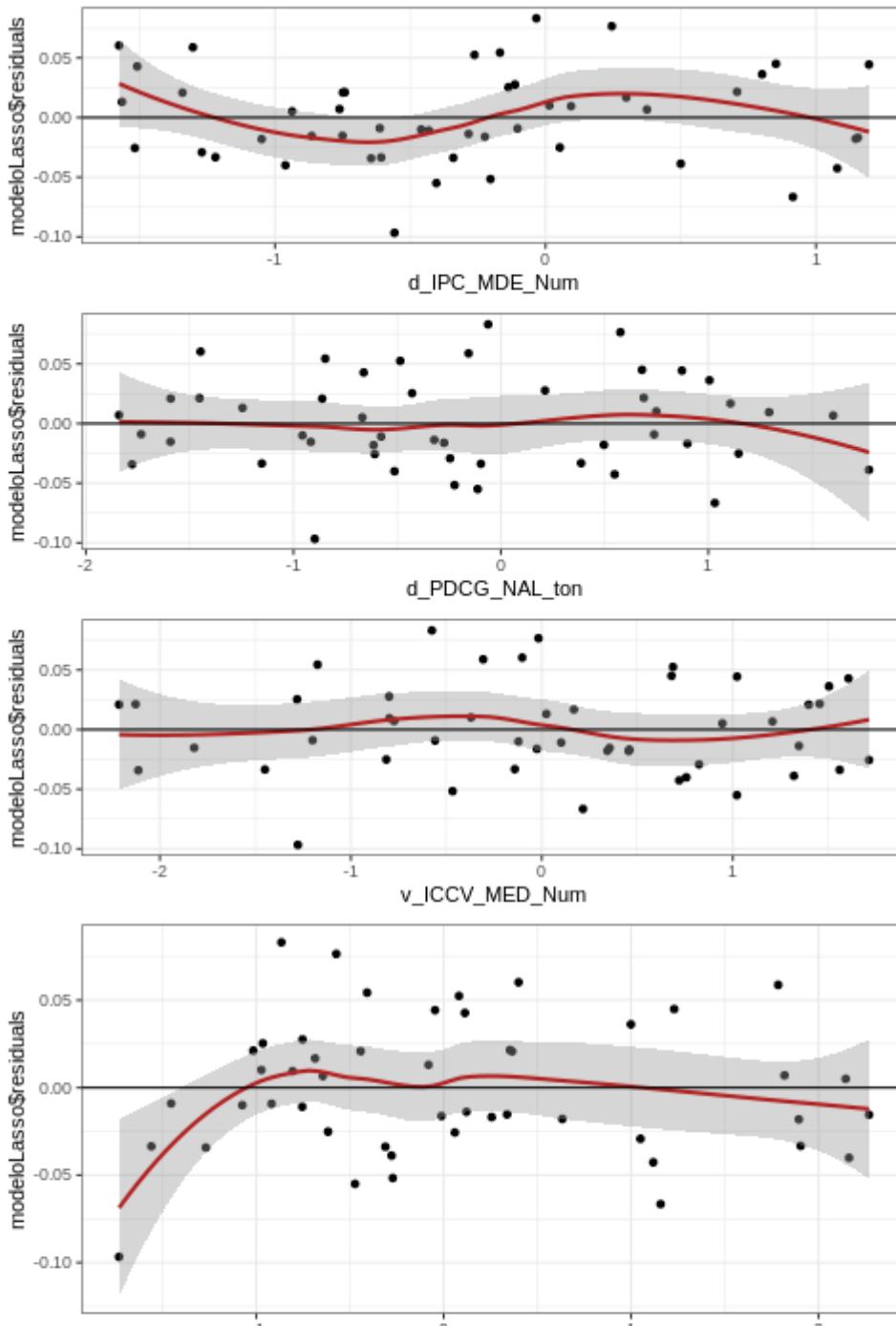
Primero se procede a revisar los diagramas de dispersión entre cada uno de los predictores y los residuos del modelo, en los cuales no se deben apreciar patrones, sino un comportamiento de los residuos aleatorio en torno a 0 con una variabilidad constante a lo largo del eje X. Este análisis se realiza para validar la relación lineal entre los predictores y la variable respuesta.

```
d TIPPBR NAL Porc | -0.12 | -0.42 | 1 | -0.66 | -0.4
```

```
1 %%R
2 plot1 <- ggplot(data = dataN2, aes(d_IPC_MDE_Num, modeloLasso$residuals)) +
3   geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
4   theme_bw()
5 plot2 <- ggplot(data = dataN2, aes(d_PDCG_NAL_ton, modeloLasso$residuals)) +
6   geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
7   theme_bw()
8 plot3 <- ggplot(data = dataN2, aes(v_ICCV_MED_Num, modeloLasso$residuals)) +
9   geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
10  theme_bw()
11 plot4 <- ggplot(data = dataN2, aes(d_TIPPBR_NAL_Porc, modeloLasso$residuals)) +
12  geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
13  theme_bw()
14 plot5 <- ggplot(data = dataN2, aes(d_CAPVIS_NAL_Num, modeloLasso$residuals)) +
15  geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
16  theme_bw()

1 %%R
2 grid.arrange(plot1, plot2, plot3)
3 grid.arrange(plot4, plot5)
```

```
R[write to console]: `geom_smooth()` using method = 'loess' and formula 'y ~ x'
R[write to console]: `geom_smooth()` using method = 'loess' and formula 'y ~ x'
R[write to console]: `geom_smooth()` using method = 'loess' and formula 'y ~ x'
R[write to console]: `geom_smooth()` using method = 'loess' and formula 'y ~ x'
R[write to console]: `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Se concluye que se cumple la linealidad para todos los predictores, al no observarse patrones en los residuales versus cada predictor



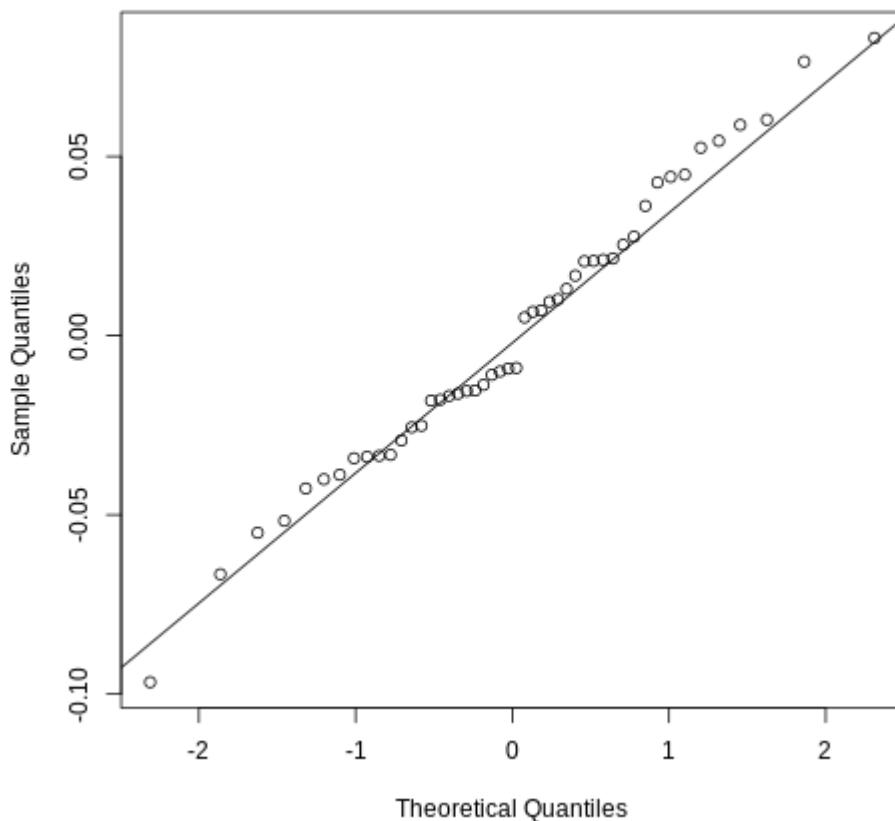
## Normalidad de los residuales

```
1 %%R
2 #Evaluación de normalidad sobre los residuales
3 #Grafico de normalidad de los residuos
4 qqnorm(modeloLasso$residuals)
5 qqline(modeloLasso$residuals)
6 #Test de normalidad
7 shapiro.test(modeloLasso$residuals)
```

Shapiro-Wilk normality test

```
data: modeloLasso$residuals
W = 0.98608, p-value = 0.8338
```

Normal Q-Q Plot



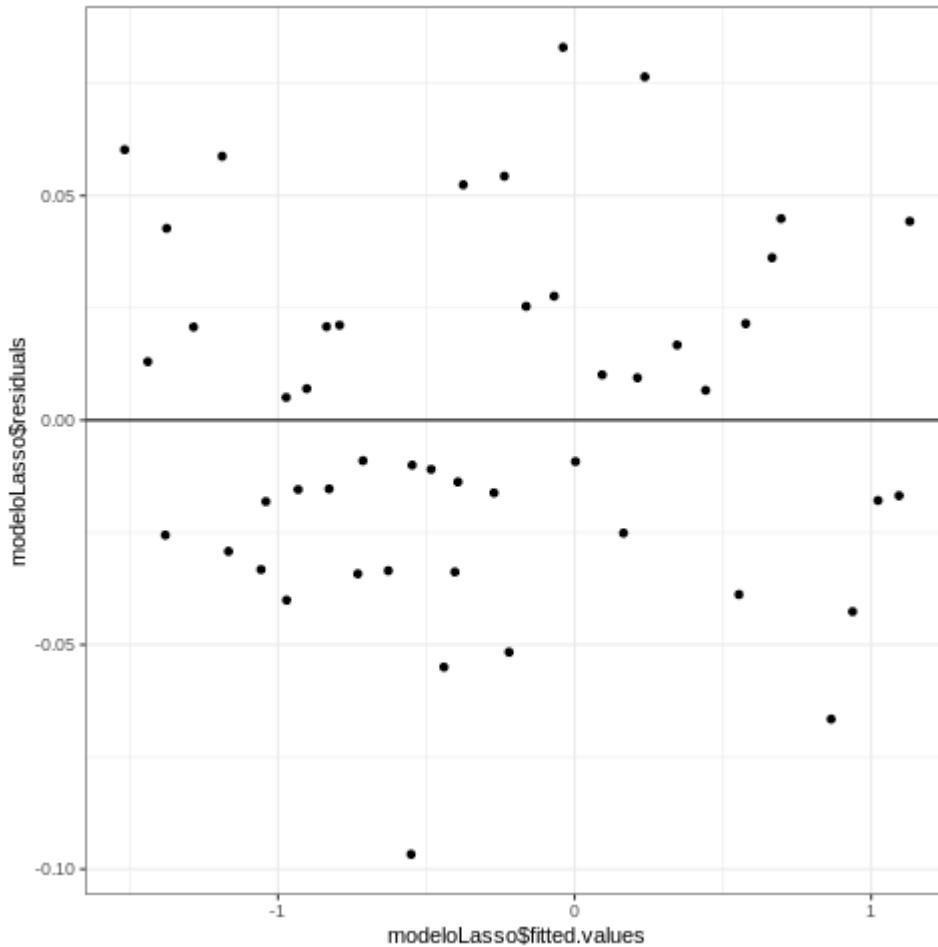
Con el resultado anterior se concluye que no hay evidencia suficiente para negar el supuesto de normalidad de los residuales

Ahora se procede a evaluar la **homocedasticidad** de los residuos, es decir, que tengan una variabilidad constante; para ello se ha realizado la gráfica de los residuales versus los valores ajustados, en esta gráfica busca validarse que no se presenten patrones y que los puntos tengan un comportamiento aleatorio en torno a cero a lo largo del eje X.

```

1 %%R
2 #Homocedasticidad
3 ggplot(data = dataN2, aes(modeloLasso$fitted.values, modeloLasso$residuals)) +
4   geom_point() +
5   geom_hline(yintercept = 0) +
6   theme_bw()

```



```

1 %%R
2 #Test
3 bptest(modeloLasso)

```

studentized Breusch-Pagan test

```

data: modeloLasso
BP = 1.2981, df = 5, p-value = 0.9351

```

No hay evidencias de falta de homocedasticidad.

```

1 %%R
2 #Se cargan las librerías necesarias
3 install.packages('fpp2')
4 install.packages('FitAR')
5 library(fpp2)
6 library(FitAR)

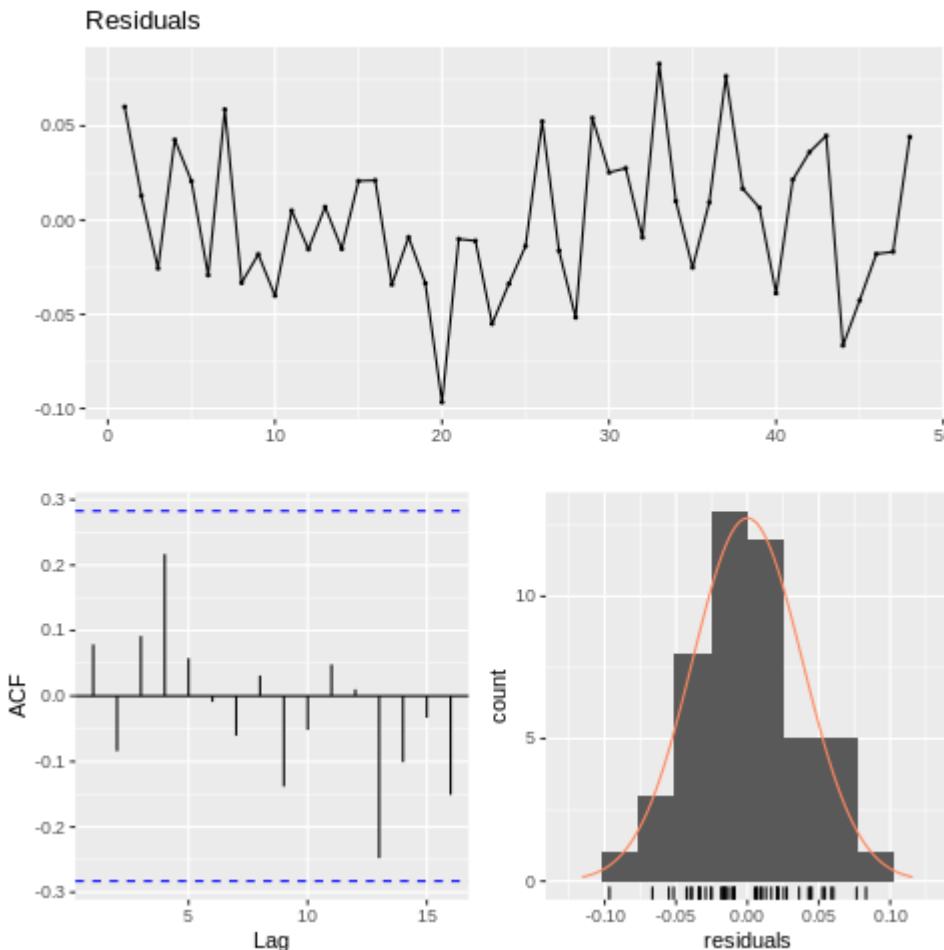
```

Se procede a revisar la autocorrelación de los residuos. Concluyendo con la prueba de Breusch-Godfrey que no hay evidencia de autocorrelación

```
1 %%R
2 checkresiduals(modeloLasso, lag=15) # de la librería forecast
```

Breusch-Godfrey test for serial correlation of order up to 15

```
data: Residuals
LM test = 14.079, df = 15, p-value = 0.5195
```



## ▼ Pronóstico

```
1 %%R
2 #Cálculo de predicciones dataset16
3 nuevo<-filter(dataN1, Año>=1.1 & Año<1.7)%>%select(d_IPC_MDE_Num,d_PDCG_NAL_ton,v_ICCV_MED_Num,d_TIPPBR_NAL_Porc,d_CAPVIS_NAL_Nt)
4 Pre<-predict(modeloLasso, nuevo)
5 Real<-filter(dataN1,(dataN1$Año>=1.1 & dataN1$Año<1.7))%>%select(d_IPVN_MDE_Num)
```

```
1 %%R
2 Pre
```

1	2	3	4	5	6	7	8
1.204230	1.290147	1.364085	1.465985	1.511557	1.627965	1.761075	1.842399

```

1 %%R
2 PreR<-matrix(c(Pre[1],Pre[2],Pre[3],Pre[4],Pre[5],Pre[6],Pre[7],Pre[8]),nrow=8,byrow=T)
3 colnames(PreR)<-c("d_IPVN_MDE_Num_Estimado")
4 PreR<-as.data.frame(PreR)
5 PreR

```

d_IPVN_MDE_Num_Estimado
1 1.204230
2 1.290147
3 1.364085
4 1.465985
5 1.511557
6 1.627965
7 1.761075
8 1.842399

```

1 %%R
2 M<-mean(data$d_IPVN_MDE_Num)
3 St<-sd(data$d_IPVN_MDE_Num)
4 PreRC<-(PreR$d_IPVN_MDE_Num_Estimado*St)+M
5 PreRC<-as.data.frame(PreRC)
6 PreRC

```

PreRC
1 131.6626
2 134.4303
3 136.8121
4 140.0946
5 141.5626
6 145.3125
7 149.6004
8 152.2201

```

1 %%R
2 RealC<-(Real$d_IPVN_MDE_Num*St)+M
3 RealC<-as.data.frame(RealC)
4 RealC

```

RealC
1 132.79
2 137.56
3 139.81
4 145.35
5 146.34
6 149.84
7 152.22
8 154.57

```

1 %%R
2 #MAPE
3 #mean(abs((Real$d_IPVN_MDE_Num-PreR$d_IPVN_MDE_Num_Estimado)/Real$d_IPVN_MDE_Num)) * 100
4 MAPE_RL=mean(abs((RealC$RealC-PreRC$PreRC)/RealC$RealC)) * 100
5 MAPE RL

```

```
[1] 2.30142
```

```
1 MAPE_RL=%%R MAPE_RL  
2 MAPE_RL=MAPE_RL[0]
```

```
1 tabla = pd.DataFrame(columns = ['Modelo','Error porcentual medio absoluto - MAPE'])  
2 tabla.loc[1] = ["Regresión lineal multivariada", MAPE_RL.round(3)]
```

```
1 %%R  
2 #RMSE  
3 #calculate MSE  
4 #sqrt(mean((Real$d_IPVN_MDE_Num-PreR$d_IPVN_MDE_Num_Estimado)^2))  
5 sqrt(mean((RealC$RealC-PreRC$PreRC)^2))
```

```
[1] 3.594748
```

#### ▼ 4.1.2 Red Neuronal (Multivariada)

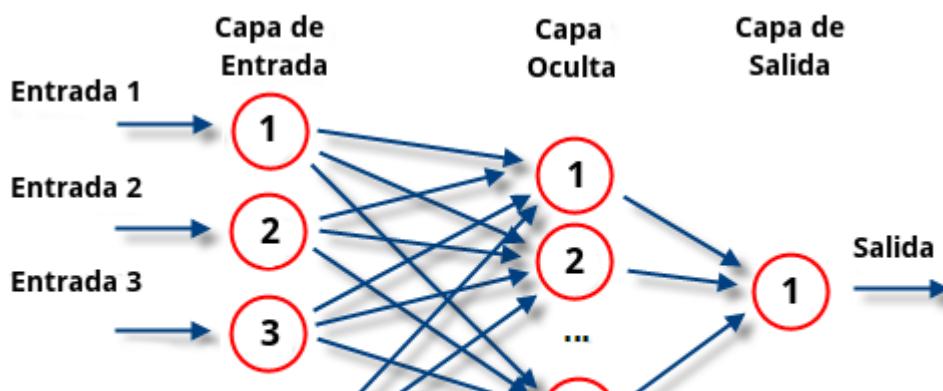
Con el fin de buscar diferentes alternativas para realizar una predicción lo suficientemente cercana a la realidad sin tener sobreajustes, también exploramos la alternativa de emplear Redes Neuronales (RNA).

Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano. Esta formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida.

Su objetivo es aprender modificándose automáticamente a si mismo de forma que puede llegar a realizar tareas complejas que no podrían ser realizadas mediante programación basada en reglas. Aplicada a la ciencia de datos, las redes neuronales pueden realizar predicciones gracias a su capacidad generalizadora por el hecho de aprender a partir de ejemplos, tal y como lo hace el cerebro humano.

Particularmente para nuestro caso de estudio, queremos realizar una evaluación de las variables que, en principio, consideramos tenían incidencia sobre el índice de precios de la vivienda nueva en la ciudad de Medellín; para posteriormente determinar que variables tienen mayor incidencia y a su vez explican mejor nuestra variable objetivo para predecir valores futuros.

[Referencia de código](#)



## ▼ Análisis exploratorio



Las variables utilizadas en este modelo son aquellas seleccionadas en la regresión lineal, ya que allí se evidenció su significancia y correlación.

Para recordar el detalle nuestras variables realizamos un análisis exploratorio con diagramas de caja y bigotes y dispersión.

```
1 df_indices = df_indices.reset_index()
```

```
1 df_indices = df_indices.drop('index', axis=1)
```

```
1 # Para la red neuronal se toman las variables que se usaron finalmente para la regresión lineal.
2 #df_indices.head()
3 df_i_vs = df_indices[['d_IPVN_MDE_Num', 'd_IPC_MDE_Num', 'd_PIB_VPCB2015_NAL_Num_Mil_million',
4                         'd_ICCV_MED_Num', 'd_PDCG_NAL_ton',
5                         'd_CAPVIS_NAL_Millon', 'd_CAPVIS_NAL_Num',
6                         'v_ICCV_MED_Num', 'd_TIPPBR_NAL_Porc']]
```

```
1 df_i_vs.head()
```

	d_IPVN_MDE_Num	d_IPC_MDE_Num	d_PIB_VPCB2015_NAL_Num_Mil_million	d_ICCV_MED_Num	d_
0	41.72	56.38		81552.57905	150.151740
1	42.85	56.93		84065.50848	152.496436
2	44.81	57.23		85285.20488	151.023240
3	45.80	57.25		87054.70825	151.344183
4	45.94	58.48		90022.40177	155.428257

```
1 df_i_vs.dropna(inplace=True)
```

```
1 df_i_vs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 57 entries, 4 to 60
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   d_IPVN_MDE_Num    57 non-null    float64 
 1   d_IPC_MDE_Num     57 non-null    float64 
 2   d_PIB_VPCB2015_NAL_Num_Mil_million 57 non-null    float64 
 3   d_ICCV_MED_Num    57 non-null    float64 
 4   d_PDCG_NAL_ton    57 non-null    float64 
 5   d_CAPVIS_NAL_Millon 57 non-null    float64 
 6   d_CAPVIS_NAL_Num   57 non-null    float64 
 7   v_ICCV_MED_Num    57 non-null    float64 
 8   d_TIPPBR_NAL_Porc 57 non-null    float64 
dtypes: float64(9)
memory usage: 4.5 KB
```

Consideramos para este ejercicio, que nuestros datos de entrenamiento sean los primeros trimestres de los años con que contamos información en nuestro DataSet (Desde 2006-01 hasta 2018-01). Como datos de pruebas dejamos los ultimos 8 trimestres comprendiendo entre el 2018-02 al 2020-01

```
1 from sklearn.model_selection import train_test_split
2 #Si en el dataset se utilizan Indices
3 X = df_i_vs.loc[:, df_i_vs.columns != 'd_IPVN_MDE_Num']
4 y = df_i_vs.loc[:, df_i_vs.columns == 'd_IPVN_MDE_Num']
5
6 #Si en el dataset se utilizan variaciones
7 #X = df.loc[:, df.columns != 'v_IPVN_MDE_Num']
8 #y = df.loc[:, df.columns == 'v_IPVN_MDE_Num']

1 # Estás lineas son para entregar el entrenamiento los primeros años, y el testeo los últimos dos años
2
3 X_train = X.iloc[0:49,:]
4 X_test = X.iloc[49:57,:]
5 y_train = y.iloc[0:49,:]
6 y_test = y.iloc[49:57,:]
7

1 mean = X_train.mean(axis=0)
2 std = X_train.std(axis=0)
3
4 X_train = (X_train - mean) / std
5 X_test = (X_test - mean) / std
```

## ▼ Implementación del modelo

Debido a la pequeña cantidad de datos presentados en este conjunto de datos, debimos tener cuidado de no crear un modelo demasiado complejo, lo que podría llevar a sobreajustar nuestros datos. Para ello adoptamos una arquitectura basada en dos capas densas, la primera con 128 y la segunda con 64 neuronas, ambas utilizando una función de activación ReLU (Rectified Linear Unit).

Se utilizó una capa densa con una activación lineal como capa de salida. Para permitirnos saber si el modelo está aprendiendo correctamente, se usó una función de pérdida de error cuadrático medio y para reportar su desempeño se adoptó la métrica de error promedio promedio. Al utilizar el método de resumen de Keras, se pudo podemos ver que se tenía un total de 14.337 parámetros, lo cual se consideró aceptable.

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()

model.add(Dense(1024, input_shape=(5, ), activation='relu', name='dense_1'))
model.add(Dense(512, activation='relu', name='dense_2'))
model.add(Dense(1, activation='relu', name='dense_output'))
# model.add(Dense(1, activation='softmax', name='dense_output'))

model.compile(optimizer='adam', loss='mse', metrics=['mape'])
model.summary()
```

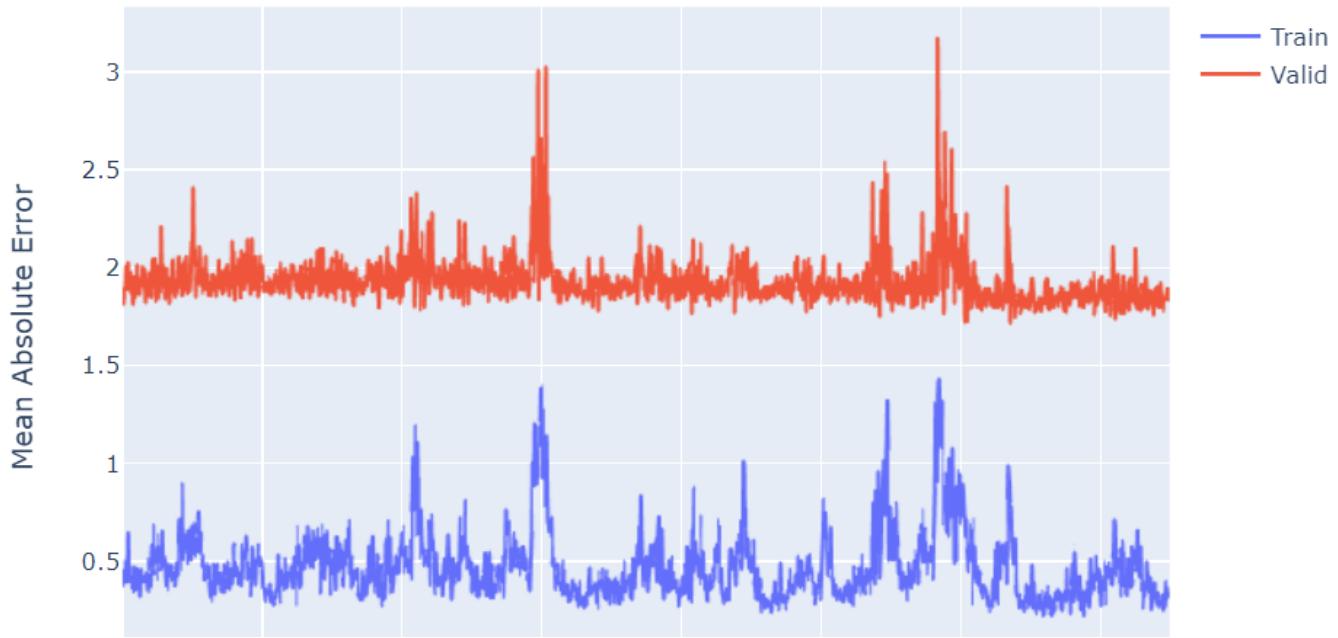
Se realiza un ciclo para entrenar y evaluar el modelo, bien sea con 40 iteraciones o hasta que el MAP

```
# 1. hasta que el mape sea menor a 1
# si llega a x repeticiones

i = 1
mae_nn = 100
while i <= 10 or mae_nn > 0.7:
    history = model.fit(X_train, y_train, epochs=1000, validation_split=0.05)
    mse_nn, mae_nn = model.evaluate(X_test, y_test)
    print("repeticiones: " + str(i*1000) + " MAPE: " + str(mae_nn))
    i += 1
```

Para ver el código completo del código, está disponible [aqui](#).

A continuación se muestra la gráfica de la métrica "Error Porcentual Absoluto Medio (MAPE)" y su comportamiento conforme se va entrenando el modelo en las iteraciones definidas. En este caso también se puede observar que el valor disminuye a lo largo del avance en las iteraciones.



```

1 # Recrea exactamente el mismo modelo desde el archivo
2 from tensorflow import keras
3 from tensorflow.keras import layers
4 #new_model = keras.models.load_model('/content/drive/MyDrive/Proyecto Integrador Semestre 2/modelo/modelo2711.h5')
5 new_model = keras.models.load_model('/content/drive/MyDrive/Proyecto Integrador Semestre 2/modelo/modelo_ANM_MV_0212.h5')

```

Una vez entrenado nuestro modelo, evaluamos los resultados para identificar que tan confiables serán sus predicciones.

La metrica seleccionada fue Error Porcentual Absoluto Medio (MAPE). Es un indicador del desempeño del pronóstico que mide el tamaño del error (absoluto) en términos porcentuales. El modelo nos entrega un MAPE de 2.281. El error cuadrático medio fue de 5,25.

En conclusión nuestro modelo se ajusta adecuadamente y con las variables seleccionadas puede predecir acertadamente el indice de precios de la vivienda nueva en la ciudad de Medellín.

```

1 # Se hacen las predicciones
2 trainPredict = pd.DataFrame(new_model.predict(X_train))
3 testPredict = pd.DataFrame(new_model.predict(X_test))

1 def mape(actual, pred):
2     actual, pred = np.array(actual), np.array(pred)
3     return np.mean(np.abs((actual - pred) / actual)) * 100

1 testPredict = pd.DataFrame(testPredict)
2 testPredict = testPredict.rename(columns={0: "Prediccion"})
3 testPredict['Prediccion']

0    138.319901
1    142.419022

```

```

2    146.069702
3    148.767838
4    154.055344
5    159.121201
6    162.817276
7    159.369537
Name: Prediccion, dtype: float32

```

```

1 import numpy as np
2 MAPE_RNM=mape(y_test,testPredict)
3 MAPE_RNM

```

2.2836401662328893

```
1 tabla.loc[2] = ["Red neuronal multivariada", MAPE_RNM.round(3)]
```

```

1 # RESULTADOS DE LA COMPARACIÓN ENTRE LA PREDICCIÓN Y EL VALOR REAL, DATOS DE TESTEO
2 df_Result = pd.DataFrame()
3 #df_Result['Dato Real'] = y_test['v_IPVN_MDE_Num'] #Para variaciones
4 df_Result['Dato Real'] = y_test['d_IPVN_MDE_Num'] #Para indices
5 df_Result['Prediccion'] = np.asarray(testPredict['Prediccion'])
6

```

```

1 df_Result['Periodo'] = np.array((df_encabezado.iloc[53:61,0]))
2 df_Result

```

	Dato Real	Prediccion	Periodo
<b>53</b>	137.56	138.319901	2018-02
<b>54</b>	139.81	142.419022	2018-03
<b>55</b>	145.35	146.069702	2018-04
<b>56</b>	146.34	148.767838	2019-01
<b>57</b>	149.84	154.055344	2019-02
<b>58</b>	152.22	159.121201	2019-03
<b>59</b>	154.57	162.817276	2019-04
<b>60</b>	157.77	159.369537	2020-01

## ▼ Explicación del modelo

```
1 !pip install shap
```

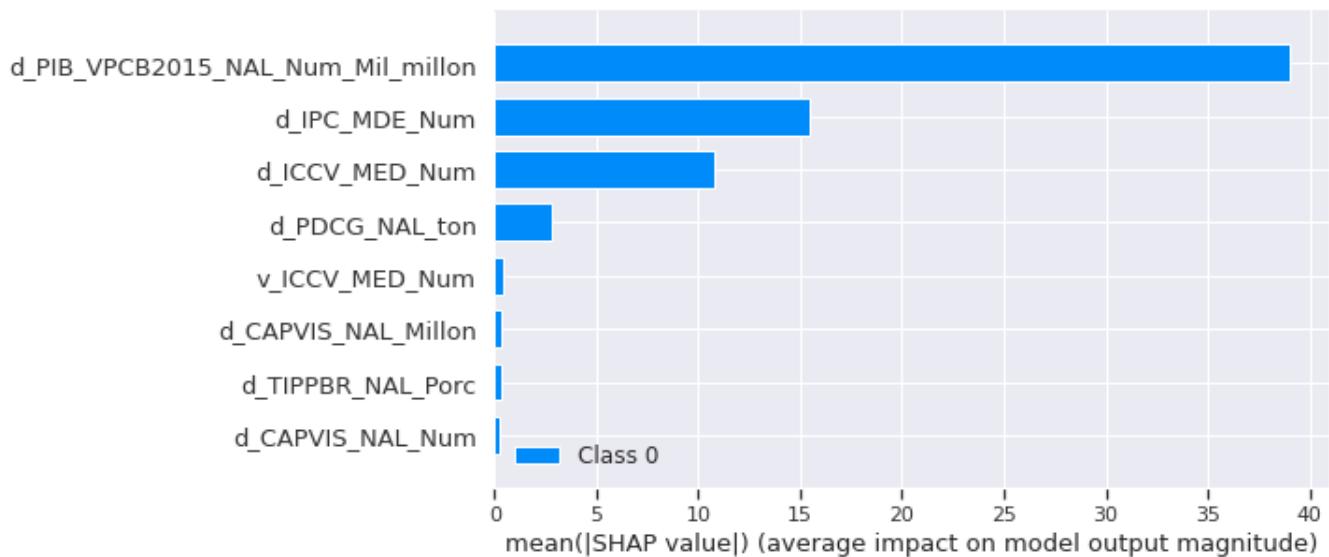
La siguiente grafica la consideramos bastante relevante, pues si bien las redes neuronales son llamadas cajas negras debido a que no se conoce con certeza cómo el algoritmo aprende y toma decisiones; si podemos conocer cuales son las variables más relevantes y que tomaron un mayor protagonismo a la hora de realizar el entrenamiento y su posterior predicción.

Esto quiere decir que para la red neuronal el PIB Nacional es la variable que por una considerable ventaja explica el comportamiento de nuestra variable objetivo; el indice de precios de la vivienda nueva en la ciudad de Medellín.

```
1 import shap
2 shap.initjs()
3 explainer = shap.DeepExplainer(new_model, X_train[:100].values)
4 shap_values = explainer.shap_values(X_test[:100].values)
5 shap.summary_plot(shap_values, X_test, plot_type='bar')
```



WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/shap/explainers/\_deep/dee  
Instructions for updating:  
Simply pass a True/False value to the `training` argument of the `\_\_call\_\_` method of yo



```
1 !pip install chart_studio
```

```
1 Real_data = go.Scatter(x = df_Result.Periodo,y = df_Result['Dato Real'])
```

```
1 def configure_plotly_browser_state():
2     import IPython
3     display(IPython.core.display.HTML('''
4         <script src="/static/components/requirejs/require.js"></script>
5         <script>
6             requirejs.config({
7                 paths: {
8                     base: '/static/base',
9                     plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
10                },
11            });
12        </script>
13    '''))
```

```
1 # Create the steam data object
```

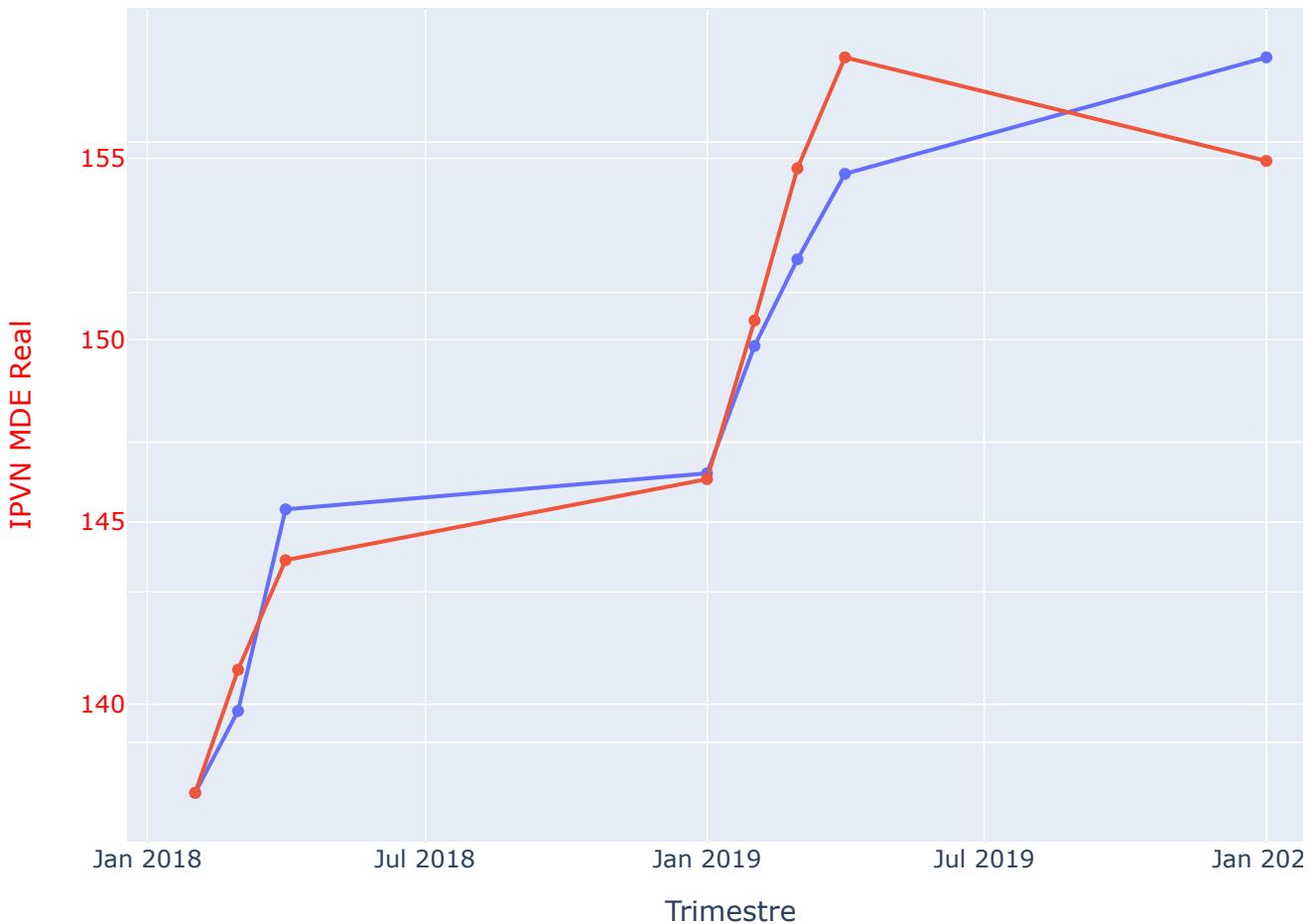
```

2 Predicho_data = go.Scatter(x=df_Result.Periodo,
3                             y=df_Result.Prediccion,
4                             # Specify axis
5                             yaxis='y2')

1 import chart_studio.plotly as py
2 import numpy as np
3 from plotly.offline import init_notebook_mode, iplot
4 import plotly.graph_objs as go
5 configure_plotly_browser_state()
6 init_notebook_mode(connected=False)
7 layout = go.Layout(height=600, width=800,
8                     title='IPVN MDE REAL vs IPVN MDE PREDICHO',
9                     # Same x and first y
10                    xaxis=dict(title='Trimestre'),
11                    yaxis=dict(title='IPVN MDE Real', color='red'),
12                    # Add a second yaxis to the right of the plot
13                    yaxis2=dict(title='IPVN MDE Predicho', color='blue',
14                               overlaying='y', side='right')
15                  )
16 fig = go.Figure(data=[Real_data, Predicho_data], layout=layout)
17 iplot(fig)

```

## IPVN MDE REAL vs IPVN MDE PREDICHO



## ▼ 4.2 Modelos Univariados

### ▼ 4.2.1 Serie de tiempo

La metodología implementada a partir de Elkin Castaño (2020), memorias del curso de la maestría de Series de tiempo Universidad Eafit

```
1 # activate R magic
2 %load_ext rpy2.ipython
```

The rpy2.ipython extension is already loaded. To reload it, use:  
`%reload_ext rpy2.ipython`

```
1 %%R
2 #Se cargan las librería necesarias
3 install.packages('fpp2')
4 install.packages('fpp2', dependencies = TRUE)
5 install.packages('urca')
6 install.packages('urca', dependencies = TRUE)
7 install.packages('TSA')
8 install.packages('TSA', dependencies = TRUE)
9 install.packages('FitAR')
10 install.packages('FitAR', dependencies = TRUE)
11 install.packages('fBasics')
12 install.packages('fBasics', dependencies = TRUE)
13 library(fpp2)
14 library(TSA)
15 library(FitAR)
16 library(urca)
17 library(fBasics)
```

```
1 %%R
2 (ipvn=ts(scan("/content/drive/MyDrive/Proyecto Integrador Semestre 2/otros Datasets/IPVNPF.txt"),start=2006, frequency=4))
```

R[write to console]: Read 56 items

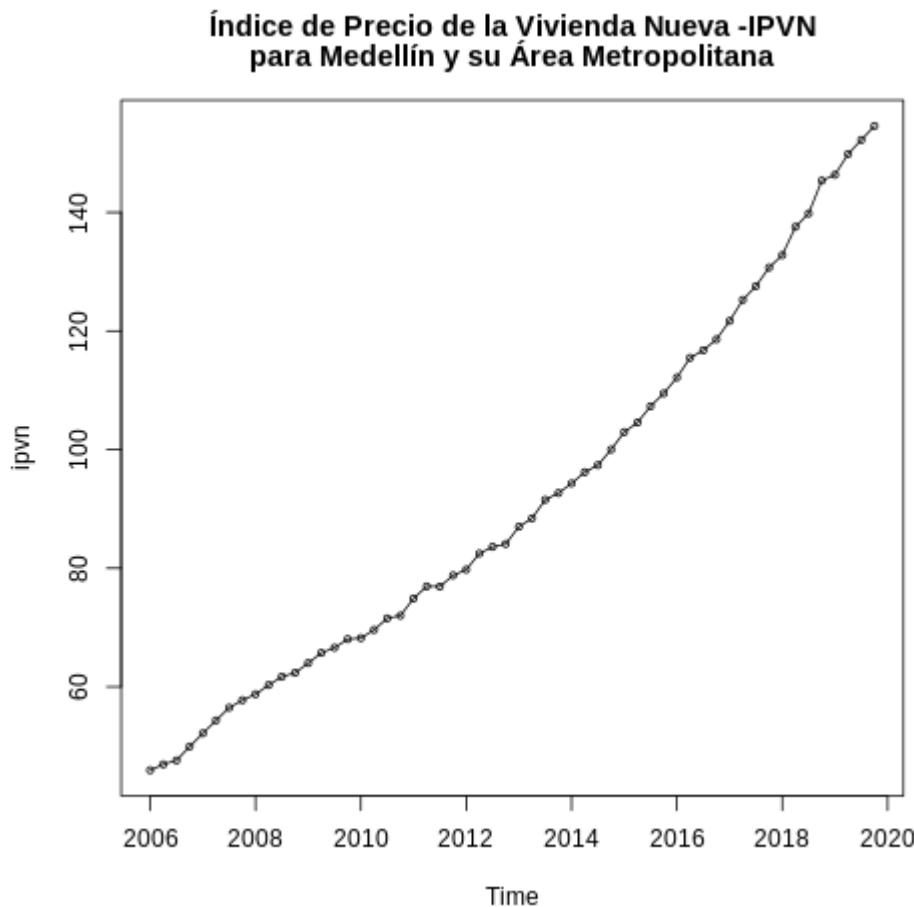
	Qtr1	Qtr2	Qtr3	Qtr4
2006	45.94	46.93	47.59	49.94
2007	52.15	54.34	56.48	57.74
2008	58.75	60.30	61.71	62.35
2009	64.00	65.72	66.60	68.03
2010	68.22	69.60	71.55	72.00
2011	74.90	76.94	76.91	78.78
2012	79.76	82.47	83.60	84.09
2013	86.99	88.42	91.54	92.67
2014	94.30	96.20	97.38	100.00
2015	102.96	104.55	107.32	109.48
2016	112.15	115.48	116.74	118.59
2017	121.69	125.25	127.57	130.70
2018	132.79	137.56	139.81	145.35
2019	146.34	149.84	152.22	154.57

## ▼ Identificación

Inicialmente, se parte del análisis gráfico de la serie de tiempo.

```
1 %%R
```

```
2 plot.ts(ipvn, type="o", cex=0.6, main=c("Índice de Precio de la Vivienda Nueva -IPVN", "para Medellín y su Área Metropolitana"))
```



Se evidencia que es una serie con tendencia determinística, por ende, no estacionaria.

## ▼ Análisis de la estabilidad de varianza

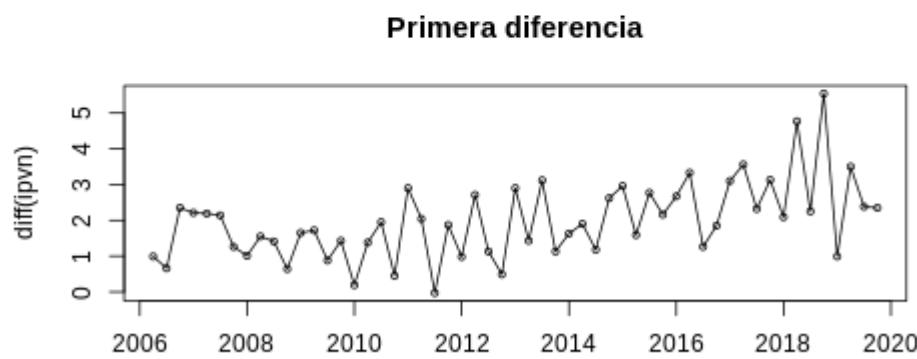
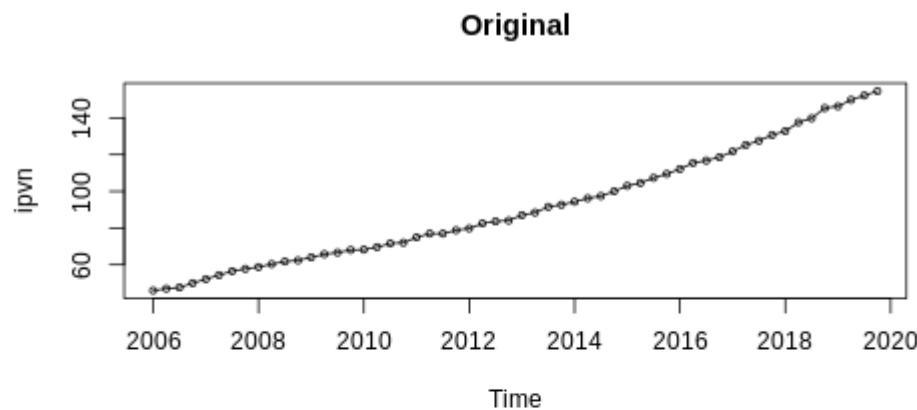
Se hace un primer análisis gráfico

```
1 %%R
```

```
2 par(mfrow=c(2,1))
```

```
3 plot.ts(ipvn, type="o", cex=0.6, main= "Original")
```

```
4 plot.ts(diff(ipvn),type="o", cex=0.6, main="Primera diferencia")
```



Pareciera que la serie no tiene estabilidad en varianza. Además, con la primera diferencia se observa que la serie tiende a convertirse en estacionaria.

A continuación, se aplica la técnica de Box - Cox para analizar si la serie requiere una transformación para estabilizar su varianza. Para esto, se utilizan varias librerías y marcos de análisis.

```
1 %%%R
2 # Datos dependientes
3 BoxCox.lambda(ipvn, method = c("loglik"), lower = -2, upper = 2)
```

```
[1] 0.1
```

En este caso, se evidencia que la serie requiere una transformación, que pareciera ser la logarítmica.

```
1 %%%R
2 # Minimizando el coeficiente de variación
3 BoxCox.lambda(ipvn, method = c("guerrero"), lower = -2, upper = 2)
```

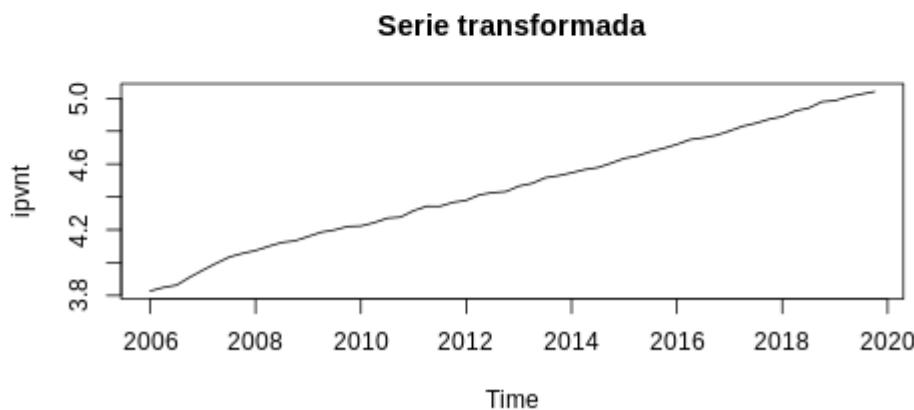
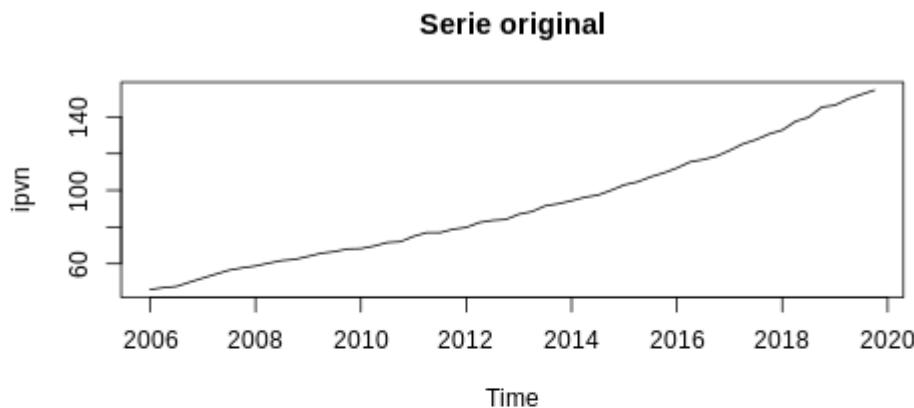
```
[1] 0.1724044
```

Por ende, se aplicará la transformación lambda=0, es decir, se aplica logaritmo a la serie original.

```
1 %%R
2 ipvnt=(log(ipvn))
```

Ahora, se analiza gráficamente si la transformación genera un cambio visual importante.

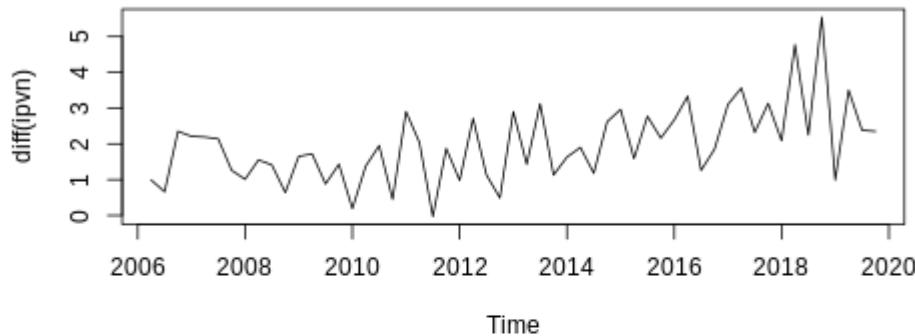
```
1 %%R
2 par(mfrow=c(2,1))
3 plot.ts(ipvn, main="Serie original")
4 plot.ts(ipvnt,main="Serie transformada")
```



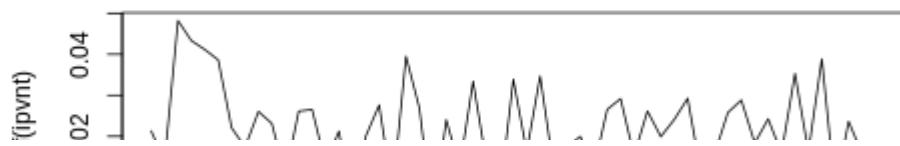
Se observa una leve mejoría, aunque no tan marcada visualmente.

```
1 %%R
2 par(mfrow=c(2,1))
3 plot.ts(diff(ipvn), main="Serie original en diferencias")
4 plot.ts(diff(ipvnt),main="Serie transformada en diferencias")
```

### Serie original en diferencias



### Serie transformada en diferencias



Se observa como con la transformación más la primera diferencia, la serie pareciera volverse estacionaria

A continuación, se determinará el valor de d (integración de orden d)

```

1 %%R
2 # correlogramas muestrales de la serie original
3 (max_rezag=round(length(ipvn)/4))
4 par(mfrow=c(2,1))
5 Acf(ipvnt, lag.max=max_rezag, ylim=c(-1,1))
6 Pacf(ipvnt, lag.max=max_rezag, ylim=c(-1,1))

```

### Series ipvnt



Se observa que la serie no es estacionaria. Especialmente, porque la ACF no tiende exponencialmente a cero, y la PACF se centra en el primer rezago.

1 50

### ▼ Prueba de raíces unitarias

#### Series ipvnt

```
1 %%R
2 (maxlag=floor(12*(length(ipvnt)/100)^(1/4)))

[1] 10
```

```
1 %%R
2 ru_ipvnt=ur.df(ipvnt, type = c("trend"), lags=maxlag, selectlags = c("AIC")) # Usando AIC
3 summary(ru_ipvnt)
```

```
#####
# Augmented Dickey-Fuller Test Unit Root Test #
#####
```

Test regression trend

Call:  
`lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)`

Residuals:

Min	1Q	Median	3Q	Max
-0.014811	-0.004543	-0.000145	0.004277	0.017818

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	0.502357	0.492714	1.020	0.31438
z.lag.1	-0.118212	0.128684	-0.919	0.36409
tt	0.002676	0.002638	1.015	0.31672
z.diff.lag1	-0.529596	0.180332	-2.937	0.00561 **
z.diff.lag2	-0.327893	0.194816	-1.683	0.10056
z.diff.lag3	-0.278702	0.190013	-1.467	0.15067
z.diff.lag4	-0.380709	0.167391	-2.274	0.02868 *

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.008048 on 38 degrees of freedom  
Multiple R-squared: 0.3871, Adjusted R-squared: 0.2904  
F-statistic: 4.001 on 6 and 38 DF, p-value: 0.003359

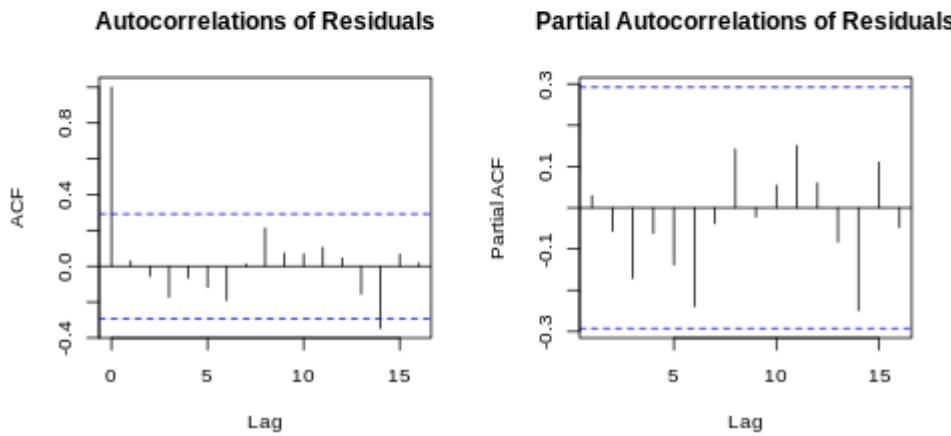
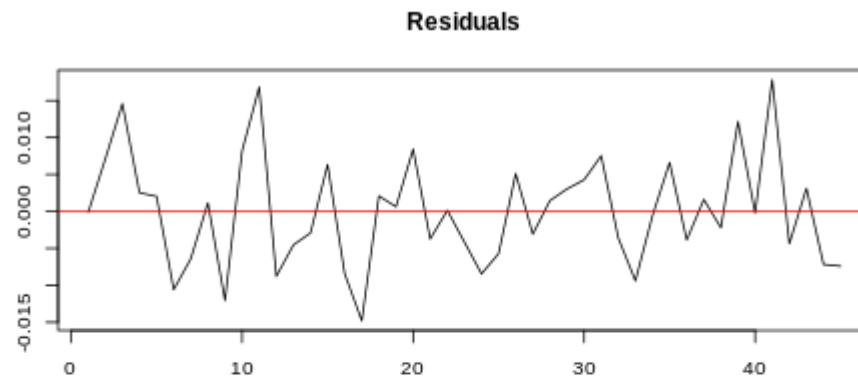
```
Value of test-statistic is: -0.9186 11.5415 3.4784
```

Critical values for test statistics:

	1pct	5pct	10pct
tau3	-4.04	-3.45	-3.15
phi2	6.50	4.88	4.16
phi3	8.73	6.49	5.47

Posteriormente, se valida el modelo, es decir, que se cumpla que los errores son ruido blanco y se distribuyen de manera normal.

```
1 %%R
2 resid=ru_ipvnt@testreg$residuals
3 plot(ru_ipvnt)
```



De manera preliminar, se evidencia que los residuales tienen varianza constante y no están correlacionados.

```
1 %%R
2 auto.arima(resid, max.p=5, max.q=5)
```

```
Series: resid
ARIMA(0,0,0) with zero mean

sigma^2 estimated as 5.47e-05: log likelihood=156.96
AIC=-311.91   AICc=-311.82   BIC=-310.11
```

Con la función auto arima se valida que los residuales no siguen un modelo ARIMA diferente a la caminata aleatoria, con media cero.

```
1 %%R
2 cheq=Arima(resid, c(0,0,0), include.constant=TRUE)
3 summary(cheq)
```

```
Series: resid
ARIMA(0,0,0) with non-zero mean
```

Coefficients:

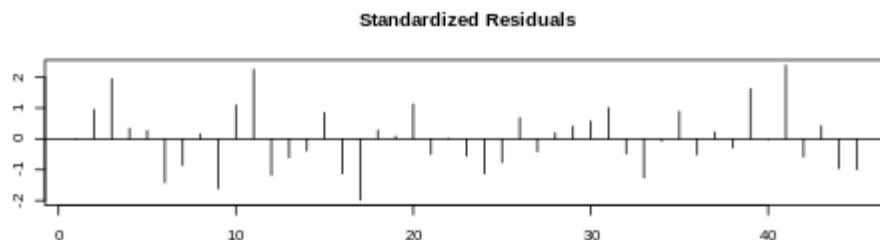
	mean
	0.0000
s.e.	0.0011

```
sigma^2 estimated as 5.594e-05: log likelihood=156.96
AIC=-309.91   AICc=-309.63   BIC=-306.3
```

Training set error measures:

ME	RMSE	MAE	MPE	MAPE	MASE
Training set -4.389513e-20	0.007395716	0.005897446	100	100	0.6892342
ACF1					
Training set 0.02903504					

```
1 %%R
2 tsdiag(cheq, gof=12)
```



A través del criterio de Ljung- Box se evidencia que los residuales no están correlacionados.

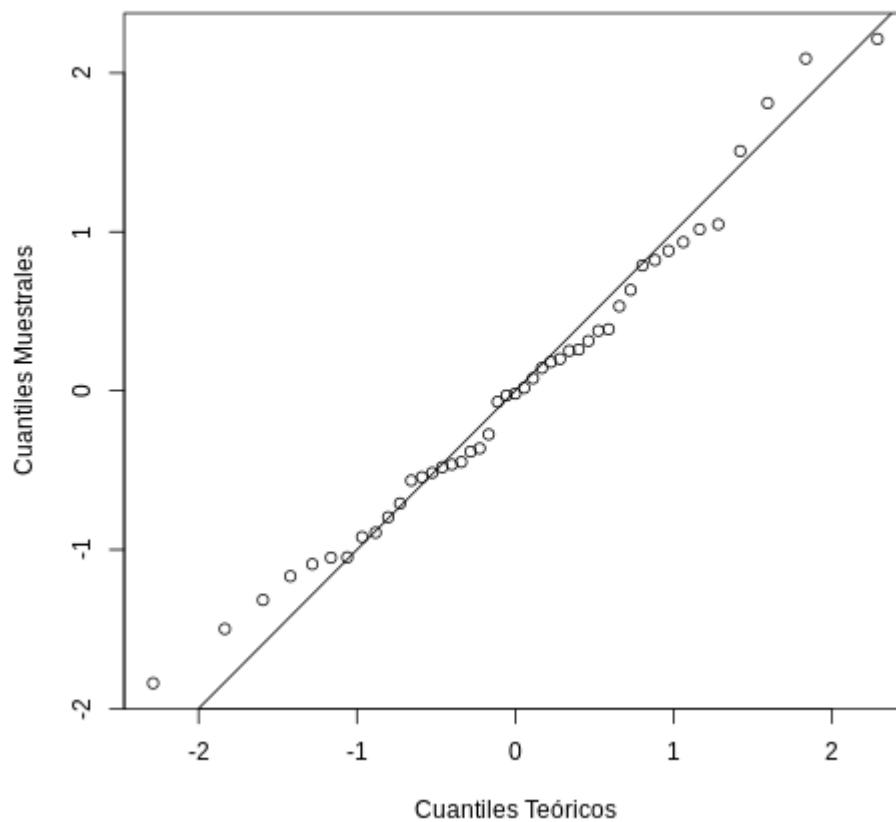
**ACF of Residuals**

```

1 %%R
2 # Verificacion de normalidad en los residuales. Estos se estandarizan
3 resid_sd=resid/ru_ipvnt@testreg$sigma
4 qqnorm(resid_sd, xlab = "Cuantiles Teóricos", ylab = "Cuantiles Muestrales")
5 abline(a=0, b=1)

```

**Normal Q-Q Plot**



Se observa evidencia de normalidad. Ahora se contrasta frente al criterio de Shapiro - Wilk.

```

1 %%R
2 shapiro.test(resid_sd)

```

#### Shapiro-Wilk normality test

```
data: resid_sd
```

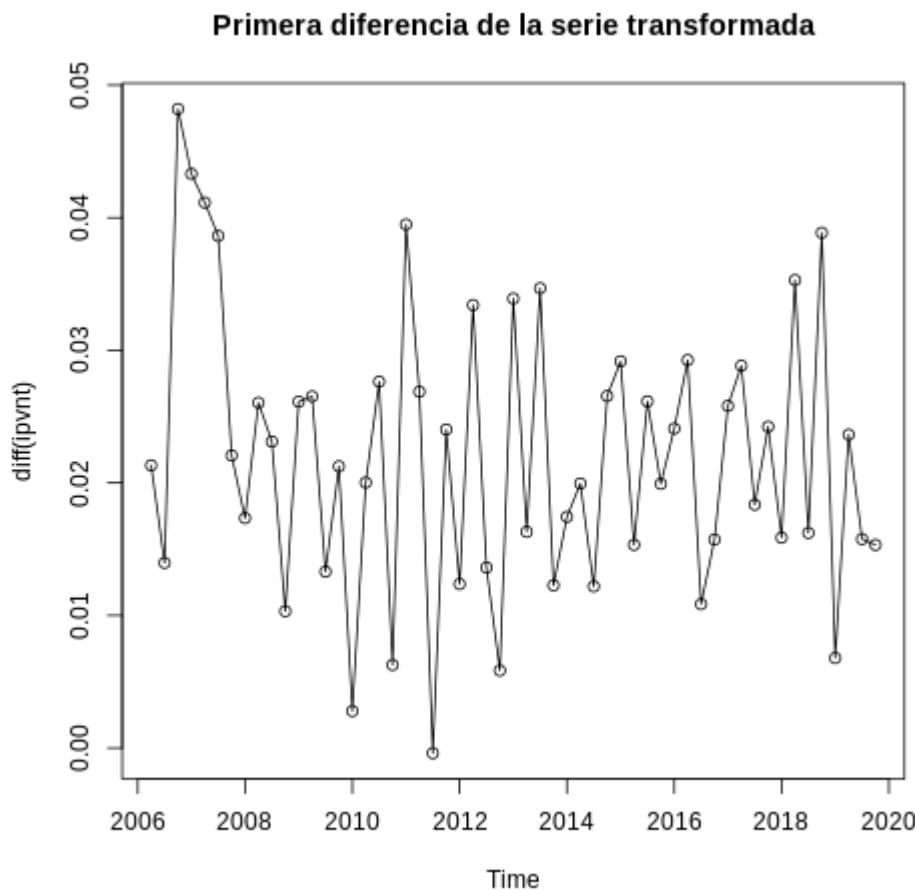
```
W = 0.97954, p-value = 0.6013
```

El modelo elegido, basado en el criterio AIC, para la prueba ADF satisface los supuestos básicos

La serie contiene al menos una raíz unitaria.

Ahora analicemos si existen más raíces unitarias.

```
1 %%R
2 # gráfica de la serie diferenciada una vez
3 plot.ts(diff(ipvnt), type="o", main= "Primera diferencia de la serie transformada")
```



```
1 %%R
2 # prueba de si hay raíz unitarias en la serie diferenciada una vez.
3 ru_dif=ur.df(diff(ipvnt), type = c("drift"), lags=maxlag, selectlags = c("AIC"))
4 summary(ru_dif)
```

```
#####
# Augmented Dickey-Fuller Test Unit Root Test #
#####

Test regression drift
```

Call:

```
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.0166113	-0.0043753	-0.0005938	0.0046521	0.0197248

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	0.046907	0.009572	4.901	1.71e-05 ***
z.lag.1	-2.269283	0.457483	-4.960	1.42e-05 ***
z.diff.lag1	0.755146	0.385242	1.960	0.0571 .
z.diff.lag2	0.504038	0.280954	1.794	0.0806 .
z.diff.lag3	0.307083	0.156647	1.960	0.0571 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.008549 on 39 degrees of freedom

Multiple R-squared: 0.746, Adjusted R-squared: 0.72

F-statistic: 28.64 on 4 and 39 DF, p-value: 3.852e-11

Value of test-statistic is: -4.9604 12.3085

Critical values for test statistics:

	1pct	5pct	10pct
tau2	-3.51	-2.89	-2.58
phi1	6.70	4.71	3.86

```
1 %%R
2 #validación de la ecuación ADF
3 resid1=ru_dif@testreg$residuals           # residuales del modelo ADF
4 plot(ru_dif)
```

### Residuals



Al igual que con la serie transformada, se observa el cumplimiento de los supuestos de ruido blanco y normalidad de los residuales.

```
1 %%R
2 auto.arima(resid1, max.p=5, max.q=5)
```

Series: resid1  
ARIMA(0,1,1)

Coefficients:  
ma1  
-0.9087  
s.e. 0.0648

sigma^2 estimated as 6.878e-05: log likelihood=144.69  
AIC=-285.37 AICc=-285.07 BIC=-281.85

```
0      5      10     15      5      10     15
```

```
1 %%R
2 cheq1=Arima(resid1, c(0,0,0), include.constant=TRUE)
3 summary(cheq1)
```

Series: resid1  
ARIMA(0,0,0) with non-zero mean

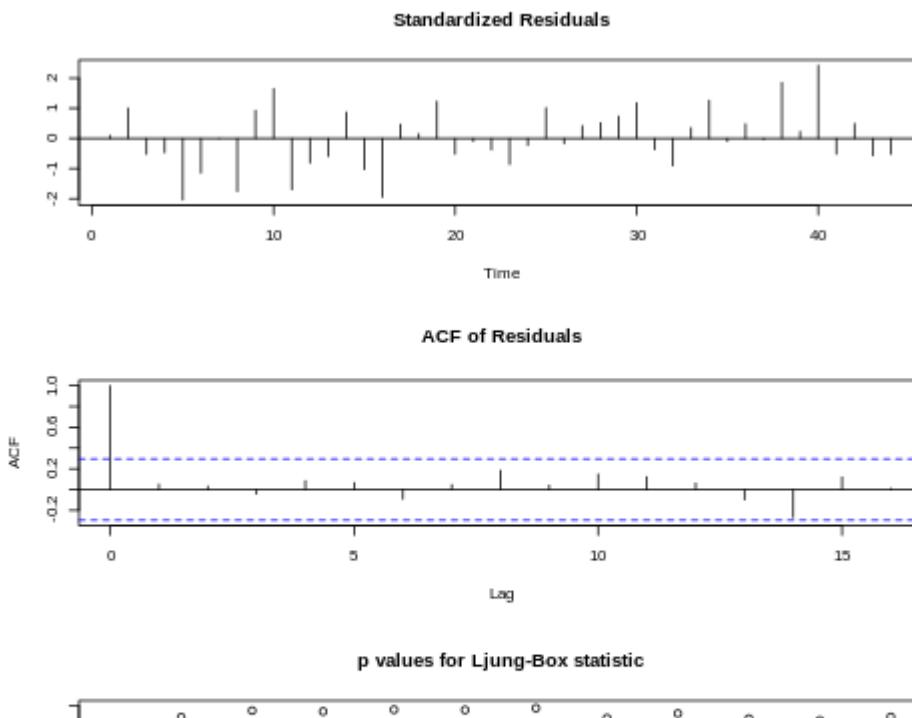
Coefficients:  
mean  
0.0000  
s.e. 0.0012

sigma^2 estimated as 6.629e-05: log likelihood=149.74  
AIC=-295.49 AICc=-295.19 BIC=-291.92

Training set error measures:

ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set 1.078427e-19	0.008049066	0.006420309	100	100	0.6967966	0.0514873

```
1 %%R
2 tsdiag(cheq1, gof=12)
```



```
1 %%R
2 shapiro.test(resid1)
```

#### Shapiro-Wilk normality test

```
data: resid1
W = 0.98766, p-value = 0.9136
```

Los residuales cumplen con las hipótesis. Sin embargo, se rechaza la hipótesis de la existencia de raíz unitaria, el valor del coeficiente es menor a los valores críticos.

La serie tiene derivada=1.

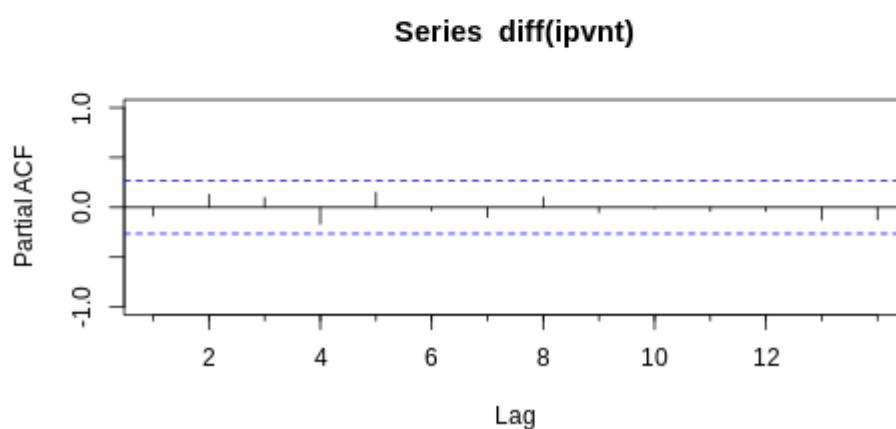
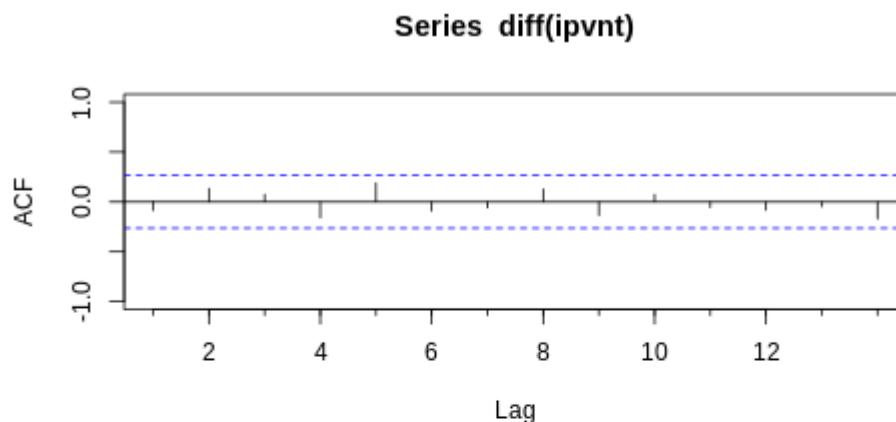
Ahora se determinará el modelo ARIMA (p,1,q).

#### ▼ Determinación del modelo ARIMA

```
1 %%R
2 (max_rezag=round(length(ipvnt)/4))

[1] 14

1 %%R
2 # correlogramas muestrales de la serie transformada
3 par(mfrow=c(2,1))
4 Acf(diff(ipvnt), lag.max=max_rezag, ylim=c(-1,1))
5 Pacf(diff(ipvnt), lag.max=max_rezag, ylim=c(-1,1))
```



Inicialmente, por los correlogramas, podría ser un ARIMA(0,1,0). A continuación, se aplicarán varios métodos para determinar el modelo que mejor se ajustaría.

```
1 %%R
2 # selección "automática" del modelo
3 auto.arima(ipvnt,max.p=10, max.q=10, ic=c("aic"))
```

Series: ipvnt  
ARIMA(0,1,0) with drift

Coefficients:  
drift  
0.0221  
s.e. 0.0014

sigma^2 estimated as 0.0001117: log likelihood=172.77  
AIC=-341.54 AICc=-341.31 BIC=-337.52

La función autorima indica que el modelo de mejor ajuste podría ser un modelo ARIMA (0,1,0).

```
1 %%R
2 # selección del modelo usando criterios de información: AIC o BIC
3 # defina d, p y q
```

```

4 a=1
5 p=5
6 q=5
7 (num_modelos=(p+1)*(q+1))

```

```
[1] 36
```

```

1 %%R
2 aic=matrix(rep(-99, times=num_modelos*3), nrow=num_modelos, ncol=3)
3 bic=matrix(rep(-99, times=num_modelos*3), nrow=num_modelos, ncol=3)
4
5 k=1
6 for(i in 0:p) {
7 for(j in 0:q) {
8 mod=Arima(ipvnt, c(i,d,j), include.drift=T, method=c("CSS-ML"))
9 aic[k, 1]=i
10 aic[k, 2]=j
11 aic[k, 3]=mod$aic
12
13 bic[k, 1]=i
14 bic[k, 2]=j
15 bic[k, 3]=mod$bic
16 k=k+1
17 }
18 }
19
20 aic=data.frame(aic)
21 attach(aic)
22 p=X1
23 q=X2
24 Aic=X3
25 (AIC=data.frame(cbind(p, d, q, Aic)))

```

	p	d	q	Aic
1	0	1	0	-341.5392
2	0	1	1	-339.8613
3	0	1	2	-339.3253
4	0	1	3	-337.3362
5	0	1	4	-335.9875
6	0	1	5	-335.8386
7	1	1	0	-339.9476
8	1	1	1	-338.1475
9	1	1	2	-337.3284
10	1	1	3	-336.9874
11	1	1	4	-335.6403
12	1	1	5	-336.6201
13	2	1	0	-338.8322
14	2	1	1	-336.9897
15	2	1	2	-341.7414
16	2	1	3	-340.5481
17	2	1	4	-339.2856
18	2	1	5	-337.6149
19	3	1	0	-337.3706
20	3	1	1	-337.0059
21	3	1	2	-337.1390
22	3	1	3	-338.6991
23	3	1	4	-338.2551
24	3	1	5	-335.6150
25	4	1	0	-337.3146
26	4	1	1	-336.1570

```

27 4 1 2 -335.5728
28 4 1 3 -334.9027
29 4 1 4 -336.7752
30 4 1 5 -333.7730
31 5 1 0 -336.6183
32 5 1 1 -334.6505
33 5 1 2 -337.5085
34 5 1 3 -335.7548
35 5 1 4 -335.1692
36 5 1 5 -333.1527

```

```

1 %%R
2 bic=data.frame(bic)
3 attach(bic)

```

R[write to console]: The following objects are masked from aic:

X1, X2, X3

```

1 %%R
2 p=X1
3 q=X2
4 Bic=X3
5 (BIC=data.frame(cbind(p, d, q, Bic)))

```

	p	d	q	Bic
1	0	1	0	-337.5246
2	0	1	1	-333.8393
3	0	1	2	-331.2959
4	0	1	3	-327.2995
5	0	1	4	-323.9435
6	0	1	5	-321.7873
7	1	1	0	-333.9256
8	1	1	1	-330.1182
9	1	1	2	-327.2917
10	1	1	3	-324.9434
11	1	1	4	-321.5890
12	1	1	5	-320.5615
13	2	1	0	-330.8029
14	2	1	1	-326.9530
15	2	1	2	-329.6974
16	2	1	3	-326.4968
17	2	1	4	-323.2270
18	2	1	5	-319.5489
19	3	1	0	-327.3339
20	3	1	1	-324.9619
21	3	1	2	-323.0877
22	3	1	3	-322.6404
23	3	1	4	-320.1891
24	3	1	5	-315.5416
25	4	1	0	-325.2706
26	4	1	1	-322.1056
27	4	1	2	-319.5141
28	4	1	3	-316.8367

```

29 4 1 4 -316.7019
30 4 1 5 -311.6924
31 5 1 0 -322.5670
32 5 1 1 -318.5918
33 5 1 2 -319.4425
34 5 1 3 -315.6815
35 5 1 4 -313.0886
36 5 1 5 -309.0647

```

```

1 %%R
2 cbind(AIC[order(Aic),], "      ", BIC[order(Bic),])

```

	p	d	q	Aic	"	p	d	q	Bic
15	2	1	2	-341.7414		0	1	0	-337.5246
1	0	1	0	-341.5392		1	1	0	-333.9256
16	2	1	3	-340.5481		0	1	1	-333.8393
7	1	1	0	-339.9476		0	1	2	-331.2959
2	0	1	1	-339.8613		2	1	0	-330.8029
3	0	1	2	-339.3253		1	1	1	-330.1182
17	2	1	4	-339.2856		2	1	2	-329.6974
13	2	1	0	-338.8322		3	1	0	-327.3339
22	3	1	3	-338.6991		0	1	3	-327.2995
23	3	1	4	-338.2551		1	1	2	-327.2917
8	1	1	1	-338.1475		2	1	1	-326.9530
18	2	1	5	-337.6149		2	1	3	-326.4968
33	5	1	2	-337.5085		4	1	0	-325.2706
19	3	1	0	-337.3706		3	1	1	-324.9619
4	0	1	3	-337.3362		1	1	3	-324.9434
9	1	1	2	-337.3284		0	1	4	-323.9435
25	4	1	0	-337.3146		2	1	4	-323.2270
21	3	1	2	-337.1390		3	1	2	-323.0877
20	3	1	1	-337.0059		3	1	3	-322.6404
14	2	1	1	-336.9897		5	1	0	-322.5670
10	1	1	3	-336.9874		4	1	1	-322.1056
29	4	1	4	-336.7752		0	1	5	-321.7873
12	1	1	5	-336.6201		1	1	4	-321.5890
31	5	1	0	-336.6183		1	1	5	-320.5615
26	4	1	1	-336.1570		3	1	4	-320.1891
5	0	1	4	-335.9875		2	1	5	-319.5489
6	0	1	5	-335.8386		4	1	2	-319.5141
34	5	1	3	-335.7548		5	1	2	-319.4425
11	1	1	4	-335.6403		5	1	1	-318.5918
24	3	1	5	-335.6150		4	1	3	-316.8367
27	4	1	2	-335.5728		4	1	4	-316.7019
35	5	1	4	-335.1692		5	1	3	-315.6815
28	4	1	3	-334.9027		3	1	5	-315.5416
32	5	1	1	-334.6505		5	1	4	-313.0886
30	4	1	5	-333.7730		4	1	5	-311.6924
36	5	1	5	-333.1527		5	1	5	-309.0647

Bajo los criterios de AIC y BIC el mejor modelo sería una ARIMA (0,1,0).

```

1 %%R
2 eacf(diff(ipvnt))

```

## AR/MA

```

0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 o o o o o o o o o o o o o
1 x o o o o o o o o o o o o o
2 x x o o o o o o o o o o o o o
3 x o o o o o o o o o o o o o
4 x o x o o o o o o o o o o o o
5 o o o o o o o o o o o o o o o
6 x o o o o o o o o o o o o o o
7 x o x o o o o o o o o o o o o

```

De acuerdo con la EACF, el posible modelo sería ARIMA (0,1,0).

En ese sentido, con todos los métodos, se concluye que el modelo sería un ARIMA (0,1,0).

## ▼ Estimación

```

1 %%R
2 mod1_CSS_ML=Arima(ipvn, c(0,1,0), include.drift=TRUE,lambda=0, method = c("CSS-ML"))
3 summary(mod1_CSS_ML)

Series: ipvn
ARIMA(0,1,0) with drift
Box Cox transformation: lambda= 0

Coefficients:
      drift
      0.0221
  s.e.  0.0014

sigma^2 estimated as 0.0001117:  log likelihood=172.77
AIC=-341.54  AICc=-341.31  BIC=-337.52

Training set error measures:
          ME        RMSE       MAE       MPE       MAPE       MASE
Training set -0.04115317  0.9076744  0.7343985  0.00141472  0.8327413  0.09256302
          ACF1
Training set -0.3230722


```

```

1 %%R
2 (res1_CSS_ML=residuals(mod1_CSS_ML))

```

	Qtr1	Qtr2	Qtr3	Qtr4
2006	3.805274e-03	-7.392651e-04	-8.094672e-03	2.613943e-02
2007	2.124170e-02	1.907622e-02	1.656579e-02	3.383397e-06
2008	-4.719219e-03	3.980753e-03	1.053691e-03	-1.174252e-02
2009	4.059214e-03	4.460012e-03	-8.758913e-03	-8.159908e-04
2010	-1.927121e-02	-2.033407e-03	5.571741e-03	-1.579058e-02
2011	1.742757e-02	4.811809e-03	-2.245019e-02	1.963053e-03
2012	-9.697230e-03	1.135227e-02	-8.451268e-03	-1.621606e-02
2013	1.184532e-02	-5.755179e-03	1.261765e-02	-9.791437e-03
2014	-4.623804e-03	-2.112030e-03	-9.868705e-03	4.489138e-03

```
2015  7.110180e-03 -6.735335e-03  4.089402e-03 -2.133339e-03
2016  2.035179e-03  7.199896e-03 -1.120831e-02 -6.337271e-03
2017  3.744464e-03  6.774715e-03 -3.706704e-03  2.179189e-03
2018 -6.195885e-03  1.323105e-02 -5.836025e-03  1.680007e-02
2019 -1.527214e-02  1.575179e-03 -6.301413e-03 -6.739972e-03
```

```
1 %%R
2 res1_est=res1_CSS_ML/(mod1_CSS_ML$sigma2^.5) # estandarización de los residuales
```

## ▼ Diagnósticos

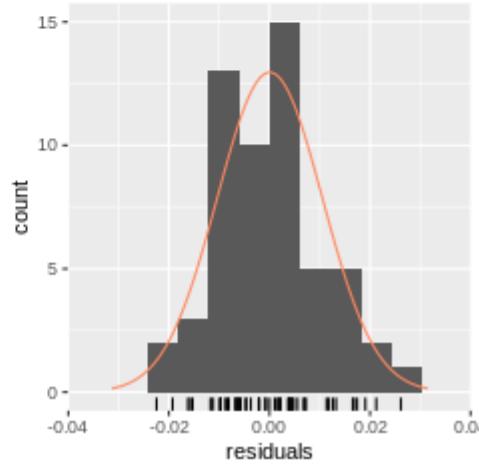
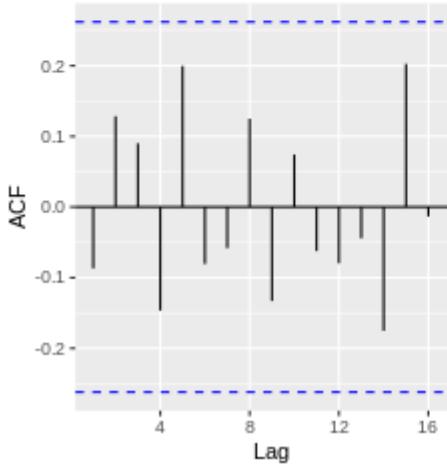
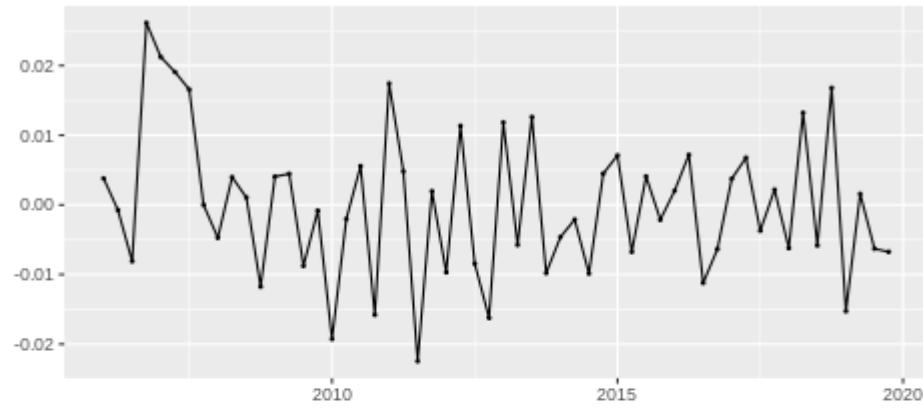
```
1 %%R
2 checkresiduals(mod1_CSS_ML, lag=15) # de la librería forecast
```

### Ljung-Box test

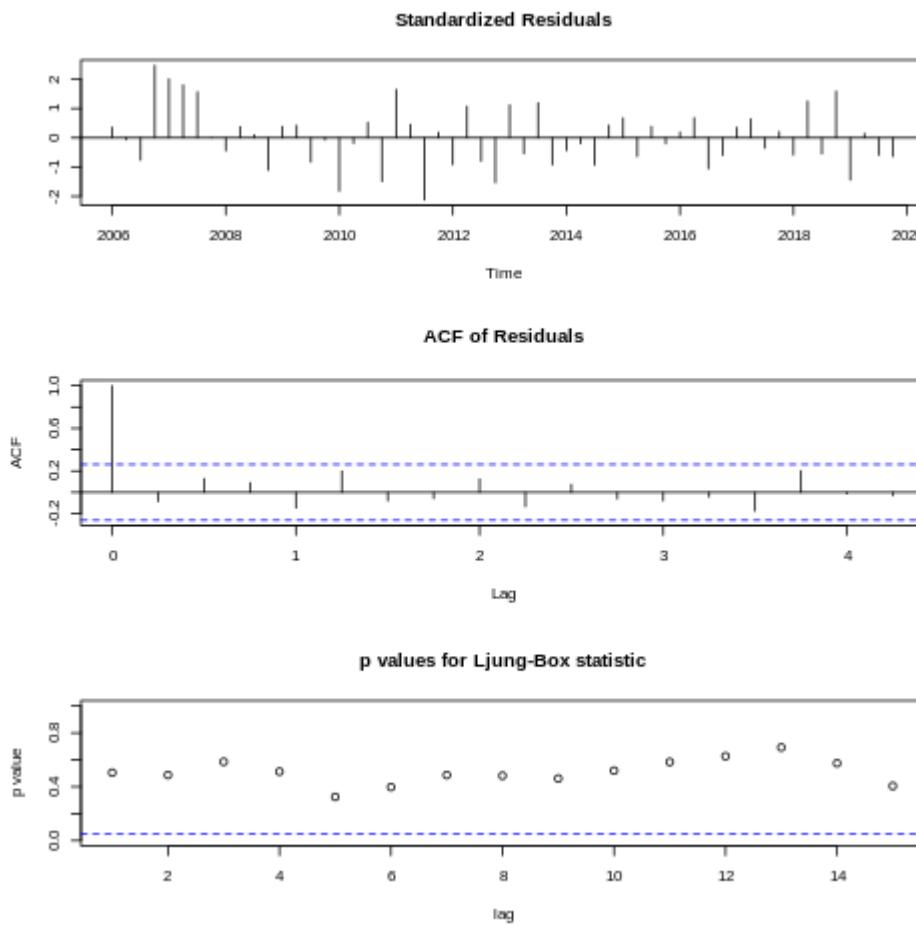
```
data: Residuals from ARIMA(0,1,0) with drift
Q* = 15.65, df = 14, p-value = 0.3352
```

Model df: 1. Total lags used: 15

### Residuals from ARIMA(0,1,0) with drift



```
1 %%R
2 tsdiag(mod1_CSS_ML, gof=15)
```



Este modelo cumple con el supuesto de resiudales ruido blanco.

```

1 %%R
2 # chequeo de normalidad
3 # gráfico cuantil-cuantil
4 qqnorm(res1_est, xlab = "Cuantiles Teóricos", ylab = "Cuantiles Muestrales")
5 abline(a=0, b=1)

```

### Normal Q-Q Plot

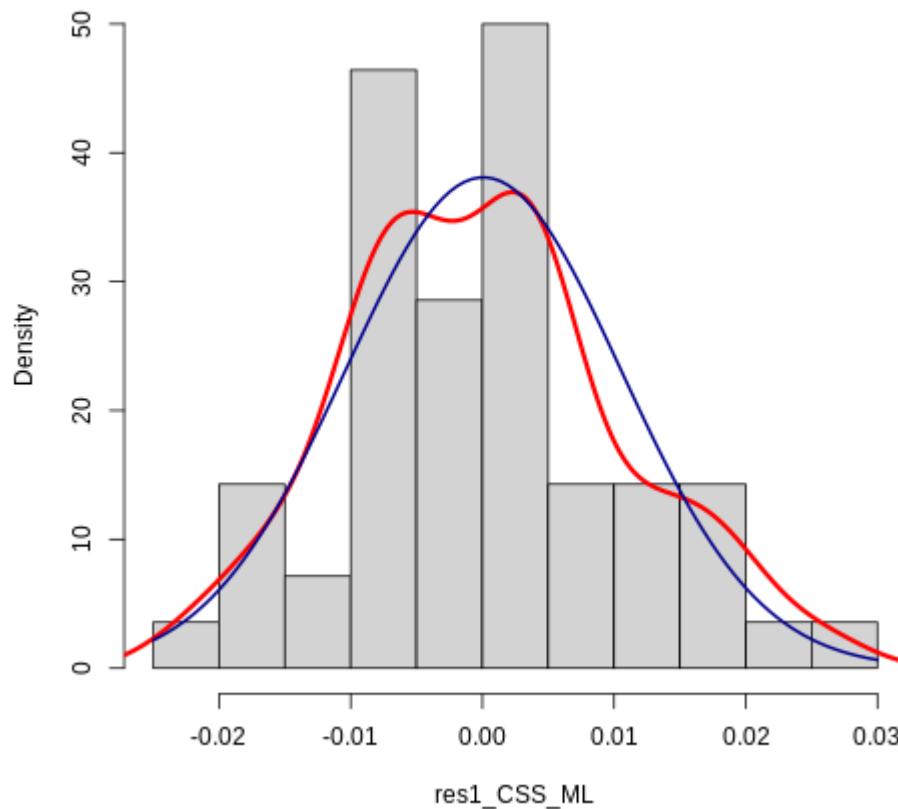


```

1 %%R
2 # histograma, densidad kernel y gráfico normal
3 hist(res1_CSS_ML, prob=T)
4 lines(density(res1_CSS_ML), lwd=3, col="red")
5 curve(dnorm(x, mean=mean(res1_CSS_ML), sd=sd(res1_CSS_ML)), col="darkblue", lwd=2, add=TRUE)

```

### Histogram of res1\_CSS\_ML



```

1 %%R
2 shapiro.test(res1_est)

```

### Shapiro-Wilk normality test

```

data: res1_est
W = 0.986, p-value = 0.7594

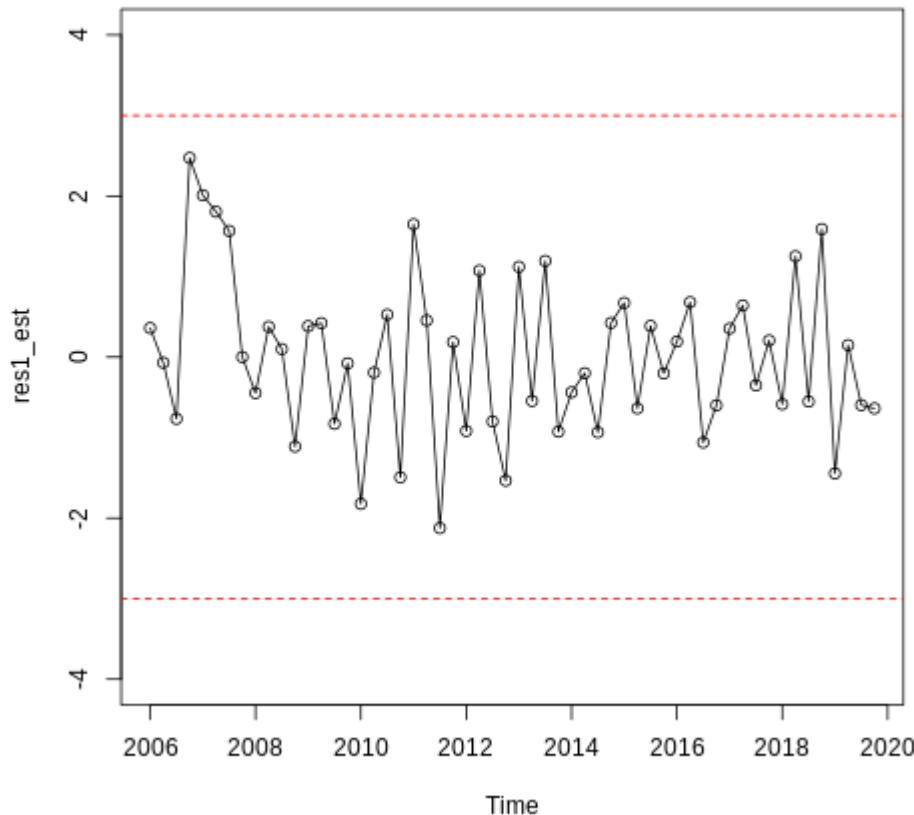
```

Asimismo, en el modelo 1 los residuales cumplen con la hipótesis de normalidad.

```

1 %%R
2 # detección de observaciones atípicas distantes
3 plot.ts(res1_est, type="o", ylim=c(-4,4))
4 abline(a=-3, b=0, col="red", lty=2)
5 abline(a=3, b=0, col="red", lty=2)

```



No se observa la presencia de datos atípicos.

#### ▼ Significancia de los coeficientes

```

1 %%R
2 # Evaluación de la significancia estadística de los coeficientes estimados
3 # Construcción de los estadísticos de prueba, t
4 # estadísticos t
5 summary(mod1_CSS_ML)

```

Series: ipvn  
ARIMA(0,1,0) with drift  
Box Cox transformation: lambda= 0

Coefficients:  
drift  
0.0221

s.e. 0.0014

sigma^2 estimated as 0.0001117: log likelihood=172.77  
AIC=-341.54 AICc=-341.31 BIC=-337.52

Training set error measures:

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.04115317	0.9076744	0.7343985	0.00141472	0.8327413	0.09256302
	ACF1					
Training set	-0.3230722					

```
1 %%R
2 (t=coef(mod1_CSS_ML)/(diag(vcov(mod1_CSS_ML))))^.5
```

```
drift
15.56991
```

```
1 %%R
2 # valores P
3 (val_p=2*pnorm(t, lower.tail=FALSE))
```

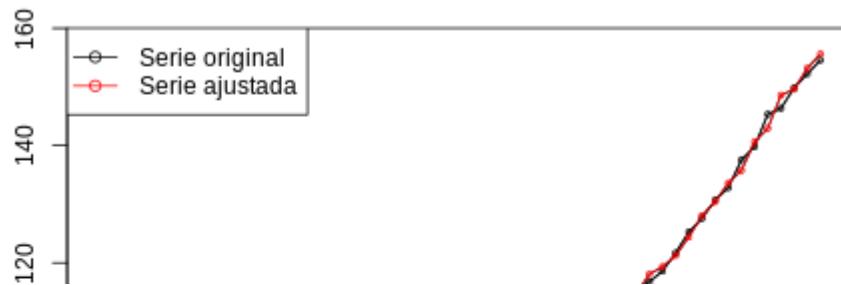
```
drift
1.165643e-54
```

```
1 %%R
2 qnorm(.955)
```

[1] 1.695398

```
1 %%R
2 # valores ajustados del modelo
3 ajust1=(mod1_CSS_ML$fitted)
4 # gráfico para los valores ajustados y los valores observados
5 ts.plot(ipvn,ajust1, main= "Serie original vs Serie ajustada") # gráfico de las series contra el tiempo
6 lines(ipvn, col="black", type="o", cex=.5)
7 lines(ajust1, col="red", type="o", cex=.5)
8 legend("topleft", legend=c("Serie original","Serie ajustada"), col=c("black","red"),lwd=c(1,1),pch=c(1,1))
```

### Serie original vs Serie ajustada

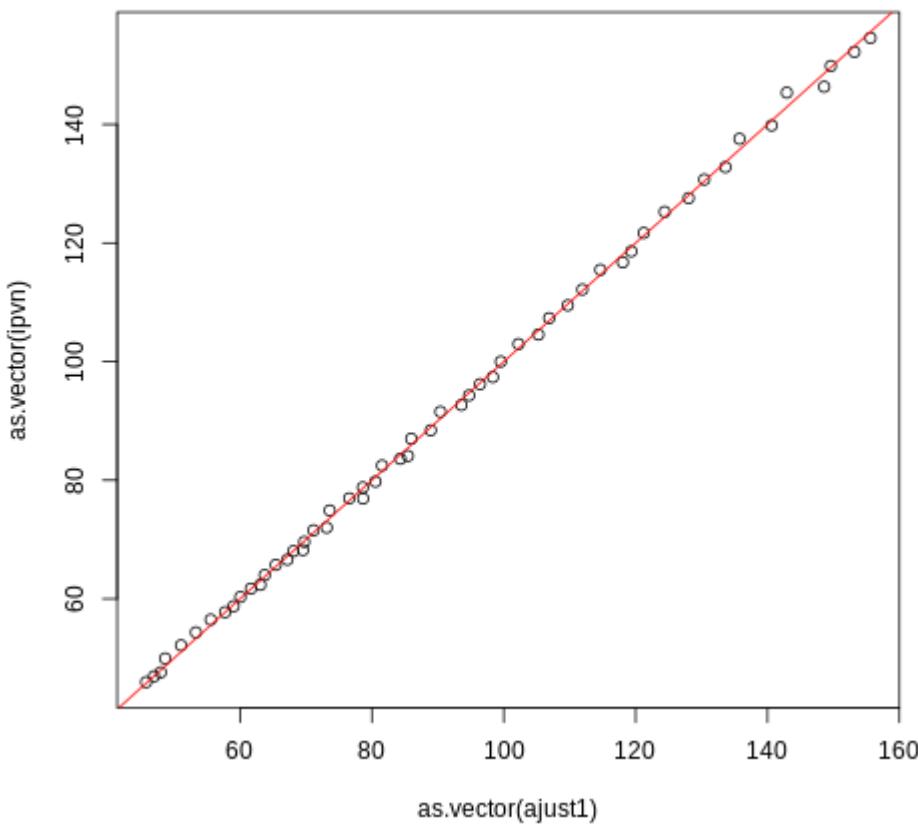


```

1 %%R
2 # gráfico de dispersión de valores observados vs valores ajustados
3 plot(as.vector(ajust1), as.vector(ipvn), type="p", main="Dispersión de la serie observada")
4 abline(0,1, col="red")           # contra la serie ajustada

```

### Dispersión de la serie observada



En términos generales, a nivel gráfico, se observa un importante ajuste por parte del modelo.

#### ▼ Evaluación del pronóstico

Se segmentó la serie en 48 datos para el entrenamiento y 8 para el testeo. Conservando la secuencialidad de los mismos

```

1 %%R
2 mod_Evalpron1=Arima(ipvn[1:48], c(0, 1, 0), include.drift=TRUE, lambda=0, method = c("CSS-ML"))
3 summary(mod_Evalpron1)

Series: ipvn[1:48]
ARIMA(0,1,0) with drift
Box Cox transformation: lambda= 0

Coefficients:
      drift
      0.0222
  s.e.  0.0015

sigma^2 estimated as 0.0001126:  log likelihood=147.53
AIC=-291.07  AICc=-290.8  BIC=-287.37

Training set error measures:
          ME        RMSE       MAE       MPE       MAPE       MASE
Training set -0.0332919 0.7717581 0.641068 0.002536006 0.8222242 0.3552251
          ACF1
Training set -0.1544876

```

Nuevamente, se valida el cumplimiento de los supuestos de los residuales (normalidad y ruido blanco).

```

1 %%R
2 res_eval1=residuals(mod_Evalpron1)
3 tsdiag(mod_Evalpron1, gof=14)

```

Standardized Residuals

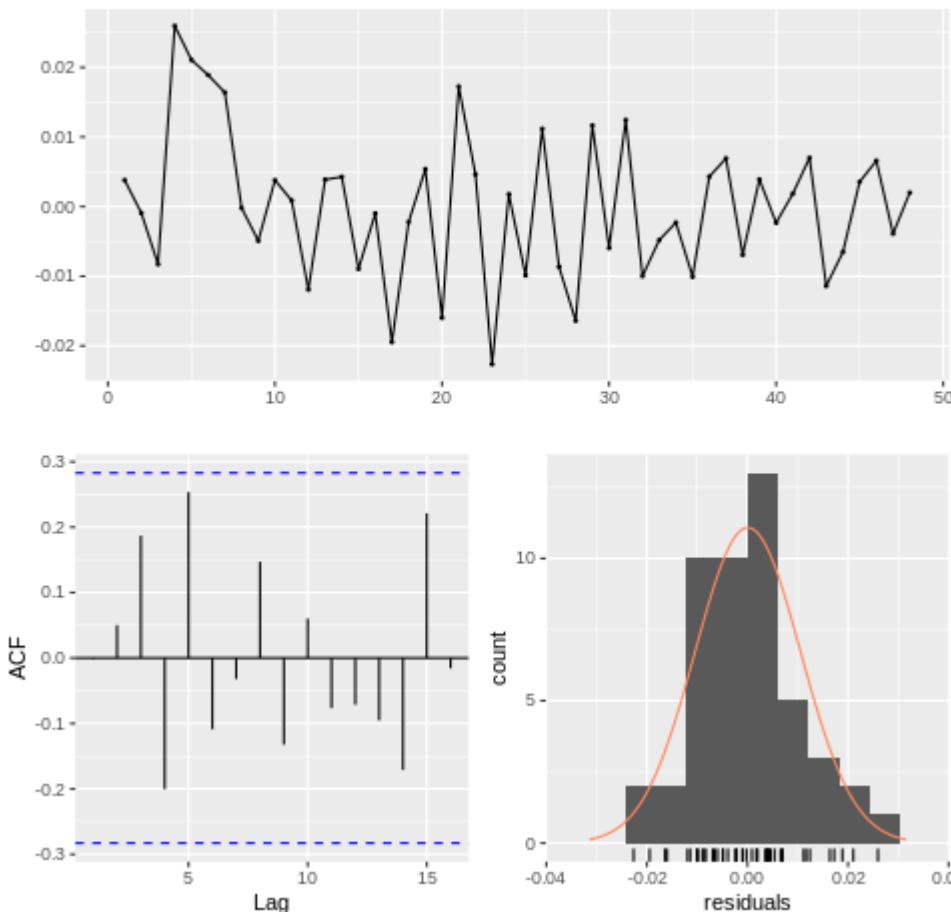
Influence

## Ljung-Box test

data: Residuals from ARIMA(0,1,0) with drift  
Q\* = 11.124, df = 9, p-value = 0.2673

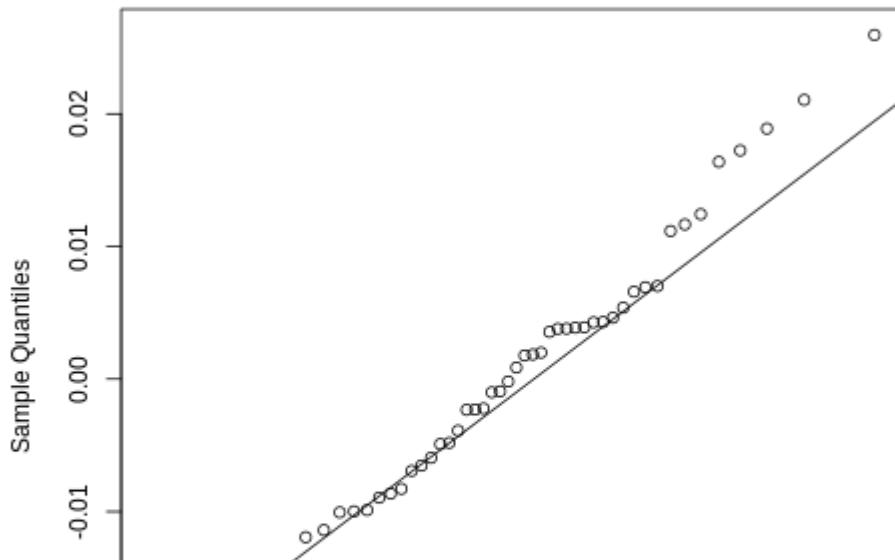
Model df: 1. Total lags used: 10

Residuals from ARIMA(0,1,0) with drift



```
1 %%R  
2 qqnorm(residuals(mod_Evalpron1))  
3 qqline(residuals(mod_Evalpron1))
```

### Normal Q-Q Plot



```

1 %%%R
2 shapiro.test(residuals(mod_Evalpron1))

```

#### Shapiro-Wilk normality test

```

data: residuals(mod_Evalpron1)
W = 0.98688, p-value = 0.8633

```

```

1 %%%R
2 (z_pred1=forecast(mod_Evalpron1, h=8, level=c(80, 95), fan=FALSE))

```

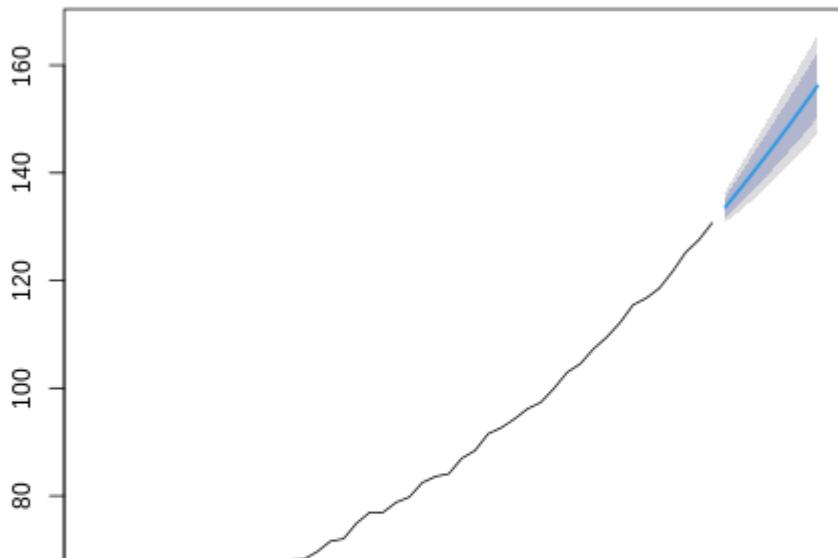
Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
49	133.6402	131.8351	135.4700	130.8894	136.4487
50	136.6464	134.0435	139.2999	132.6858	140.7253
51	139.7204	136.4678	143.0505	134.7767	144.8453
52	142.8634	139.0301	146.8024	137.0427	148.9314
53	146.0772	141.7020	150.5875	139.4393	153.0311
54	149.3633	144.4698	154.4225	141.9446	157.1697
55	152.7233	147.3259	158.3183	144.5464	161.3627
56	156.1588	150.2663	162.2824	147.2376	165.6206

```

1 %%%R
2 plot(z_pred1)

```

### Forecasts from ARIMA(0,1,0) with drift



```

1 %%R
2 # Evaluación de los pronósticos
3 # lista de los valores reales y los pronósticos
4 cbind(ipvn[49:56], z_pred1$mean)

```

Time Series:  
 Start = 49  
 End = 56  
 Frequency = 1  
 ipvn[49:56] z\_pred1\$mean

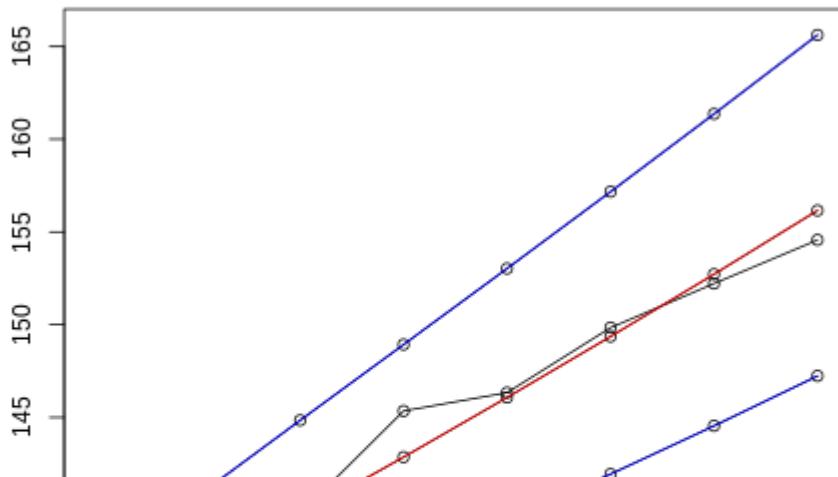
	ipvn[49:56]	z_pred1\$mean
49	132.79	133.6402
50	137.56	136.6464
51	139.81	139.7204
52	145.35	142.8634
53	146.34	146.0772
54	149.84	149.3633
55	152.22	152.7233
56	154.57	156.1588

```

1 %%R
2 ts.plot(ipvn[49:56], z_pred1$mean, z_pred1$lower[,2], z_pred1$upper[,2], type="o", main="Pronóstico e intervalos de confianza")
3 lines(z_pred1$mean, col="red")
4 lines(z_pred1$lower[,2], col="blue")
5 lines(z_pred1$upper[,2], col="blue")

```

## Pronóstico e intervalos de confianza



### Precisión

```
5 | σ |
```

```
1 %%R
2 (rcm=(mean((ipvn[49:56]-ts(z_pred1$mean))^2))^.5)
```

```
[1] 1.1631
```

```
1 %%R
2 (rcmp=100*(mean(((ipvn[49:56]-ts(z_pred1$mean))/ipvn[49:56])^2))^.5)
```

```
[1] 0.7969527
```

```
1 %%R
2 (eam=mean(abs(ipvn[49:56]-ts(z_pred1$mean)))) # error absoluto medio
```

```
[1] 0.8964418
```

```
1 %%R
2 (MAPE_ST=100*mean(abs((ipvn[49:56]-ts(z_pred1$mean))/ipvn[49:56]))) #MAPE
```

```
[1] 0.6169318
```

```
1 MAPE_ST=%R MAPE_ST
2 MAPE_ST=MAPE_ST[0]
```

```
1 tabla.loc[3] = ["Serie de tiempo univariante", MAPE_ST.round(3)]
```

```
1 %%R
2 (ecm=mean((ipvn[49:56]-ts(z_pred1$mean))^2)) #error cuadrático medio
```

```
[1] 1.352802
```

## ▼ Pronósticos finales

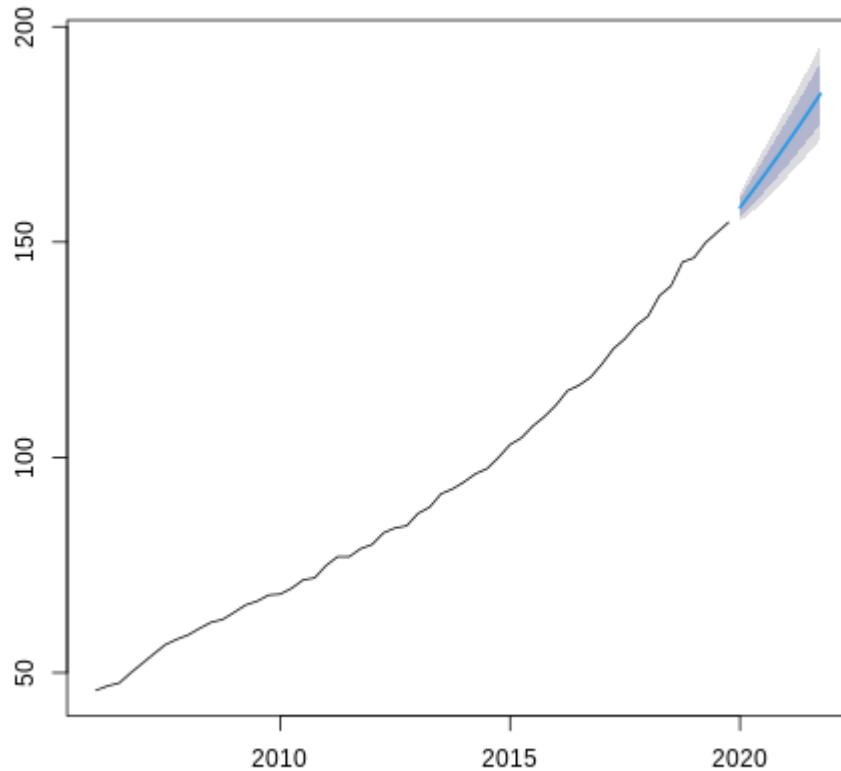
Finalmente, se hacen los pronósticos para los siguientes 8 trimestres, considerando todos los valores de la serie.

```
1 %%R
2 par(mfrow=c(1,2))
3 (z_pron1<-forecast(mod1_CSS_ML, h=8, level=c(80,95), fan=F))
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
2020 Q1	158.0177	155.8919	160.1726	154.7781	161.3252
2020 Q2	161.5424	158.4774	164.6666	156.8786	166.3448
2020 Q3	165.1456	161.3164	169.0658	159.3254	171.1785
2020 Q4	168.8293	164.3171	173.4653	161.9776	175.9707
2021 Q1	172.5950	167.4460	177.9024	164.7828	180.7777
2021 Q2	176.4448	170.6868	182.3971	167.7152	185.6288
2021 Q3	180.3805	174.0308	186.9618	170.7605	190.5424
2021 Q4	184.4039	177.4730	191.6056	173.9100	195.5311

```
1 %%R
2 plot(z_pron1)
```

**Forecasts from ARIMA(0,1,0) with drift**



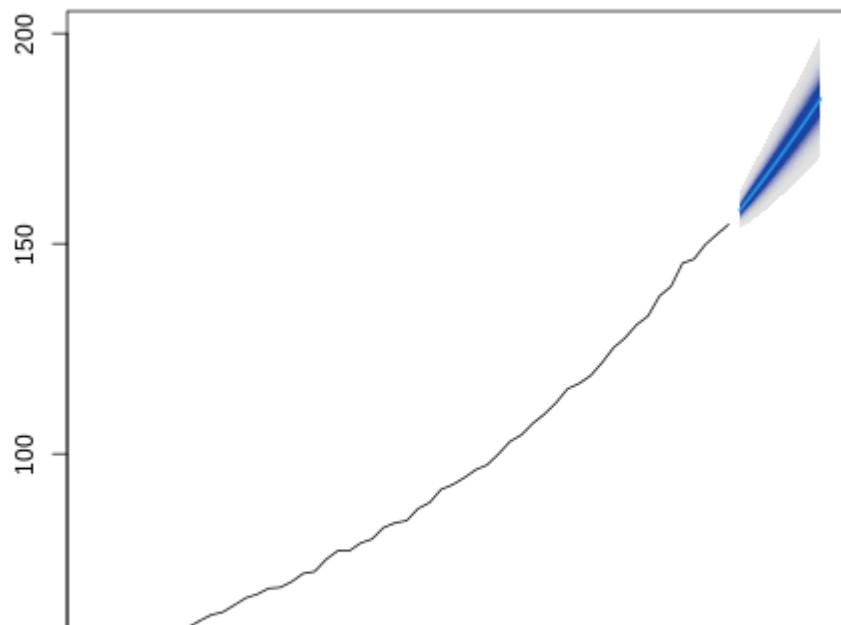
```
1 %%R
2 (z_pron1<-forecast(mod1_CSS_ML, h=8, level=c(80,95), fan=T))
```

	Point	Forecast	Lo 51	Hi 51	Lo 54	Hi 54	Lo 57	Hi 57
2020 Q1		158.0177	156.8690	159.1748	156.7886	159.2565	156.7052	159.3413
2020 Q2		161.5424	159.8842	163.2178	159.7682	163.3362	159.6480	163.4592
2020 Q3		165.1456	163.0718	167.2458	162.9270	167.3945	162.7769	167.5488
2020 Q4		168.8293	166.3836	171.3108	166.2130	171.4867	166.0362	171.6693
2021 Q1		172.5950	169.8022	175.4339	169.6075	175.6352	169.4058	175.8443
2021 Q2		176.4448	173.3196	179.6265	173.1019	179.8523	172.8764	180.0869
2021 Q3		180.3805	176.9320	183.8962	176.6920	184.1460	176.4434	184.4054
2021 Q4		184.4039	180.6376	188.2488	180.3757	188.5222	180.1044	188.8061
	Lo 60	Hi 60	Lo 63	Hi 63	Lo 66	Hi 66	Lo 69	Hi 69
2020 Q1	156.6184	159.4296	156.5276	159.5220	156.4322	159.6193	156.3313	159.7224
2020 Q2	159.5230	163.5873	159.3922	163.7215	159.2548	163.8628	159.1096	164.0124
2020 Q3	162.6208	167.7097	162.4576	167.8782	162.2861	168.0556	162.1048	168.2435
2020 Q4	165.8523	171.8596	165.6601	172.0590	165.4582	172.2690	165.2448	172.4914
2021 Q1	169.1960	176.0623	168.9768	176.2907	168.7466	176.5312	168.5033	176.7862
2021 Q2	172.6419	180.3315	172.3970	180.5877	172.1397	180.8577	171.8678	181.1438
2021 Q3	176.1849	184.6760	175.9149	184.9594	175.6313	185.2581	175.3317	185.5746
2021 Q4	179.8224	189.1022	179.5277	189.4126	179.2184	189.7395	178.8916	190.0862
	Lo 72	Hi 72	Lo 75	Hi 75	Lo 78	Hi 78	Lo 81	Hi 81
2020 Q1	156.2238	159.8323	156.1082	159.9507	155.9825	160.0795	155.8440	160.2218
2020 Q2	158.9548	164.1720	158.7885	164.3440	158.6078	164.5312	158.4087	164.7380
2020 Q3	161.9117	168.4441	161.7043	168.6602	161.4789	168.8956	161.2307	169.1556
2020 Q4	165.0176	172.7290	164.7735	172.9848	164.5084	173.2636	164.2164	173.5717
2021 Q1	168.2443	177.0583	167.9660	177.3516	167.6639	177.6712	167.3312	178.0245
2021 Q2	171.5784	181.4493	171.2676	181.7786	170.9302	182.1374	170.5586	182.5342
2021 Q3	175.0129	185.9127	174.6705	186.2772	174.2988	186.6744	173.8896	187.1136
2021 Q4	178.5438	190.4564	178.1704	190.8555	177.7651	191.2907	177.3190	191.7719
	Lo 84	Hi 84	Lo 87	Hi 87	Lo 90	Hi 90	Lo 93	Hi 93
2020 Q1	155.6885	160.3818	155.5092	160.5668	155.2944	160.7888	155.0205	161.0730
2020 Q2	158.1851	164.9709	157.9276	165.2399	157.6192	165.5632	157.2261	165.9771
2020 Q3	160.9520	169.4485	160.6311	169.7870	160.2471	170.1939	159.7578	170.7152
2020 Q4	163.8887	173.9188	163.5114	174.3201	163.0601	174.8025	162.4853	175.4209
2021 Q1	166.9579	178.4225	166.5283	178.8828	166.0145	179.4364	165.3604	180.1463
2021 Q2	170.1419	182.9813	169.6623	183.4985	169.0890	184.1207	168.3593	184.9187
2021 Q3	173.4308	187.6087	172.9028	188.1816	172.2718	188.8709	171.4689	189.7552
2021 Q4	176.8189	192.3144	176.2435	192.9422	175.5560	193.6978	174.6814	194.6676
	Lo 96	Hi 96	Lo 99	Hi 99				
2020 Q1	154.6247	161.4852	153.7739	162.3787				
2020 Q2	156.6588	166.5781	155.4411	167.8831				
2020 Q3	159.0521	171.4726	157.5392	173.1193				
2020 Q4	161.6568	176.3199	159.8826	178.2765				
2021 Q1	164.4180	181.1788	162.4018	183.4281				
2021 Q2	167.3085	186.0801	165.0624	188.6122				
2021 Q3	170.3133	191.0428	167.8449	193.8523				
2021 Q4	173.4231	196.0800	170.7375	199.1642				

1 %%R

2 plot(forecast(z\_pron1))

### Forecasts from ARIMA(0,1,0) with drift



#### ▼ 4.2.2 Red Neuronal Univariada

#### ▼ Ajuste

Tomando la variable respuesta del conjunto de datos.

```
1 df_indices.head()
```

	Periodo	Año	Trim	d_IPVN_MDE_Num	d_TD_MDE_Porc	d_TO_MDE_Porc	d_PT_MDE_Num_Mill
0	2005-01	2005	I	41.72	15.050086	50.983674	3107.2550
1	2005-02	2005	II	42.85	14.461586	50.981004	3119.6576
2	2005-03	2005	III	44.81	14.811179	50.667551	3131.7430
3	2005-04	2005	IV	45.80	10.760881	53.078909	3143.5746
4	2006-01	2006	I	45.94	14.317852	51.264581	3155.2636

```
1 # Sacando solo la variable respuesta y el periodo del dataset
2 fecha = df_indices.iloc[4:,0]
3 ipvn = df_indices.iloc[4:,3]
```

```
1 #Un análisis descriptivo rápido de la serie.
2 ipvn.describe()
```

```

count      57.000000
mean       92.870526
std        32.213213
min        45.940000
25%        66.600000
50%        86.990000
75%        116.740000
max        157.770000
Name: d_IPVN_MDE_Num, dtype: float64

```

```
1 ipvn=pd.DataFrame(ipvn)
```

Se tiene un total de 57 registros trimestrales, y la serie no es estacionaria, presentando una tendencia determinística de alta dependencia con los trimestres

```

1 #Se cargan las librerías para correr la red neuronal
2 import numpy
3 import math
4 from keras.models import Sequential
5 from keras.layers import Dense
6 from keras.layers import LSTM
7 from sklearn.preprocessing import MinMaxScaler
8 from sklearn.metrics import mean_squared_error

```

La red que se utilizará es LSTM (redes de memoria larga a corto plazo) a través de la biblioteca Keras, que es un red neuronal recurrente.

Estas redes se usan para analizar datos de series temporales considerando la dimensión de tiempo, basadas en el uso de información secuencial. Asimismo, se basan en bucles que llevan a que la salida de la red o de una parte de ella en un momento dado sirva como entrada de la propia red en el siguiente momento.

Una categoría de estas redes, se denomina redes de memoria larga-corto plazo por sus siglas LSTM y es una técnica de aprendizaje profundo. Fueron desarrolladas por Hochreiter and Schmidhuber (1997) y tienen la capacidad de aprender dependencias a largo plazo. Se desarrollaron para hacer frente al problema del gradiente de desaparición que se puede encontrar al entrenar a los RNN tradicionales.

## Referencia

### ▼ Preparación de la estructura de datos

```

1 # Con esta función, se convierte un array de valores en una matriz de conjuntos de datos
2 def create_dataset(dataset, look_back=1):
3     dataX, dataY = [], []
4     for i in range(len(dataset)-look_back-1):
5         a = dataset[i:(i+look_back), 0]

```

```

6     dataX.append(a)
7     dataY.append(dataset[i + look_back, 0])
8 return numpy.array(dataX), numpy.array(dataY)
9

1 # semilla aleatoria para reproducibilidad
2 numpy.random.seed(7)

1 # Se normaliza el conjunto de datos. Nota: El método utilizado para la red neuronal es muy sensible a la escala, especialmente,
2 scaler = MinMaxScaler(feature_range=(0, 1))
3 ipvn = scaler.fit_transform(ipvn)

1 # Se dividen los datos entre entrenamiento y test. Se utiliza un 80% para el entrenamiento y un 20% para la validación
2 train_size = int(len(ipvn) * 0.8)
3 test_size = len(ipvn) - train_size

```

Es importante considerar que, en este tipo de redes neuronales, los datos deben estar de la forma: muestras, pasos de tiempo y características. Por defecto, los datos vienen de la forma, muestras (trimestre) y características (valor del índice). Sin embargo, se le puede agregar el paso del tiempo, tal como se detallará más adelante.

```

1 #Se construye la matriz de datos de entrenamiento y testeo
2 train, test = ipvn[0:train_size,:], ipvn[train_size:len(ipvn),:]

1 #Se pasan a una array
2 train=np.array(train)
3 test=np.array(test)

1 # Se remodela X=t y Y=t+look_back, con look_back=3.
2 look_back = 3
3 trainX, trainY = create_dataset(train, look_back)
4 testX, testY = create_dataset(test, look_back)

1 #Se valida cómo quedan organizados los datos
2 m=pd.DataFrame(trainX)
3 m.head(10)

```

	0	1	2
0	0.000000	0.008853	0.014755
1	0.008853	0.014755	0.035769

```

1 # remodelamos la entrada para que sea muestras, pasos de tiempo, características
2 trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
3 testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

4 0.055531 0.075114 0.094250

```

## ▼ Creación de la red neuronal

```
5 0.001250 0.105517 0.114510
```

Crearemos la red de la siguiente manera:

- 1 capa visible de entrada
- 1 capa oculta con 8 neuronas
- 1 capa de salida
- 100 iteraciones
- Métrica de pérdida: error porcentual medio absoluto
- Función de optimización: Adam

Nota: estos parámetros se obtienen después de realizar varias iteraciones con diferentes combinaciones

El modelo se creo en otro script, al cual se puede acceder [aquí](#), el modelo se guardó en el script anteriormente mencionado, y se leyó para mostrar sus métricas de evaluación acá.

```

# creamos la LSTM network
model = Sequential()
model.add(LSTM(8, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='Adam')
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)

```

```

1 # Recrea exactamente el mismo modelo solo desde el archivo
2 from tensorflow import keras
3 from tensorflow.keras import layers
4 #new_model = keras.models.load_model('/content/drive/MyDrive/MAESTRIA/Semestre2_2020_2/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')
5 #new_model = keras.models.load_model('/content/drive/MyDrive/MAESTRIA/Semestre2_2020_2/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')
6 new_model = keras.models.load_model('/content/drive/MyDrive/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')

1 def mape(actual, pred):
2     actual, pred = np.array(actual), np.array(pred)
3     return np.mean(np.abs((actual - pred) / actual)) * 100

```

```

1 # Se hacen las predicciones
2 trainPredict = new_model.predict(trainX)
3 testPredict = new_model.predict(testX)

1 # Se transforman las predicciones para compararla con la variable original de IPVN
2 trainPredict_t = scaler.inverse_transform(trainPredict)
3 trainY_t = scaler.inverse_transform([trainY])
4 testPredict_t = scaler.inverse_transform(testPredict)
5 testY_t = scaler.inverse_transform([testY])

```

## ▼ Evaluación

```

1 # Mape de test
2 MAPE_RNU=mape(testY_t[0],testPredict_t[:,0])
3 MAPE_RNU

```

**0.5311476202930392**

```
1 tabla.loc[4] = ["Red neuronal univariante", MAPE_RNU.round(3)]
```

```

1 # Mape de entrenamiento
2 mape(trainY_t,trainPredict_t.T)

```

**0.7715475476923558**

Se seleccionó este modelo como el que mejor MAPE tuvo.

```

1 # Se calcula el error cuadrático medio
2 trainScore = math.sqrt(mean_squared_error(trainY_t, trainPredict_t.T))
3 print('Resultado del entrenamiento: %.2f RMSE' % (trainScore))

```

**Resultado del entrenamiento: 0.76 RMSE**

```

1 testScore = math.sqrt(mean_squared_error(testY_t, testPredict_t.T))
2 print('Resultado del test: %.2f RMSE' % (testScore))

```

**Resultado del test: 1.15 RMSE**

```

1 pred = pd.Series(testPredict_t[:,0], name = "Prediccion")
2 obs = pd.Series(testY_t[0], name = "Real")

```

```

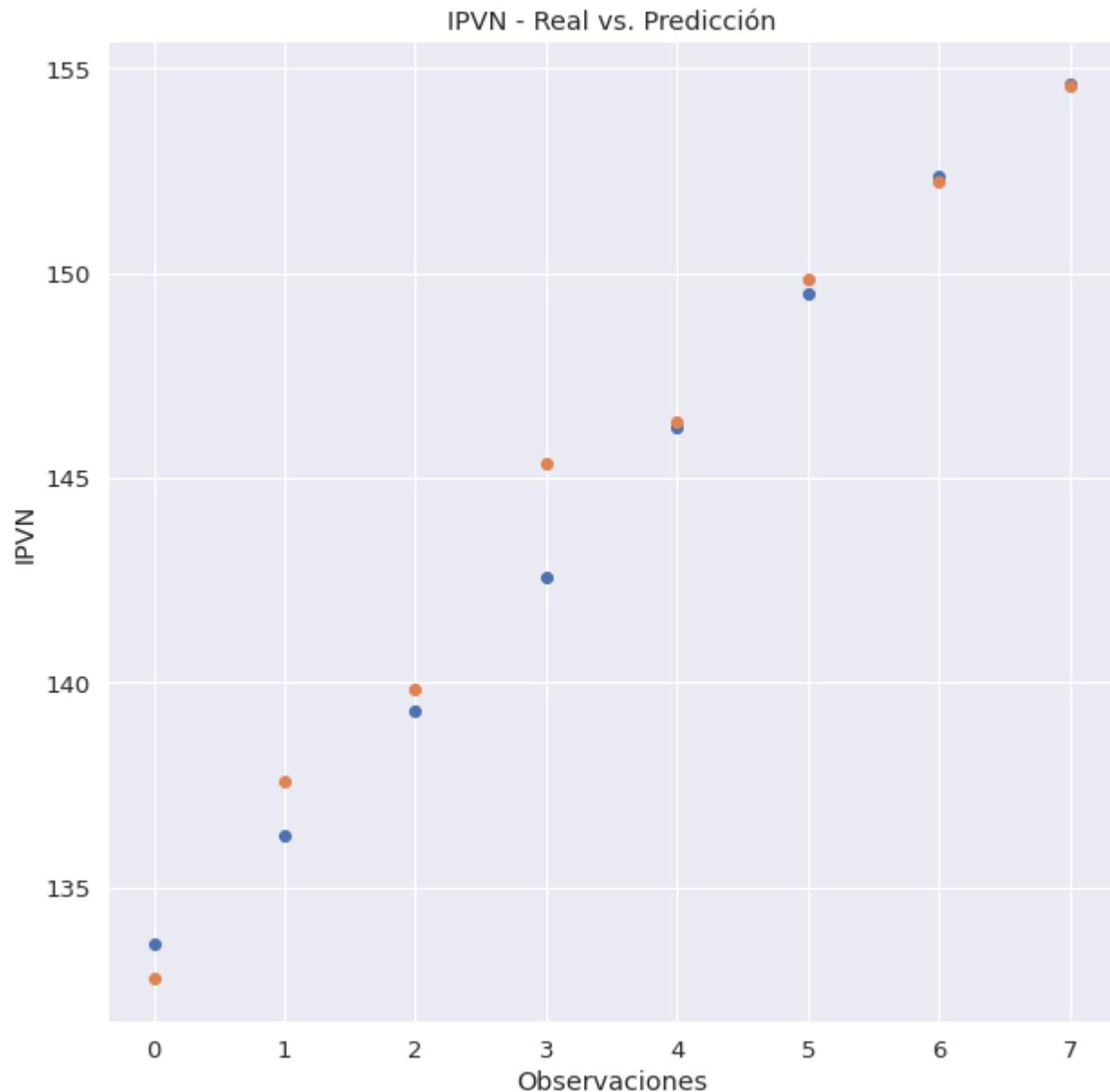
1 df_resultado = pd.concat([pred, obs], axis=1)
2 df_resultado

```

	Prediccion	Real
0	133.600677	132.79
1	136.246857	137.56
2	139.285583	139.81
3	142.545349	145.35
4	146.250214	146.34

```
1 plt.plot(df_resultado,"o")
2 plt.title("IPVN - Real vs. Predicción")
3 plt.xlabel("Observaciones")
4 plt.ylabel("IPVN")
5 plt.figure(figsize=(18,4))
```

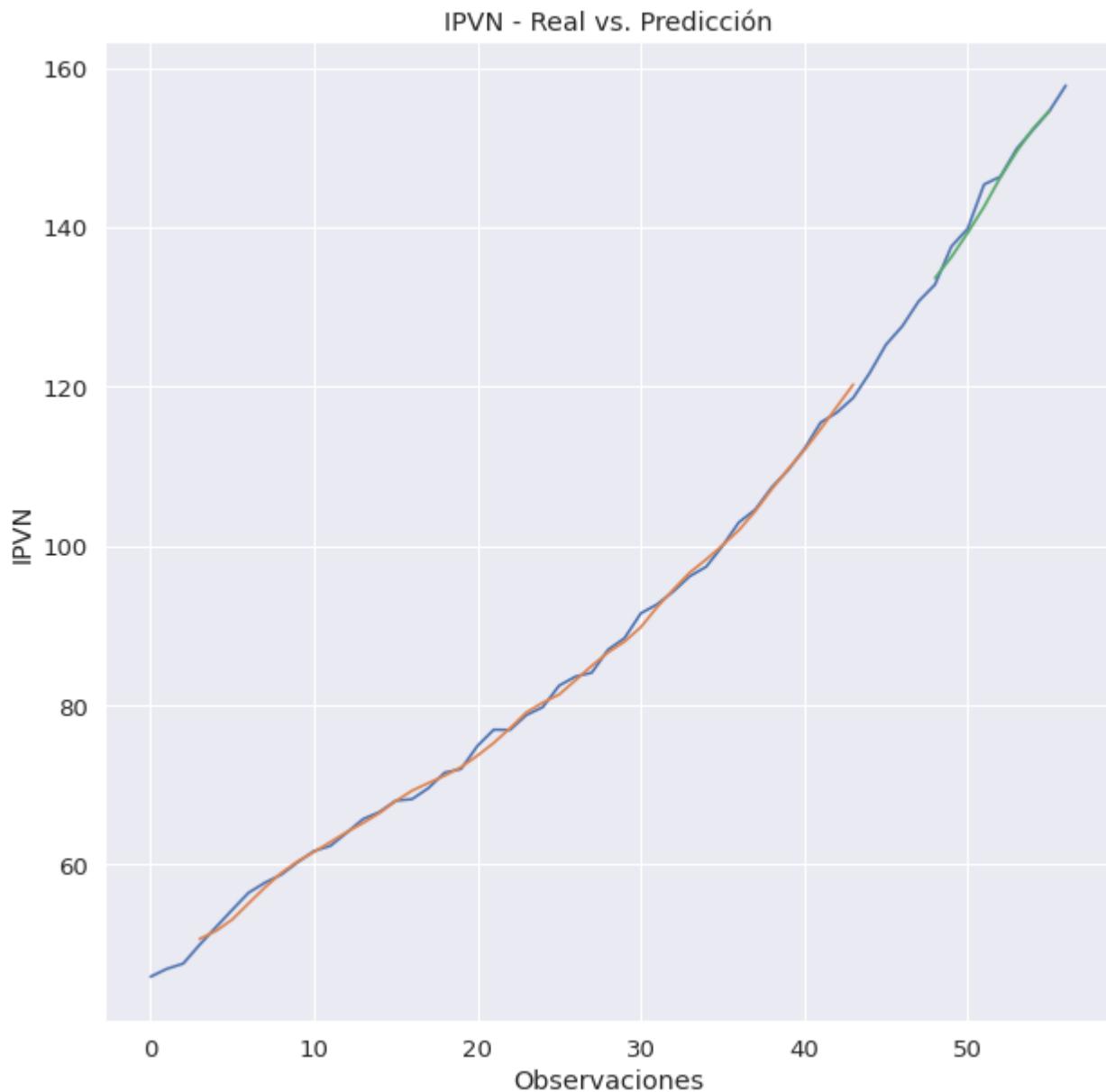
<Figure size 1296x288 with 0 Axes>



<Figure size 1296x288 with 0 Axes>

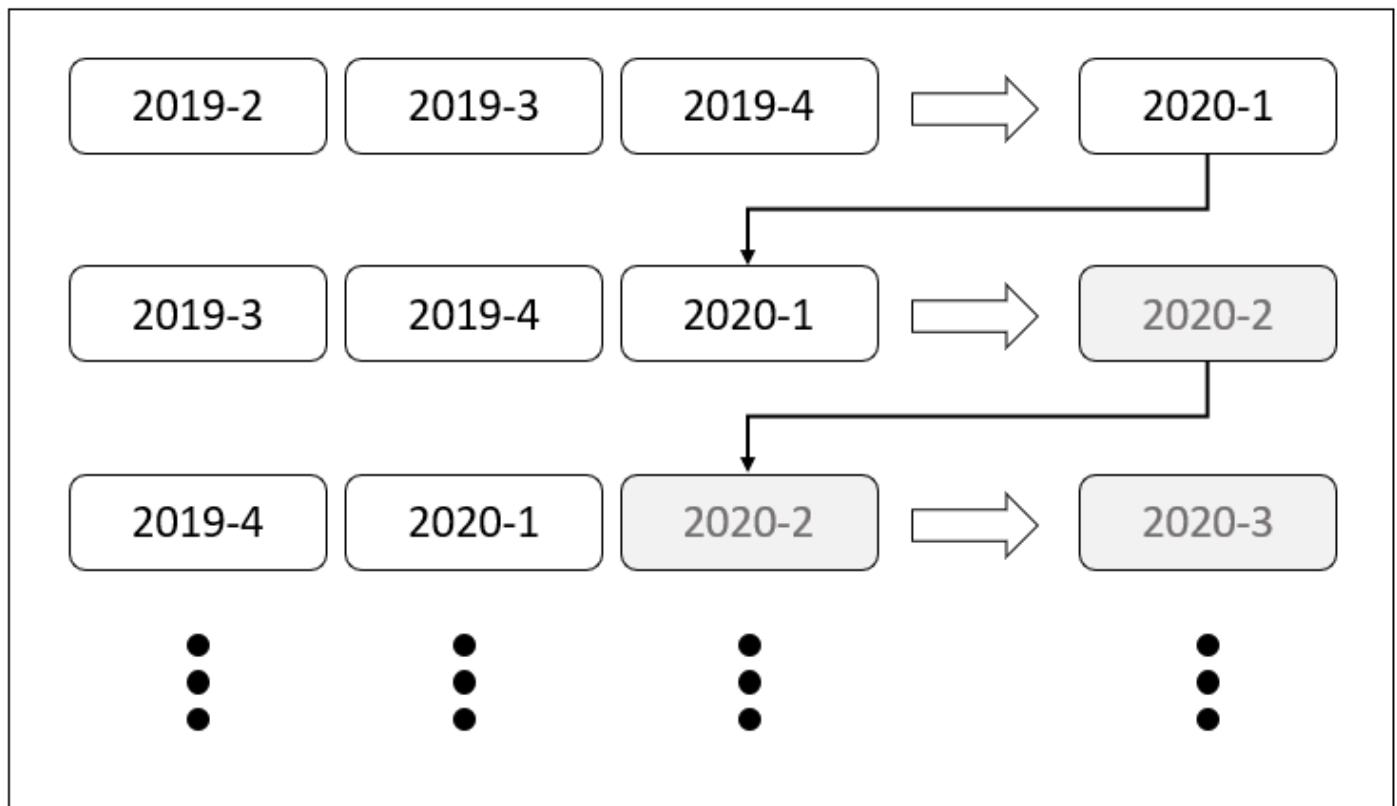
Se observa visualmente una alta precisión, especialmente, para las últimas cuatro observaciones

```
1 #Se grafica el ajuste y la predicción
2
3 # shift train predictions for plotting
4 trainPredictPlot = numpy.empty_like(ipvn)
5 trainPredictPlot[:, :] = numpy.nan
6 trainPredictPlot[look_back:len(trainPredict_t)+look_back, :] = trainPredict_t
7 # shift test predictions for plotting
8 testPredictPlot = numpy.empty_like(ipvn)
9 testPredictPlot[:, :] = numpy.nan
10 testPredictPlot[len(trainPredict_t)+(look_back*2)+1:len(ipvn)-1, :] = testPredict_t
11 # plot baseline and predictions
12 plt.plot(scaler.inverse_transform(ipvn))
13 plt.plot(trainPredictPlot)
14 plt.plot(testPredictPlot)
15 plt.title("IPVN - Real vs. Predicción")
16 plt.xlabel("Observaciones")
17 plt.ylabel("IPVN")
18 plt.show()
```



## ▼ Pronósticos finales

Para predecir el comportamiento del IPVN con la red neuronal, se ajusta el modelo a todo el conjunto de datos y se realiza la predicción de la siguiente manera:



Con los últimos dos datos y la predicción del trimestre anterior, se predice el siguiente trimestre.

```
1 ipvn = df_indices.d_IPVN_MDE_Num
```

```
1 ipvn=pd.DataFrame(ipvn)
2 ipvn = ipvn.iloc[4:, :]
```

```
1 ipvn.head()
```

### d\_IPVN\_MDE\_Num

<b>4</b>	45.94
<b>5</b>	46.93
<b>6</b>	47.59
<b>7</b>	49.94
<b>8</b>	52.15

1 #Se cargan las librerías para correr la red neuronal

<https://colab.research.google.com/drive/1a-JSJAllyqTR8EM3hAjsHVTW3TmaPV4q#scrollTo=tngb1GwVGzyx&printMode=true>

```

2 import numpy as np
3 import math
4 from keras.models import Sequential
5 from keras.layers import Dense
6 from keras.layers import LSTM
7 from sklearn.preprocessing import MinMaxScaler
8 from sklearn.metrics import mean_squared_error

1 # Con esta función, se convierte un array de valores en una matriz de conjuntos de datos
2 def create_dataset(dataset, look_back=1):
3     dataX, dataY = [], []
4     for i in range(len(dataset)-look_back-1):
5         a = dataset[i:(i+look_back), 0]
6         dataX.append(a)
7         dataY.append(dataset[i + look_back, 0])
8     return np.array(dataX), np.array(dataY)
9

1 # semilla aleatoria para reproducibilidad
2 np.random.seed(7)

1 #Se pasan a una array
2 ipvn = np.array(ipvn)

1 # Se normaliza el conjunto de datos. Nota: El método utilizado para la red neuronal es muy sensible a la escala, especialmente,
2 scaler = MinMaxScaler(feature_range=(0, 1))
3 ipvn = scaler.fit_transform(ipvn)

1 ipbn = np.array(ipvn)

1 # Se remoldela X=t y Y=t+look_back, con look_back=3.
2 look_back = 3
3 X, Y = create_dataset(ipbn, look_back)

1 # remodelamos la entrada para que sea muestras, pasos de tiempo, características
2 X= np.reshape(X, (X.shape[0], 1, X.shape[1]))

1 # Recrea exactamente el mismo modelo solo desde el archivo
2 from tensorflow import keras
3 from tensorflow.keras import layers
4 #new_model = keras.models.load_model('/content/drive/MyDrive/MAESTRIA/Semestre2_2020_2/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')
5 #new_model = keras.models.load_model('/content/drive/MyDrive/MAESTRIA/Semestre2_2020_2/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')
6 new_model_pred = keras.models.load_model('/content/drive/MyDrive/Proyecto Integrador Semestre 2/modelo/modelo0112S.h5')

1 new_model.fit(X, Y, epochs=100, batch_size=1, verbose=0)

<tensorflow.python.keras.callbacks.History at 0x7f52e70edb38>

1 # Se hacen las predicciones
2 ipvnPredict = new_model_pred.predict(X)

1 def mape(actual, pred):

```

```
2     actual, pred = np.array(actual), np.array(pred)
3     return np.mean(np.abs((actual - pred) / actual)) * 100

1 # Se transforman las predicciones para compararla con la variable original de IPVN
2 yhat_t = scaler.inverse_transform(ipvnPredict)
3 Y_t = scaler.inverse_transform([Y])

1 # predicción con todos los datos:
2 yhat_T = pd.Series(yhat_t.T[0], name = "Predicción")
3 yhat_T.tail()

48    142.545349
49    146.250214
50    149.511475
51    152.345917
52    154.594849
Name: Predicción, dtype: float32

1 Y_T = pd.Series(Y_t[0], name= "Real")
2 Y_T.tail()

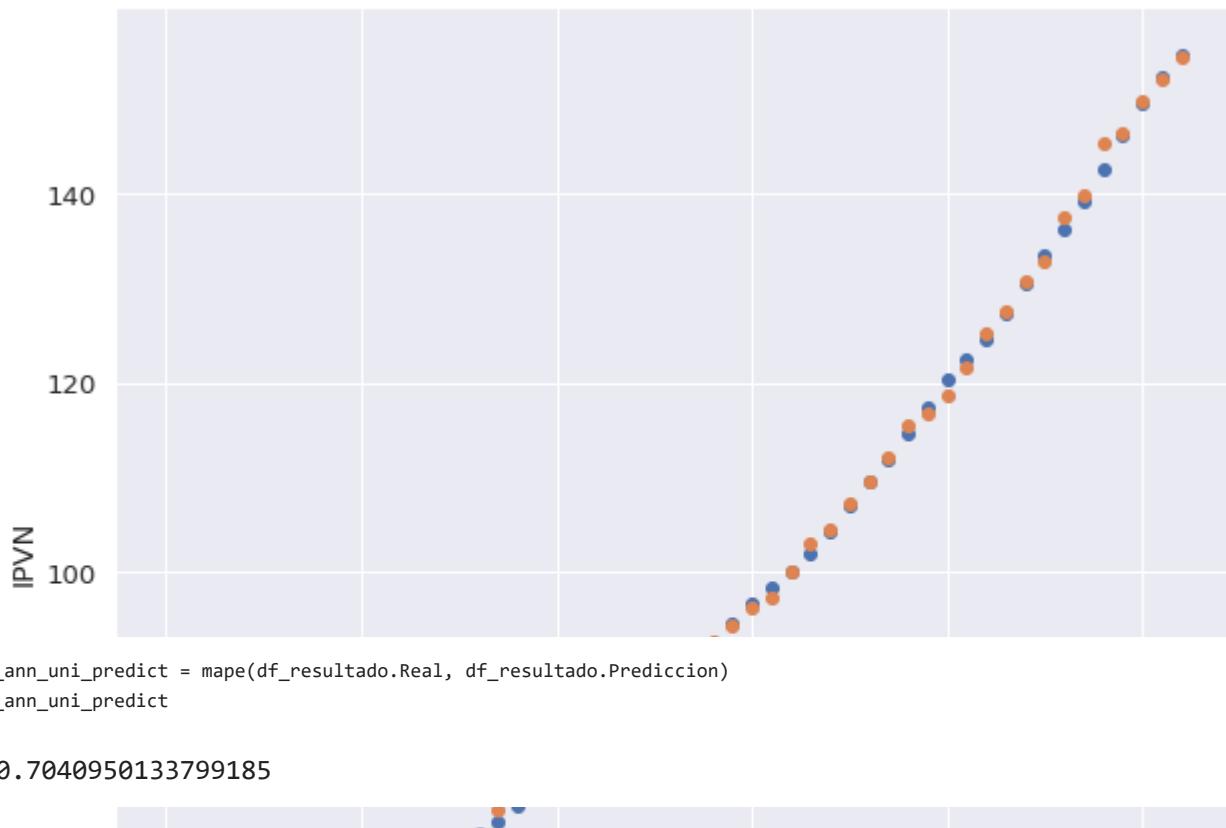
48    145.35
49    146.34
50    149.84
51    152.22
52    154.57
Name: Real, dtype: float64

1 import matplotlib.pyplot as plt
2 df_resultado = pd.concat([yhat_T,Y_T], axis=1)

1 plt.plot(df_resultado,"o")
2 plt.title("IPVN - Real vs. Predicción")
3 plt.xlabel("Observaciones")
4 plt.ylabel("IPVN")
5 plt.figure(figsize=(18,4))
```

&lt;Figure size 1296x288 with 0 Axes&gt;

IPVN - Real vs. Predicción



```

1 mape_ann_uni_predict = mape(df_resultado.Real, df_resultado.Prediccion)
2 mape_ann_uni_predict

```

```
0.7040950133799185
```

```

1 forecast = []
2 length = 1
3 n_features = 1
4 first_eval_batch = np.asarray(X[-length:])
5
6 current_batch = first_eval_batch.reshape(1,length,3)
7
8 for i in range(6):
9     current_pred = new_model.predict(current_batch)[0]
10
11    forecast.append(current_pred)
12
13    current_batch = np.array([current_batch[0,0,1], current_batch[0,0,2], current_pred[0]])
14    current_batch = current_batch.reshape(1,length,3)


```

```

1 forecast = scaler.inverse_transform(forecast)
2 forecast

```

```
array([[156.28651353],
       [159.22591094],
       [162.16808789],
       [165.03678342],
       [167.56982662],
       [169.94461545]])
```

```
1 df_ipvn = pd.read_csv("/content/drive/MyDrive/Proyecto Integrador Semestre 2/otros Datasets/proyeccion.csv")
```

```

1 trim2 = ['2006-01', '2008-01',
2          '2010-01', '2012-01',
3          '2014-01', '2016-01']

```

```

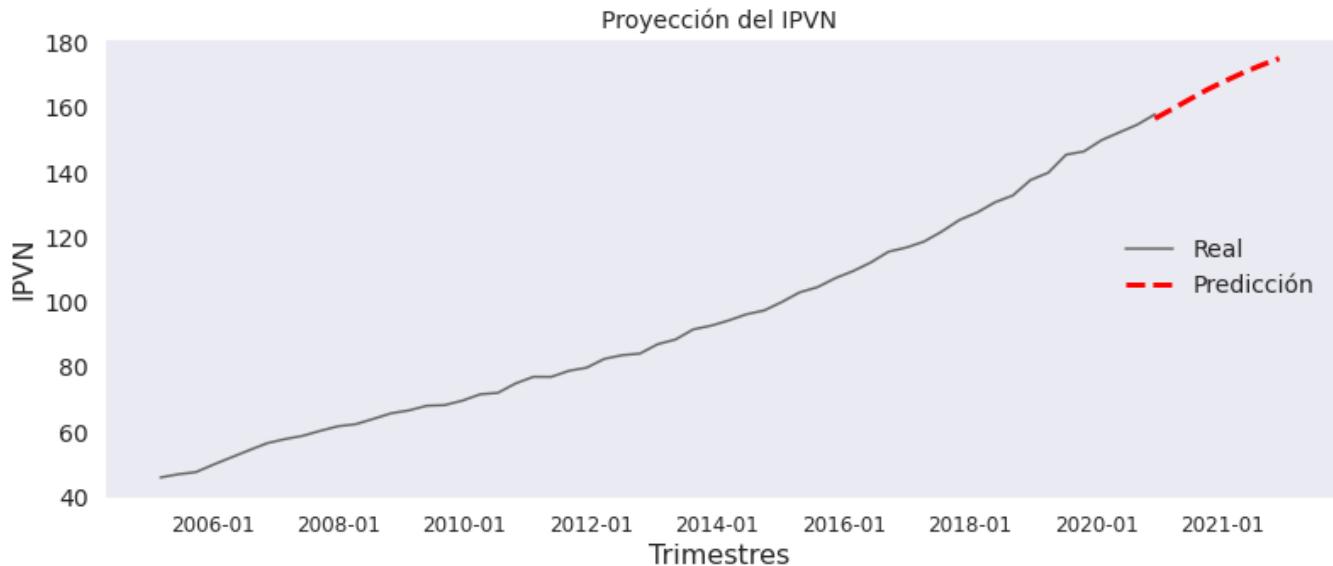
3      2014-01 , 2015-01 ,
4      '2018-01', '2020-01',
5      '2021-01']

```

```

1
2 plt.figure(figsize=(13,5))
3
4 SMALL_SIZE = 14
5 MEDIUM_SIZE = 16
6 BIGGER_SIZE = 20
7
8 plt.rc('font', size=SMALL_SIZE)         # controls default text sizes
9 plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
10 plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
11 plt.rc('xtick', labelsize=12)           # fontsize of the tick labels
12 plt.rc('ytick', labelsize=SMALL_SIZE)     # fontsize of the tick labels
13 plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
14 plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
15
16 plt.plot(df_ipvn.Periodo, df_ipvn.d_IPVN_MDE_Num, color="dimgray")
17 plt.plot(df_ipvn.Periodo, df_ipvn.Proyección, color="red", linewidth=3, linestyle='--')
18 plt.xticks(np.arange(2.9, len(df_ipvn.Periodo), (len(df_ipvn.Periodo)/len(trim2))), trim2)
19 plt.title("Proyección del IPVN")
20 plt.xlabel("Trimestres")
21 plt.ylabel("IPVN")
22 plt.grid(b=None)
23 plt.legend(["Real", "Predicción"], loc='center right', frameon=False)
24 plt.show()

```



## ▼ Evaluación de modelos

Con el fin de determinar el modelo de mejor pronóstico, se utilizó como criterio el error porcentual medio absoluto- MAPE. Al respecto, estos fueron los resultados obtenidos:

1 tabla

	Modelo	Error porcentual medio absoluto - MAPE
1	Regresión lineal multivariada	2.301
2	Red neuronal multivariada	2.284
3	Serie de tiempo univariante	0.617
4	Red neuronal univariante	0.531

En términos generales, las técnicas univariantes presentaron una mayor precisión. Tanto en la modelación multivariada como univariante, las redes neuronales presentaron mejores resultados frente a las técnicas estadísticas tradicionales, sin embargo, requieren de un mayor esfuerzo computacional y de una mayor experimentación con sus implicaciones de tiempo.

En definitiva, el modelo con mayor precisión, medido a través del MAPE, fue la red neuronal univariante.

## ▼ Conclusiones

-Para el caso particular de la serie de Índice de Precios de la Vivienda Nueva - IPVN para Medellín y el área metropolitana, el enfoque de modelación univariante ofrece una mayor precisión de pronóstico frente al enfoque multivariado. Sin embargo, no permite entender la incidencia de otras variables sobre su comportamiento, limitando el entendimiento del indicador a nivel de variables de oferta y demanda.

-En el caso particular de las redes neuronales, si bien presentan en ambos enfoques una mayor precisión frente a los métodos estadísticos tradicionales, requieren computacionalmente de más tiempo y rendimiento. Por otro lado, para la definición de parámetros, no hay claridad sobre unas reglas prácticas para llegar a estos sin necesidad de la experimentación.

-Si bien, se ofrecen resultados a nivel del índice, se implementaron modelos con las variaciones interanuales, obteniéndose precisiones bajas, y con un mayor esfuerzo computacional en el caso de las redes neuronales. Destacando que con el pronóstico del índice también se pueden obtener las variaciones, y con un nivel de precisión mayor.

Una manera para que las redes neuronales a nivel multivariado obtengan una mayor precisión y menores costos computacionales, es trabajar con las variables obtenidas en la ingeniería de características. Por ende, implementar dentro de la red neuronal este proceso de selección, representaría una oportunidad de mejora para el problema acá analizado.

-Es importante validar en una regresión lineal no solo los supuestos que son necesarios para extender los resultados a inferencias, si no también la sensibilidad del modelo, para identificar si la cantidad de datos tiene un impacto sobre las variables escogidas; esto puede dar indicios de qué variables finalmente tomar para realizar la construcción del modelo.

-Realizar un análisis de los resultados obtenidos en los betas de un modelo de regresión lineal, ayuda a tener una mejor comprensión del impacto de las variables del modelo sobre la variable respuesta, este ejercicio invita a realizar una mejor comprensión de los resultados y tener un cuestionamiento de los mismos, retando al investigador a indagar por qué dichos resultados tienen sentido o no.

## ▼ Referencias

A simple (and interpretable) Neural Network model for House Pricing regression: <https://n9.cl/1du0>

Castaño, Elkin (2020). Memorias del curso de Series de Tiempo. Universidad Eafit, Maestría de Finanzas.

Unipython (2020). Predicción con series temporales con LSTM, Redes Neuronales Recurrentes. Disponible en: <https://unipython.com/prediccion-con-series-temporales-con-lstm-redes-neuronales-recurrentes/>

Torres, Jordi (2019). Redes neuronales recurrentes. Disponible en: Redes Neuronales Recurrentes - Jordi TORRES.AI

Wei, W. (2006) Time Series Analysis: Univariate and Multivariate Methods.