Capitulo 2

Revisión de Estadistica y probabilidad

2.1 Introducción

Comenzamos con una breve revisión de la estadística, para presentar algunos de los conceptos claves y la notación que usamos en este y los siguientes capítulos. La Estadística nos ayuda con la **recopilación y análisis de datos** con el fin de obtener información, dibujar, obtener conclusiones y apoyo a la toma de decisiones. Los métodos estadísticos son necesarios cuando Tenemos información incompleta sobre un fenómeno. Típicamente tenemos incompleta la información porque no podemos recopilar datos de todos los miembros de una población o si hay incertidumbre en las observaciones que hacemos (por ejemplo, debido a la medición y el ruido). Cuando no podemos encuestar a una población completa, una muestra elegida al azar puede estudiarse en su lugar, y podemos usar métodos estadísticos y calcular estadística descriptiva con (parámetros como la media y la desviación estándar) para hacer inferencias sobre las propiedades de toda la población (también llamado espacio muestral) de manera sistemática.

2.1.1 Población y muestra

La población es el conjunto total de individuos, objetos, eventos, o medidas de consideración que tienen las mismas características y sobre el cual estamos interesados en obtener conclusiones.

“Es demasiado grande para poder estudiarlos”

La muestra es un subconjunto de la población al que tenemos accesos y sobre el cual realmente deberemos estudiar las características de interés de la población.

“la muestra debería ser representativa”

2.1.2 Los Datos.

Hanks, puntualiza que a menudo pasamos por alto el hecho de que los datos significan algo (“La materia prima de la estadística y la simulación”) y que es importante entender su significado. Tenemos que mirar más allá de los números y comprender lo que representan si vamos a dar valor agregado a la resolución del problema, “no tiene nada que ver con algoritmos o ingeniería ni nada de eso”, la comprensión de los datos es un arte y muy importante.

Aquí es donde los analista de datos, científicos de datos, abren su caja de herramientas para encontrar el enfoque de análisis y algoritmos para trabajar con los datos. Hay cientos de técnicas para procesar los datos y obtener información, para la toma de decisiones. La investigación de Operaciones, la teoría de decisión, teoría de los juegos, la Simulación, Data Mining, Machine Learning y todas las metodologías que han existido desde hace tiempo. “Una vez que entiendes los datos y conoces el problema que estamos tratando de resolver, es cuando se puede elegir el algoritmo y obtener la solución óptima”.

Wikipedia, nos dice que un **dato** es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa. Los Datos cualitativos nos describen el aspecto de un objeto o fenómeno mediante alguna característica (cualidad) especifica (Ejemplo el estado civil de las personas, sexo, estrato social). Los datos cuantitativos, podemos expresar de forma numérica la magnitud de cierta característica de un objeto o fenómeno.

2.1.3 Fuentes de datos

Los datos pueden tener diversas fuentes a las que se puede acceder directa o indirectamente. La calidad, pertinencia y veracidad de un dato depende en gran medida de su origen y el modo en el que éste se obtiene. Una de las bases del método científico es la de poder identificar, aislar y de ser posible, reproducir la fuente de los datos que son objetos de un estudio.

2.2. Análisis estadísticos de datos.

No es objetivo específico de este libro detallar con profundidad la estadística pero básicamente todo técnico en simulación debe conocer cómo realizar un trabajo estadístico o una investigación estadística:

1. Diseño formulación del análisis estadístico: en esta actividad se planifica las tareas que debemos realizar y en muchos de los casos se debe definir la hipótesis.

2. Observación estadística y toma de datos: se realiza una exploración exhaustiva de todos los elementos de la población, la observación parcial solo se observa una parte de los elementos de la población "Muestra", se utiliza cuando es necesario disponer de una parte representativa de la población.

3. Exploración de los datos: aquí realizamos la reducción estadística que describimos, resumimos, tabulamos, graficamos los datos para mirarlos desde diferentes ángulos, permitiendo tener datos de calidad.

4. Modelado de los datos: una vez que hemos generado los diferentes estadísticos ya podemos observar el comportamiento de los datos y podemos utilizar los diferentes modelos estadísticos dando así respuestas de predicciones, relaciones y correlaciones de los diferentes atributos de los datos.

5. Realizar estimaciones: aquí realizamos generalizaciones, estimaciones, pronósticos y otras acciones con los atributos de los datos permitiendo tener información variada para poder realizar conclusiones y análisis de resultados.

6. Análisis de Resultados: realizamos un análisis de los datos de los informes y estimaciones para determinar si son correctos y confirmar o rechazar la hipótesis.

2.2.1 Datos continuos y Datos discretos.

Los datos cuantitativos se dividen en datos continuos que puede admitir cualquier valor intermedio dentro de un intervalo de los números reales. Por ejemplo el rendimiento académico, la estatura, el peso, salario, tiempo, volumen que pueden ser obtenidos por medición. Los datos discretos son aquella que admite interrupciones verdaderas en su medición y por lo tanto no admite valores intermedios. Por ejemplo, número de alumnos de la clase, el número de hijos de una familia, el número de clientes en espera en un almacén.

2.3 Estadística descriptiva.

Técnicas para resumir y describir datos cuantitativos: Esta descripción le informará de la localización, dispersión, forma de la distribución de sus datos, utilizando técnicas como Tablas de frecuencia, Gráficos: Barras, sectorial, polígono de frecuencias, curva normal. Descripciones numéricas. Promedios (media, mediana y moda), medidas de variabilidad (desviación estándar, varianza), medida de la relación entre las variables, etc.

Para el análisis de datos del estudio, se tomará la información de los registros diarios de la atención en emergencias del Hospital Rodríguez Zambrano en los meses de abril y mayo del 2017. Tomando en cuenta el total de pacientes que ingresan, los atendidos y no atendidos.

Tabla 2.1 fuentes de datos

Día Semana	FECHA	TOTAL_PACIENTES	ATENDIDOS	NO-ATENDIDOS
Sábado	01/04/2017	93	62	31
Domingo	02/04/2017	111	86	25
Lunes	03/04/2017	147	79	68
Martes	04/04/2017	161	123	38
Miercoles	05/04/2017	124	100	24
Jueves	06/04/2017	159	130	29
Viernes	07/04/2017	148	113	35
Sábado	08/04/2017	89	75	14
Domingo	09/04/2017	68	56	12
Lunes	10/04/2017	167	116	51
Martes	11/04/2017	136	100	35
Miercoles	12/04/2017	128	117	11
Jueves	13/04/2017	110	93	17
Viernes	14/04/2017	88	58	30
Sábado	15/04/2017	123	89	34
Domingo	16/04/2017	136	116	20
Lunes	17/04/2017	189	149	40
Martes	18/04/2017	151	111	40
Miercoles	19/04/2017	130	95	35
Jueves	20/04/2017	122	108	14
Viernes	21/04/2017	110	84	26
Sábado	22/04/2017	94	62	32
Domingo	23/04/2017	102	71	31
Lunes	24/04/2017	166	130	36
Martes	25/04/2017	122	86	36
Miercoles	26/04/2017	108	89	19
Jueves	27/04/2017	120	99	21
Viernes	28/04/2017	116	91	25
Sábado	29/04/2017	90	63	27
Domingo	30/04/2017	98	64	34
Lunes	01/05/2017	150	121	29
Martes	02/05/2017	159	118	40
Miercoles	03/05/2017	180	143	37
Jueves	04/05/2017	194	141	53
Viernes	05/05/2017	150	109	41
Sábado	06/05/2017	148	106	42
Domingo	07/05/2017	137	105	32
Lunes	08/05/2017	182	140	42
Martes	09/05/2017	176	143	32
Miercoles	10/05/2017	164	125	38
Jueves	11/05/2017	182	142	40
Viernes	12/05/2017	182	151	31
Sábado	13/05/2017	146	114	32
Domingo	14/05/2017	99	71	28
Lunes	15/05/2017	188	134	52
Martes	16/05/2017	178	137	41
Miercoles	17/05/2017	166	121	44

Observación estadística de los datos.-

Por ahora tenemos en forma natural los datos obtenidos del registro de datos de la institución en la tabla 2.1 que no constituyen una fuente de mucha información, los registros en forma de tabla tienen los siguientes metadatos:

- Día de la semana
- FECHA
- TOTAL_PACIENTES
- ATENDIDOS
- NO-ATENDIDOS

La **Distribución de frecuencias** separa los datos en clases y muestra el número de ocurrencias en cada clase, o frecuencia de clase, este tipo de agrupación facilita la interpretación y la presentación en cuadros numéricos, que luego se representan en gráficos.

Una de las mejores maneras de describir una variable es representar los valores que aparecen en el conjunto de datos y el número de veces que aparece cada valor. La representación más común de una distribución es un histograma, que es un gráfico que muestra la frecuencia de cada valor.

Histogramas

En Python, podemos graficar fácilmente un histograma con la ayuda de la función **hist** de matplotlib, simplemente debemos pasarle los datos y la cantidad de contenedores en los que queremos dividirlos.

Aqui vamos a construir un Histograma de frecuencias:

- importamos:
- pandas
- matplotlib
- numpy
- Creamos un DataFrame con Pandas
- Presentamos los datos

La biblioteca Matplotlib.

Matlplotib es la biblioteca de graficación basada en Python más popular y sobre la que una gran catidad de proyectos se basan para despliegue de gráficos y visualización de datos.

In [1]:

```
# importamos la libreria Pandas, matplotlib y numpy  que van a ser de mucha utilidad
 para poder hacer gráficos

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# leemos los datos de la tabla del directorio Data de trabajo
datos = pd.read_csv('C:/Users/JORGEANIBAL/LIBRO/Data/datos.csv')
#Presentamos los datos en un DataFrame de Pandas
datos
```

Out[1]:

	DIASEMANA	FECHA	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
0	Sábado	01/04/2017	93	62	31
1	Domingo	02/04/2017	111	86	25
2	Lunes	03/04/2017	147	79	68
3	Martes	04/04/2017	161	123	38
4	Miercoles	05/04/2017	124	100	24
5	Jueves	06/04/2017	159	130	29
6	Viernes	07/04/2017	148	113	35
7	Sábado	08/04/2017	89	75	14
8	Domingo	09/04/2017	68	56	12
9	Lunes	10/04/2017	167	116	51
10	Martes	11/04/2017	136	100	35
11	Miercoles	12/04/2017	128	117	11
12	Jueves	13/04/2017	110	93	17
13	Viernes	14/04/2017	88	58	30
14	Sábado	15/04/2017	123	89	34
15	Domingo	16/04/2017	136	116	20
16	Lunes	17/04/2017	189	149	40
17	Martes	18/04/2017	151	111	40
18	Miercoles	19/04/2017	130	95	35
19	Jueves	20/04/2017	122	108	14
20	Viernes	21/04/2017	110	84	26
21	Sábado	22/04/2017	94	62	32
22	Domingo	23/04/2017	102	71	31
23	Lunes	24/04/2017	166	130	36
24	Martes	25/04/2017	122	86	36
25	Miercoles	26/04/2017	108	89	19
26	Jueves	27/04/2017	120	99	21
27	Viernes	28/04/2017	116	91	25
28	Sábado	29/04/2017	90	63	27
29	Domingo	30/04/2017	98	64	34
...	...	...	...	...	...
31	Martes	02/05/2017	159	118	40
32	Miercoles	03/05/2017	180	143	37
33	Jueves	04/05/2017	194	141	53
34	Viernes	05/05/2017	150	109	41
35	Sábado	06/05/2017	148	106	42
36	Domingo	07/05/2017	137	105	32
37	Lunes	08/05/2017	182	140	42
38	Martes	09/05/2017	176	143	32

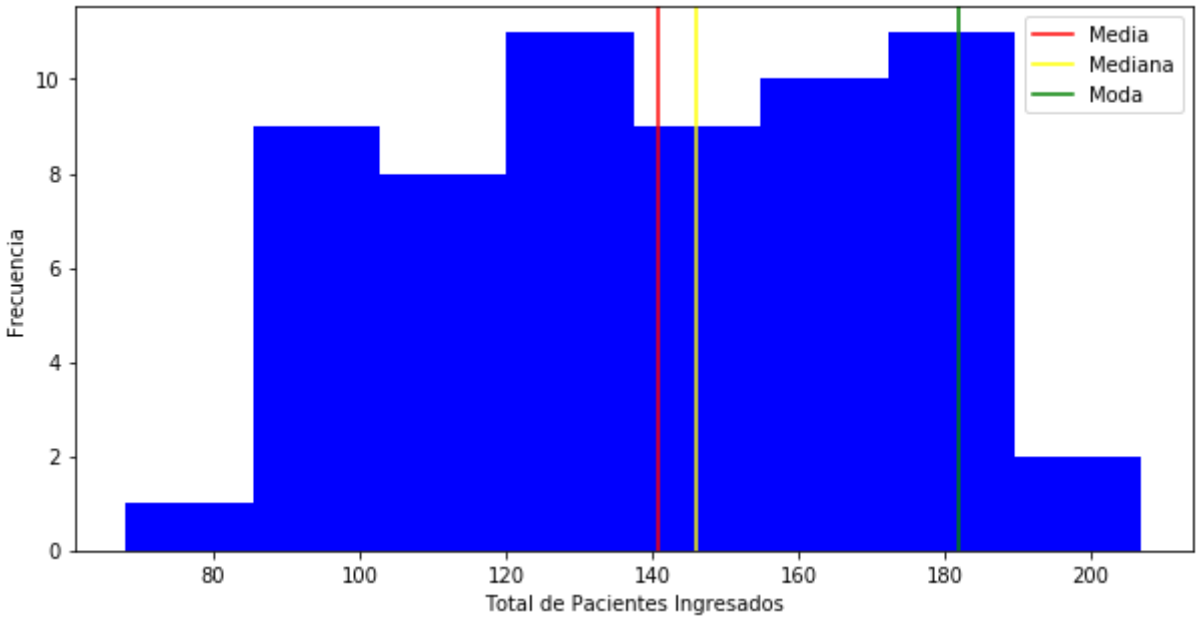
	DIASEMANA	FECHA	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
39	Miercoles	10/05/2017	164	125	38
40	Jueves	11/05/2017	182	142	40
41	Viernes	12/05/2017	182	151	31
42	Sábado	13/05/2017	146	114	32
43	Domingo	14/05/2017	99	71	28
44	Lunes	15/05/2017	188	134	52
45	Martes	16/05/2017	178	137	41
46	Miercoles	17/05/2017	166	121	44
47	Jueves	18/05/2017	161	124	36
48	Viernes	19/05/2017	173	109	64
49	Sábado	20/05/2017	113	71	42
50	Domingo	21/05/2017	94	61	32
51	Lunes	22/05/2017	178	146	32
52	Martes	23/05/2017	207	152	55
53	Miercoles	24/05/2017	177	129	48
54	Jueves	25/05/2017	111	86	25
55	Viernes	26/05/2017	133	95	38
56	Sábado	27/05/2017	137	97	40
57	Domingo	28/05/2017	157	110	47
58	Lunes	29/05/2017	162	119	43
59	Martes	30/05/2017	140	122	18
60	Miercoles	31/05/2017	154	129	18

61 rows × 5 columns

Preparamos el Gráfico

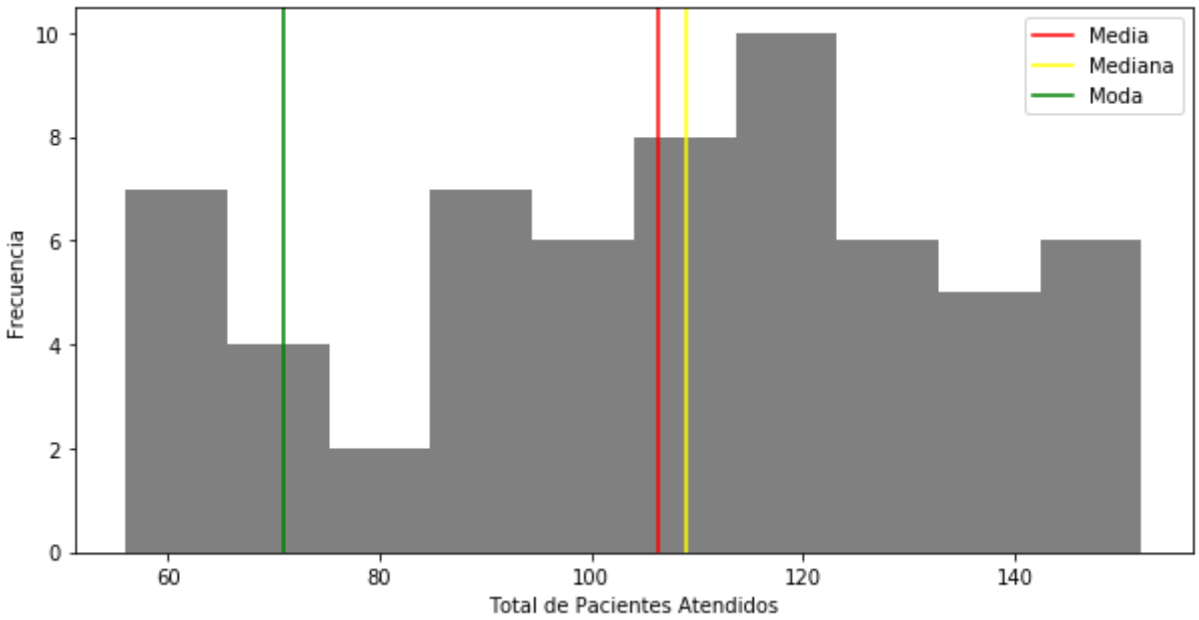
Utilizamos la biblioteca **matplotlib.pyplot.hist(x,bins,Range=None,Color=None....)** donde **x** son los datos que vamos a graficar, **bins=8** el número de clases, y ademas el **color=blue**, en el gráfico tambien incorporamos la media **xmean()** , la mediana **x.median()**, y la moda **x.mode()[0]**, utilizando **plt.axvline**

In [2]:
```
# Preparando para el grafico para la columna TOTAL PACIENTES
x=datos["TOTAL_PACIENTES"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=8,color='blue')
plt.axvline(x.mean(),color='red',label='Media')
plt.axvline(x.median(),color='yellow',label='Mediana')
plt.axvline(x.mode()[0],color='green',label='Moda')
plt.xlabel('Total de Pacientes Ingresados')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```



In [3]:
```
# Preparando para el grafico PARA LA COLUMNA ATENDIDOS
# Aqui cambiamos el color del histograma, y para las clases dejamos que matplotlib lo asigne

x=datos["ATENDIDOS"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=None,color='grey')
plt.axvline(x.mean(),color='red',label='Media')
plt.axvline(x.median(),color='yellow',label='Mediana')
plt.axvline(x.mode()[0],color='green',label='Moda')
plt.xlabel('Total de Pacientes Atendidos')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```



MEDIDAS DE TENDENCIA CENTRAL

MEDIA, MEDIANA, MODA

calculo de la media

es una medida de uso común, en una muestra de **n** datos, la media muestra el promedio aritmético simple de los datos utilizamos mean() para su calculo

In [4]:
```
# enviando las medias a t1, t2, t3 para su utilización
print("Media:", )

t1, t2, t3 = datos.mean()
print( "la Media de TOTAL PACIENTES:   ", t1)
print( "la Media de ATENDIDOS      :   ", t2)
print( "la Media de NO ATENDIDOS   :   ", t3)

print("DIRECTAMENTE DEL DATAFRAME ")
print()
datos.mean()
```

```
Media:
la Media de TOTAL PACIENTES:    140.72131147540983
la Media de ATENDIDOS      :    106.32786885245902
la Media de NO ATENDIDOS   :    34.131147540983605
DIRECTAMENTE DEL DATAFRAME
```

Out[4]:
```
TOTAL_PACIENTES    140.721311
ATENDIDOS          106.327869
NO_ATENDIDOS        34.131148
dtype: float64
```

calculo de la mediana

la mediana es el valor ubicado en el centro de los datos ordenados

In [5]:
```
# enviando las medias a m1, m2, m3 para su utilización
print("Mediana:", )

m1, m2, m3 = datos.median()

print( "la Mediana de TOTAL PACIENTES:   ", m1)
print( "la Mediana de ATENDIDOS      :   ", m2)
print( "la Mediana de NO ATENDIDOS   :   ", m3)

print("DIRECTAMENTE DEL DATAFRAME ")
print()
datos.median()
```

```
Mediana:
la Mediana de TOTAL PACIENTES:    146.0
la Mediana de ATENDIDOS      :    109.0
la Mediana de NO ATENDIDOS   :    34.0
DIRECTAMENTE DEL DATAFRAME
```

Out[5]:
```
TOTAL_PACIENTES    146.0
ATENDIDOS          109.0
NO_ATENDIDOS        34.0
dtype: float64
```

calculo de la moda

La moda es el dato que ocurre con más frecuencia en la muestra, puede existir o no, o también puede existir más de una moda

In [6]:
```
print("Moda:")

mo1 = datos["TOTAL_PACIENTES"].mode()
mo2 = datos["ATENDIDOS"].mode()
mo3 = datos["NO_ATENDIDOS"].mode()

print( "la Moda de TOTAL PACIENTES:   ", mo1)
print( "la Moda de ATENDIDOS      :   ", mo2)
print( "la Moda de NO ATENDIDOS   :   ", mo3)


pd.DataFrame(mo1,mo2,mo3)
```

```
Moda:
la Moda de TOTAL PACIENTES:    0    182
dtype: int64
la Moda de ATENDIDOS      :    0    71
1    86
dtype: int64
la Moda de NO ATENDIDOS   :    0    32
dtype: int64
```

Out[6]:

	32
71	NaN
86	NaN

Aqui observamos que para la variable ATENDIDOS tiene 2 modas la una de 71 y 86

Resumen de medidas de tendencia central

los DataFrame de pandas permite entregar un informe completo de todas las medidas de tendencia central utilizando la función **describe()** donde nos presentará todas las medidas de tendencia central

In [7]:
```
#  Tomamos los datos de las columnas
datos[['TOTAL_PACIENTES','ATENDIDOS','NO_ATENDIDOS']].describe()
# describe(), nos presenta directamente la media, desviación standar, el valor mínim
o, valor máximo, el 1er cuartil, 2do Cuartil, 3er Cuartil
```

Out[7]:

	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
count	61.000000	61.000000	61.000000
mean	140.721311	106.327869	34.131148
std	32.517242	26.840095	11.811683
min	68.000000	56.000000	11.000000
25%	113.000000	86.000000	27.000000
50%	146.000000	109.000000	34.000000
75%	166.000000	125.000000	40.000000
max	207.000000	152.000000	68.000000

Seleccionamos los datos del mes de abril y mayo para analizar los datos por mes

Para seleccionar las columnas de un dataframe se usa una sintaxis de rangos por medio de dos puntos :.

[::] Donde:

corresponde al índice a partir del cual se iniciará el rango.

corresponde al índice final del rango, el cual no será incluído en el resultado.

corresponde al tamaño incrementos/decrementos que se aplicará al rango.

en el caso de nuestro ejercicio utilizaremos de la siguiente forma:

In [8]:
```
# seleccionamos los datos del mes de abril 30 registros para calcular estadisticos de
l mes de abril
df_1 = datos[:30]
print ("Estadisticos del mes de abril")
print ("-----------------------------")
df_1[['TOTAL_PACIENTES','ATENDIDOS','NO_ATENDIDOS']].describe()
```

```
Estadisticos del mes de abril
-----------------------------
```

Out[8]:

	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
count	30.000000	30.000000	30.000000
mean	123.533333	93.833333	29.666667
std	28.369159	24.176197	11.905760
min	68.000000	56.000000	11.000000
25%	103.500000	76.000000	21.750000
50%	122.000000	92.000000	30.500000
75%	144.250000	112.500000	35.000000
max	189.000000	149.000000	68.000000

Para el caso del mes de mayo seleccionamos desde el rango de datos del mes

In [9]:
```
df_2 = datos[30:61]
df_2
```

Out[9]:

	DIASEMANA	FECHA	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
30	Lunes	01/05/2017	150	121	29
31	Martes	02/05/2017	159	118	40
32	Miercoles	03/05/2017	180	143	37
33	Jueves	04/05/2017	194	141	53
34	Viernes	05/05/2017	150	109	41
35	Sábado	06/05/2017	148	106	42
36	Domingo	07/05/2017	137	105	32
37	Lunes	08/05/2017	182	140	42
38	Martes	09/05/2017	176	143	32
39	Miercoles	10/05/2017	164	125	38
40	Jueves	11/05/2017	182	142	40
41	Viernes	12/05/2017	182	151	31
42	Sábado	13/05/2017	146	114	32
43	Domingo	14/05/2017	99	71	28
44	Lunes	15/05/2017	188	134	52
45	Martes	16/05/2017	178	137	41
46	Miercoles	17/05/2017	166	121	44
47	Jueves	18/05/2017	161	124	36
48	Viernes	19/05/2017	173	109	64
49	Sábado	20/05/2017	113	71	42
50	Domingo	21/05/2017	94	61	32
51	Lunes	22/05/2017	178	146	32
52	Martes	23/05/2017	207	152	55
53	Miercoles	24/05/2017	177	129	48
54	Jueves	25/05/2017	111	86	25
55	Viernes	26/05/2017	133	95	38
56	Sábado	27/05/2017	137	97	40
57	Domingo	28/05/2017	157	110	47
58	Lunes	29/05/2017	162	119	43
59	Martes	30/05/2017	140	122	18
60	Miercoles	31/05/2017	154	129	18

**Igualmente realizamos de la misma forma para el mes de mayo

In [10]:
```
print ("Estadisticos del mes de Mayo ")
print ("----------------------------")

df_2[['TOTAL_PACIENTES','ATENDIDOS','NO_ATENDIDOS']].describe()
```

```
Estadisticos del mes de Mayo
----------------------------
```

Out[10]:

	TOTAL_PACIENTES	ATENDIDOS	NO_ATENDIDOS
count	31.000000	31.000000	31.000000
mean	157.354839	118.419355	38.451613
std	27.485206	23.845019	10.138503
min	94.000000	61.000000	18.000000
25%	143.000000	107.500000	32.000000
50%	161.000000	121.000000	40.000000
75%	178.000000	138.500000	42.500000
max	207.000000	152.000000	64.000000

OTROS GRÁFICOS ADICIONALES

Gráficos lineales

Nos permite mostrar la tendencia de los datos, para el caso del ejercicio el gráfico muestra la tendencia de los datos de las semanas del mes por cada una de las variables.

In [11]:
```
#x = np.arange(61)

x = datos["DIASEMANA"]
t1 = datos["TOTAL_PACIENTES"]
t2 = datos["ATENDIDOS"]
t3 = datos["NO_ATENDIDOS"]

plt.figure(figsize=(12,8))
plt.plot(x,t1,x,t2,x,t3,marker='o')
plt.xlabel('Dias de Observación')
plt.ylabel('Cantidad de Pacientes')


plt.legend(('TOTAL PACIENTES','ATENDIDOS', 'NO ATENDIDOS'), prop = {'size':10},loc='u
pper right')
```

Out[11]:
```
<matplotlib.legend.Legend at 0x20706209080>
```

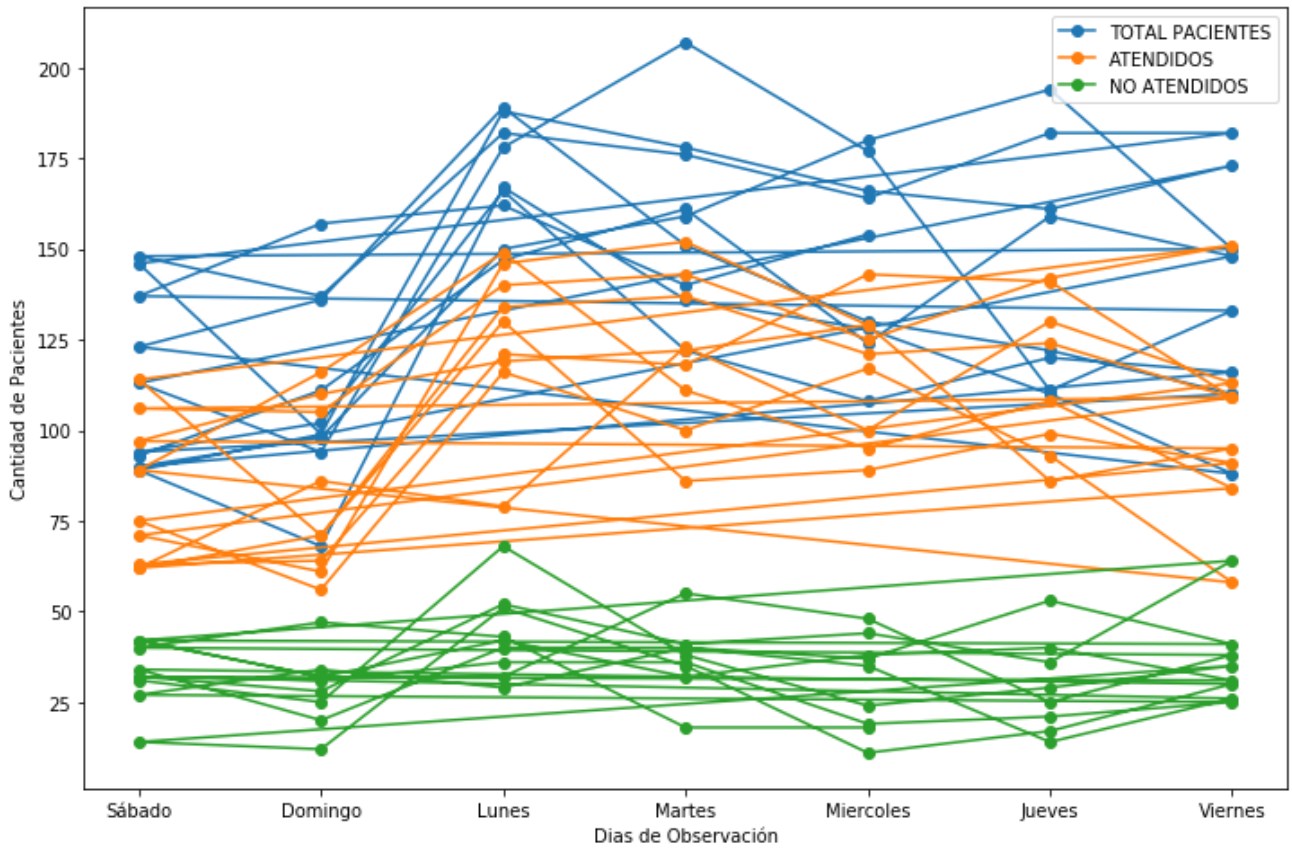


Gráfico en varias ventanas

utilizamos subplot para hacer varios cuadros estadístico separadas de diferentes variables del problema. Esto nos permite tener una mejor apreciación de los datos

In [12]:
```
x = range(61)
plt.figure(figsize=(15,15))
plt.subplot(131)
t1 = datos["TOTAL_PACIENTES"]
p1, = plt.plot(x,t1,'r-')
plt.ylabel('PACIENTES')
plt.title(' PACIENTES ATENDIDOS')


plt.subplot(132)
t2 = datos["ATENDIDOS"]
p1, = plt.plot(x,t2,'b-')

plt.title(' ATENDIDOS')

plt.subplot(133)
t3 = datos["NO_ATENDIDOS"]
p1, = plt.plot(x,t3,'g-')

plt.title(' NO ATENDIDOS')
```

Out[12]: Text(0.5, 1.0, ' NO ATENDIDOS')

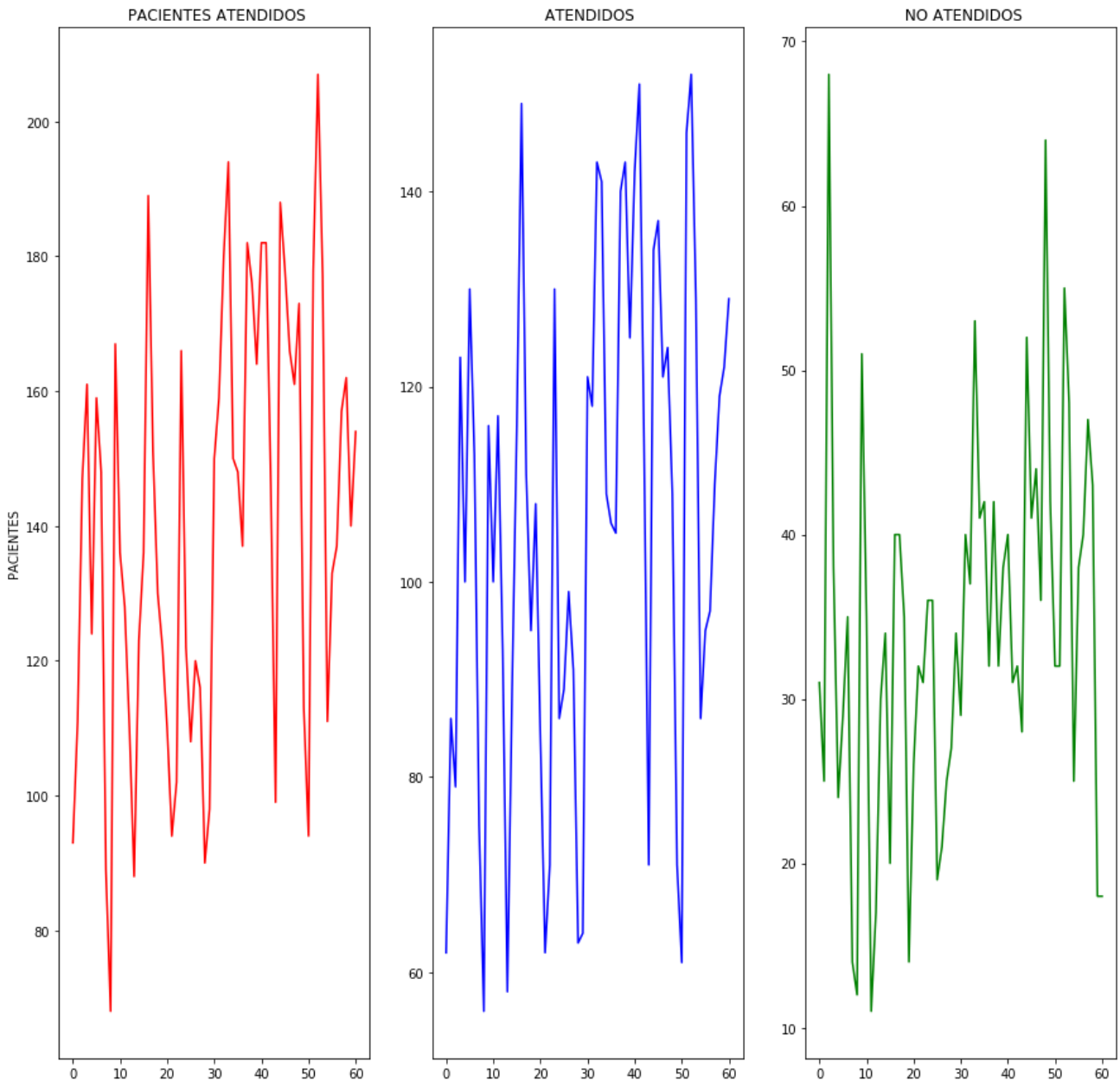


TABLA DE FRECUENCIAS

In [13]:
```
# OBTENER LOS DATOS  UNICOS DE LA TABLA
lis = datos["TOTAL_PACIENTES"].unique()
lis
dfclases=pd.DataFrame(lis,columns=["TOTAL_PACIENTES"])
dfclases
```

Out[13]:

	TOTAL_PACIENTES
0	93
1	111
2	147
3	161
4	124
5	159
6	148
7	89
8	68
9	167
10	136
11	128
12	110
13	88
14	123
15	189
16	151
17	130
18	122
19	94
20	102
21	166
22	108
23	120
24	116
25	90
26	98
27	150
28	180
29	194
30	137
31	182
32	176
33	164
34	146
35	99
36	188
37	178
38	173

	TOTAL_PACIENTES
39	113
40	207
41	177
42	133
43	157
44	162
45	140
46	154

In [18]:

```
#TABLA DE FRECUENCIAS ABSOLUTAS
# OBTENER FRECUENCIAS ABSOLUTAS DE CADA CLASE
datafi=pd.crosstab(index=datos["TOTAL_PACIENTES"], columns = "fi")

# Creamos una lista con los valores de las frecuencias

li = datafi.values
# agregamos una columna al dataframe
dfclases["fi"] = li
#observamos dfclase
dfclases
```

Out[18]:

	TOTAL_PACIENTES	fi
0	93	1
1	111	1
2	147	1
3	161	1
4	124	1
5	159	2
6	148	1
7	89	1
8	68	1
9	167	1
10	136	2
11	128	2
12	110	1
13	88	1
14	123	1
15	189	2
16	151	1
17	130	1
18	122	1
19	94	1
20	102	1
21	166	2
22	108	2
23	120	1
24	116	1
25	90	1
26	98	2
27	150	2
28	180	1
29	194	1
30	137	1
31	182	2
32	176	2
33	164	1
34	146	1
35	99	2
36	188	1
37	178	1
38	173	1

	TOTAL_PACIENTES	fi
39	113	1
40	207	2
41	177	1
42	133	3
43	157	1
44	162	1
45	140	1
46	154	1

In [82]:
```
#total = dfclases.sum(axis=0)
#total
```

Out[82]:
```
TOTAL_PACIENTES    6548
fi                   61
dtype: int64
```

In [83]:
```
total
```

Out[83]:
```
TOTAL_PACIENTES    6548
fi                   61
dtype: int64
```

In [84]:

```
# Columna de Frecuencia relativa

total = dfclases.sum(axis=0)

datahi = dfclases["fi"]/total["fi"]           # aqui calculamos la frecuencia

datahi.values
# agregamos nueva columna de frecuencia relativa
dfclases["hi"] = datahi
dfclases
```

Out[84]:

	TOTAL_PACIENTES	fi	hi
0	93	1	0.016393
1	111	1	0.016393
2	147	1	0.016393
3	161	1	0.016393
4	124	1	0.016393
5	159	2	0.032787
6	148	1	0.016393
7	89	1	0.016393
8	68	1	0.016393
9	167	1	0.016393
10	136	2	0.032787
11	128	2	0.032787
12	110	1	0.016393
13	88	1	0.016393
14	123	1	0.016393
15	189	2	0.032787
16	151	1	0.016393
17	130	1	0.016393
18	122	1	0.016393
19	94	1	0.016393
20	102	1	0.016393
21	166	2	0.032787
22	108	2	0.032787
23	120	1	0.016393
24	116	1	0.016393
25	90	1	0.016393
26	98	2	0.032787
27	150	2	0.032787
28	180	1	0.016393
29	194	1	0.016393
30	137	1	0.016393
31	182	2	0.032787
32	176	2	0.032787
33	164	1	0.016393
34	146	1	0.016393
35	99	2	0.032787
36	188	1	0.016393
37	178	1	0.016393
38	173	1	0.016393

	TOTAL_PACIENTES	fi	hi
39	113	1	0.016393
40	207	2	0.032787
41	177	1	0.016393
42	133	3	0.049180
43	157	1	0.016393
44	162	1	0.016393
45	140	1	0.016393
46	154	1	0.016393

In [85]:
```
total1 =  dfclases.sum(axis=0)  # totales
```

In [86]:
```
total1
```

Out[86]:
```
TOTAL_PACIENTES    6548.0
fi                   61.0
hi                    1.0
dtype: float64
```

In [87]:

```
# la suma de las frecuencias Relativas nos da 1
# aqui vamos a calcular la frecuencia absoluta
FA = dfclases["fi"].values
# obtenemos FA
a=[]
b=0
for c in FA:
    b = c + b
    a.append(b)
dfclases["FA"] = a

HI = dfclases["hi"].values
# obtenemos HI
a=[]
b=0
for c in HI:
    b = c + b
    a.append(b)
dfclases["HI"] = a

dfclases
```

Out[87]:

	TOTAL_PACIENTES	fi	hi	FA	HI
0	93	1	0.016393	1	0.016393
1	111	1	0.016393	2	0.032787
2	147	1	0.016393	3	0.049180
3	161	1	0.016393	4	0.065574
4	124	1	0.016393	5	0.081967
5	159	2	0.032787	7	0.114754
6	148	1	0.016393	8	0.131148
7	89	1	0.016393	9	0.147541
8	68	1	0.016393	10	0.163934
9	167	1	0.016393	11	0.180328
10	136	2	0.032787	13	0.213115
11	128	2	0.032787	15	0.245902
12	110	1	0.016393	16	0.262295
13	88	1	0.016393	17	0.278689
14	123	1	0.016393	18	0.295082
15	189	2	0.032787	20	0.327869
16	151	1	0.016393	21	0.344262
17	130	1	0.016393	22	0.360656
18	122	1	0.016393	23	0.377049
19	94	1	0.016393	24	0.393443
20	102	1	0.016393	25	0.409836
21	166	2	0.032787	27	0.442623
22	108	2	0.032787	29	0.475410
23	120	1	0.016393	30	0.491803
24	116	1	0.016393	31	0.508197
25	90	1	0.016393	32	0.524590
26	98	2	0.032787	34	0.557377
27	150	2	0.032787	36	0.590164
28	180	1	0.016393	37	0.606557
29	194	1	0.016393	38	0.622951
30	137	1	0.016393	39	0.639344
31	182	2	0.032787	41	0.672131
32	176	2	0.032787	43	0.704918
33	164	1	0.016393	44	0.721311
34	146	1	0.016393	45	0.737705
35	99	2	0.032787	47	0.770492
36	188	1	0.016393	48	0.786885
37	178	1	0.016393	49	0.803279
38	173	1	0.016393	50	0.819672

	TOTAL_PACIENTES	fi	hi	FA	HI
39	113	1	0.016393	51	0.836066
40	207	2	0.032787	53	0.868852
41	177	1	0.016393	54	0.885246
42	133	3	0.049180	57	0.934426
43	157	1	0.016393	58	0.950820
44	162	1	0.016393	59	0.967213
45	140	1	0.016393	60	0.983607
46	154	1	0.016393	61	1.000000

In [88]:
```
total1 =  dfclases.sum(axis=0)
```

In [89]:
```
total1
```

Out[89]:
```
TOTAL_PACIENTES    6548.000000
fi                   61.000000
hi                    1.000000
FA                 1427.000000
HI                   23.393443
dtype: float64
```

In [5]:
```
datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61 entries, 0 to 60
Data columns (total 5 columns):
DIASEMANA          61 non-null object
FECHA              61 non-null object
TOTAL_PACIENTES    61 non-null int64
ATENDIDOS          61 non-null int64
NO_ATENDIDOS       61 non-null int64
dtypes: int64(3), object(2)
memory usage: 2.5+ KB
```

In [7]:
```
datos["ATENDIDOS"]+100
```

Out[7]:
```
0     162
1     186
2     179
3     223
4     200
5     230
6     213
7     175
8     156
9     216
10    200
11    217
12    193
13    158
14    189
15    216
16    249
17    211
18    195
19    208
20    184
21    162
22    171
23    230
24    186
25    189
26    199
27    191
28    163
29    164
     ...
31    218
32    243
33    241
34    209
35    206
36    205
37    240
38    243
39    225
40    242
41    251
42    214
43    171
44    234
45    237
46    221
47    224
48    209
49    171
50    161
51    246
52    252
53    229
54    186
55    195
56    197
57    210
58    219
59    222
60    229
Name: ATENDIDOS, Length: 61, dtype: int64
```

In [ ]: