Análisis Inteligente de Datos **2^{do} Semestre 2017**

Taller 1: Visualización y Análisis Descriptivo de Datos con Python

Profesor: Dr. Ing. Rodrigo Salas *Análisis Inteligente de Datos*

1.1 DataFrame de Pandas

Los paquetes de python que serán utilizados son: `, numpy, scypy, matplotlib.pyplot, seaborn, panda`.

```
import numpy as np
import pandas as pd
import scipy.stats as ss
import matplotlib.pyplot as plt
import seaborn as sns
```

Para realizar el estudio descriptivo de los datos utilizaremos la clase `DataFrame` de la biblioteca `pandas`[1]. La clase `DataFrame` corresponde a una tabla bi-dimensional donde cada columna corresponde a un atributo y cada fila a una muestra, el objeto es un contenedor del tipo diccionario. El contrato de la clase `DataFrame` es:

```
class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)¶
```

donde

- **data:** estructura de datos que puede ser un ndarray de numpy (arreglo), diccionario o otro DataFrame.
- **index:** es el índice de las filas utilizado para acceder a los elementos. Por defecto se usa un índice entre 0 y n (`np.arange(n)`)
- **columns:** es el índice de las columnas utilizado para acceder a los atributos. Por defecto se usa un índice entre 0 y n (`np.arange(n)`)
- **dtype:** Utilizada para forzar un tipo de dato.
- **copy:** Copia los datos de la entrada desde las estructuras de ndarray.

Ejemplo:

```
datos_1=np.random.randn(10)
datos_2=np.random.randn(10)
d = {'atributo_1': datos_1, 'atributo_2': datos_2}
df = pd.DataFrame(data=d)
```

[1] `http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html`

Para acceder al atributo 2 se usa: `df['atributo_2']`. Para acceder al elemento con índice 3: `df.loc[3]`.

Algunos de los atributos más utilizados son:

- **loc**: Utilizado para acceder a un elemento.
- **axes**: Nombres de los atributos (columnas) y de los índices (filas)
- **shape**: Número de filas y columans de la tabla
- **size**: Cantidad de elementos contenidos en toda la tabla.
- **T**: Transpuesta de la tabla.
- **index**
- **values**

Algunos de los métodos más utilizados son:

- **describe()**
- **boxplot()**
- **cov([min_periods])**
- **cummax([axis, dtype, out, skipna])**
- **cummin([axis, dtype, out, skipna])**
- **cumprod([axis, dtype, out, skipna])**
- **cumsum([axis, dtype, out, skipna])**
- **groupby([by, axis, level, as_index, sort, ...])**
- **head([n])**
- **hist(data[, column, by, grid, xlabelsize, ...])**
- **kurt([axis, skipna, level, numeric_only])**
- **kurtosis([axis, skipna, level, numeric_only])**
- **mad([axis, skipna, level])**
- **max([axis, skipna, level, numeric_only])**
- **mean([axis, skipna, level, numeric_only])**
- **median([axis, skipna, level, numeric_only])**
- **min([axis, skipna, level, numeric_only])**
- **quantile([q, axis, numeric_only])**
- **sample([n, frac, replace, weights, ...])**
- **skew([axis, skipna, level, numeric_only])**

- **sort_values(by[, axis, ascending, inplace, ...])**
- **sort_index(by[, axis, ascending, inplace, ...])**
- **std([axis, skipna, level, ddof, numeric_only])**
- **sum([axis, skipna, level, numeric_only])**
- **tail([n])**
- **var([axis, skipna, level, ddof, numeric_only])**

Otros métidos que también podrían ser útiles:

- **abs()**
- **append()**
- **copy()**
- **to_csv([path_or_buf, sep, na_rep, ...])**
- **to_dict(*args, **kwargs)**
- **to_excel(excel_writer[, sheet_name, na_rep, ...])**

Ejemplo de manipulación de `DataFrame`

```
df.head()
df.index
df.columns
df.values
df.describe()

df.sort_values(by='atributo_1')

datos_3=np.random.randn(10)
df['atributo_3']=datos_3

df['atributo_2']
df[0:4]
df.loc[0:4]
df.loc[3:7,['atributo_1','atributo_3']]
df[df['atributo_1']>0]
```

1.2 Visualización la distribución de los datos con Seaborn

Se desea tener una noción respecto a la distribución de los datos.

Para realizar el estudio utilizaremos los siguientes paquetes:

```
import numpy as np
import pandas as pd
from scipy import stats, integrate
import matplotlib.pyplot as plt
import seaborn as sns
```

Utilizaremos color para el fondo de los gráficos:

```
sns.set(color_codes=True)
```

Además colocaremos una semilla al generador de números aleatorios:

```
np.random.seed(sum(map(ord, "distributions")))
```

El primer gráfico que estudianremos es el `distplot()`, el cual grafica el histograma y la estimación del kernel de densidad (KDE)

```
# Se generan 100 números aleatorios con distribución normal estándar
x = np.random.normal(size=100)

# Se grafica con el método del paquete de Seaborn
sns.distplot(x);
```

Podemos graficar el histograma (sin el KDE) con el gráfico de puntos (rug plot):

```
sns.distplot(x, kde=False, rug=True);
```

Se puede definir la cantidad de barras (bins):

```
sns.distplot(x, bins=20, kde=False, rug=True);
```

Para graficar sólo el kernel de densidad.

```
sns.distplot(x, hist=False, rug=True);
# o también se puede usar el método kdeplot
sns.kdeplot(x, shade=True);
```

El método aproxima de KDE realiza una aproximación de la función de densidad mediante una promediación de gaussianas centradas en los datos. (Ver [Seaborn] para obtener más información). En el método de estimación de KDE se puede controlar el ancho de las gaussianas con el parámetro de bandwidth `bw`

```
sns.kdeplot(x)
sns.kdeplot(x, bw=.2, label="bw: 0.2")
sns.kdeplot(x, bw=2, label="bw: 2")
plt.legend();
```

Para el gráfico de puntos se utiliza:

```
sns.rugplot(x)
```

1.3 Gráficos Multivariados

Para explorar los gráficos multivariados, generaremos datos provenientes de gaussianas bivariadas:

```
mean, cov = [0, 1], [(1, .5), (.5, 1)]
data = np.random.multivariate_normal(mean, cov, 200)
df = pd.DataFrame(data, columns=["x", "y"])
```

El gráfico de dispersión (scatter-plot) se obtiene usando el paquete de matplotlib:

```
plt.scatter(x="x", y="y", data=df)
```

El gráfico `jointplot` muestra el gráfico de dispersión y los histogramas marginales de cada variable:

```
sns.jointplot(x="x", y="y", data=df)
```

El histograma bivariado puede observarse con el gráfico Hexbin plot:

```
x, y = np.random.multivariate_normal(mean, cov, 1000).T
sns.jointplot(x=x, y=y, kind="hex", color="k");
```

Al utilizar algún tema en el gráfico: `darkgrid, whitegrid, dark, white, ticks`

```
with sns.axes_style("white"):
    sns.jointplot(x=x, y=y, kind="hex", color="k");
```

Estimación del kernel de densidad bivariado (se puede utilizar el método `kdeplot()`) :

```
sns.jointplot(x="x", y="y", data=df, kind="kde");
```

Gráfico de curvas de nivel y de marcas de datos:

```
f, ax = plt.subplots(figsize=(6, 6))
sns.kdeplot(df.x, df.y, ax=ax)
sns.rugplot(df.x, color="g", ax=ax)
sns.rugplot(df.y, vertical=True, ax=ax);
```

```
g = sns.jointplot(x="x", y="y", data=df, kind="kde", color="m")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("$X$", "$Y$");
```

1.4 Visualización de relaciones entre dos variables

Se desea analizar las relaciones entre las variables y para ello de utiliza `pairplot()`

Por ejemplo, utlizaremos el conjunto de datos de la planta de iris.

```
iris = sns.load_dataset("iris")
sns.pairplot(iris);
```

También se puede estudiar la relación de las variables con la estimación de la densidad:

```
g = sns.PairGrid(iris)
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.kdeplot, cmap="Blues_d", n_levels=6);
```

1.5 Gráficos de Box-Plot (caja-bigotes

Gráfico de boxplot horizontal:

```
import numpy as np
import seaborn as sns

sns.set(style="ticks", palette="muted", color_codes=True)

# Cargar el conjunto de datos de los planetas

planetas = sns.load_dataset("planets")

# Graficar el período orbital con cajas horizontales
ax = sns.boxplot(x="distance", y="method", data=planetas, whis=np.inf, color="c")
```

```
#Agregar los puntos en cada observacion
sns.stripplot(x="distance", y="method", data = planetas, jitter=True, ...
   size = 3, color=".3", linewidth=0)

# Colocar el eje en escala logarítmica
ax.set_xscale("log")
sns.despine(trim=True)
```

Gráfico de Boxplot Agrupado

```
import seaborn as sns
sns.set(style="ticks")

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and sex
sns.boxplot(x="day", y="total_bill", hue="sex", data=tips, palette="PRGn")
sns.despine(offset=10, trim=True)
```

1.6 Lectura y escritura de datos

Pandas provee una API de entrada y salida, de forma tal que se puedan leer datos de diferentes tipos de fuentes.

Los métodos más comunes para la lectura de los datos son: `read_csv, read_excel, read_sql, read_json`, por otro lado, los métodos de escritura son: `to_csv, to_excel, to_sql, to_json`

Por ejemplo, para leer un archivo csv

```
pd.read_csv('foo.csv')
```

```
pd.read_csv('foo.csv', index_col=0)
```

```
pd.read_csv('foo.csv', index_col='date')
```

Lectura de un archivo indicando el tipo de separador:

```
data = pd.read_csv('examples/brain_size.csv', sep=';', na_values=".")
```

Lectura desde un archivo Excel

```
xlsx = pd.ExcelFile('path_to_file.xls)
df = pd.read_excel(xlsx, 'Sheet1')
```

1.7 Ejemplo: Análisis de los datos del Tamaño del Cerebro

Utilizaremos los datos de Willerman 1991 respecto al tamaño, peso y coeficiente intelectual de los cerebros,

```
data = pd.read_csv('brain_size.csv', sep=';', na_values=".")
```

Para obtener el promedio de la variable VIQ del género femenino

```
data[data['Gender'] == 'Female']['VIQ'].mean()
```

Realizando el análisis según el género:

```
groupby_gender = data.groupby('Gender')
for gender, value in groupby_gender['VIQ']:
        print((gender, value.mean()))
```

Gráfico de Box-Plot

```
# Box plots of different columns for each gender
groupby_gender = data.groupby('Gender')
groupby_gender.boxplot(column=['FSIQ', 'VIQ', 'PIQ'])
```

Gráfico de dispersión:

```
pd.tools.plotting.scatter_matrix(data[['Weight', 'Height', 'MRI_Count']])
pd.tools.plotting.scatter_matrix(data[['PIQ', 'VIQ', 'FSIQ']])
```

1.8 Tarea

Realizar un estudio estadístico descriptivo para uno de los siguientes conjuntos de datos (escoger solo uno):

- Bank Marketing Data Set: `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`
- Student Performance Data Set: `https://archive.ics.uci.edu/ml/datasets/Student+Performance`
- Census Income Data Set: `https://archive.ics.uci.edu/ml/datasets/Census+Income`

Para la presente tarea se deberá realizar primer utilizando el software Python y posteriormente repetir el estudio con cualquiera de los siguientes: Qlik [Qlik] o Tableau [Tableau].

1. Realizar un estudio de estadística descriptiva a dos variables numéricas. Hacer una tabla con los resultados.

2. Visualizar cada variable con por lo menos 3 gráficos descriptivos.
3. Separar los datos según el *target* y repetir el estudio estadístico descriptivo y los gráficos.
4. Para el caso de Qlik o Tablea agregar un par de gráficos adicionales, realizar un dashboard, vincular los gráficos y colocar filtros.
5. En cada problema deberá discutir y explicar los resultados obtenidos. Interpretar las gráficas resultantes.

Consideraciones:

- Deberá entregar un informe en formato doc o pdf.
- Además deberá hacer entrega de los códigos generados para el estudio.
- Deberá compartir en la nube el enlace de Qlik o Tableau según corresponda.
- Fecha de entrega: 29 de Septiembre del 2017 a las 23:55 utilizando la plataforma moodle
- Cada día de atraso será penalizado con 20 puntos.
- Pueden discutir los problemas con sus compañeros, sin embargo la entrega y el trabajo es de grupos de 2 a 3 estudiantes máximo.

References

[1] ScyPy.org , *Statistical Functions (scipy.stats)*. `http://docs.scipy.org/doc/scipy/reference/stats.html`

[2] Pandas, *Python Data Analysis Library*. `http://pandas.pydata.org/`

[3] Seaborn, *Seaborn: statistical data visualization*. `https://stanford.edu/~mwaskom/software/seaborn/`

[4] Numpy, *Fundamental package for scientific computing with Python*. `http://www.numpy.org/`

[5] Matplotlib, *Biblioteca gráfica para python*. `http://matplotlib.org`

[6] Scipy Lecture Notes `http://www.scipy-lectures.org`

[7] Tableau `https://www.tableau.com`

[8] Qlik `https://www.qlik.com`