Data Wrangling

Transformación de datos

Carlos Andrés Pérez Guzmán

Capítulo 1: Filtrado y selección

Carga de librarías

```
import pandas as pd
```

Carga de datos

```
df = pd.read_csv("nycflights.csv")
#df.info()
df.head()
```

Filtro del valor de una columna

```
df1 = df[df["origin"]=="LGA"]
df1.head()
```

```
df1 = df[df["origin"]!="LGA"]
df1.head()
```

El caracter "&" equivale al conector lógico "**AND**"

```
df1 = df[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50)]
df1.head()
```

El carater "|" equivale al conector lógico "**OR**"

```
df1 = df[(df["month"]==5) | (df["month"]==12)]
df1.head()
```

Es posible usar otras variables para filtrar datos

```
month_var = 5
df1 = df[(df["month"] == month_var) & (df["air_time"]>50)]
df1.head()
```

Se puede negar cualquier criterio de filtro con "~"

```
month_var = 5
df1 = df[~((df["month"] == month_var) & (df["air_time"]>50))]
df1.head()
```

Filtro a través del método loc

```
df1 = df.loc[df["origin"]=="LGA"]
df1.head()
```

```
df1 = df.loc[df["origin"]!="LGA"]
df1.head()
```

El caracter "&" equivale al conector lógico "**AND**"

```
# & equivale a "AND"

df1 = df.loc[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50)]
df1.head()
```

El carater "|" equivale al conector lógico "**OR**"

```
df1 = df.loc[(df["month"]==5) | (df["month"]==12)]
df1.head()
```

Es posible usar otras variables para filtrar datos

```
month_var = 5
df1 = df.loc[(df["month"] == month_var) & (df["air_time"]>50)]
df1.head()
```

Se puede negar cualquier criterio de filtro con "~"

```
month_var = 5
df1 = df.loc[~((df["month"] == month_var) & (df["air_time"]>50))]
df1.head()
```

Filtro por posición de filas y columnas

```
df.iloc[:4] #First 4 rows, all columns
df.iloc[1:5,] #Second to fifth row
df.iloc[5,0] #Sixth row and first column
df.iloc[1:5,0] #Second to Fifth row, first column
df.iloc[1:5,:5] #Second to Fifth row, first 5 columns
df.iloc[2:7,1:3] #Third to Seventh row, 2nd and 3rd column
```

Filtro por posición de filas y nombre de columnas

```
# Selección de 5 filas y columnas con nombre "origin" y "distance"
df1 = df.loc[df.index[10:16],["origin","distance"]]
df1.head()
```

```
# Filtro múltiple
df1 = df.loc[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50),
             ["origin","distance","year"]]
df1.head()
```

Filtro para seleccionar múltiples valores

```
df1 = df[df["origin"].isin(["JFK", "LGA"])]
df1.head()
```

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin
0	2013	6	30	940	15	1216	-4	VX	N626VA	407	JFK
1	2013	5	7	1657	-3	2104	10	DL	N3760C	329	JFK
2	2013	12	8	859	-1	1238	11	DL	N712TW	422	JFK
3	2013	5	14	1841	-4	2122	-34	DL	N914DL	2391	JFK
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	3652	LGA

Los valores que se deben conservar pueden ser almacenados en una lista

```
variables = ["JFK","LGA"]
df1 = df[df["origin"].isin(variables)]
df1.head()
```

Los valores que **no** se deben conservar pueden ser almacenados en una lista

```
remove_var = ["JFK","LGA"]
df1 = df[~df["origin"].isin(remove_var)]
df1.head()
```

Filtar columnas

```
# Selección de columna: 'flight' hasta 'dest' para todas las filas
df1 = df.loc[:,"flight":"dest"]
df1.head()
```

```
# Selección de columna: 'flight' y 'dest'
df1 = df.loc[:,["flight","dest"]]
df1.head()
```

```
# Selección de columna: 'flight' y 'dest'
df1 = df[["flight","dest"]]
df1.head()
```

```
# Selección de columnas diferentes de 'flight', 'dest' y 'year'
# También es posible eliminar filas si fuese necesario
df1 =df.drop(['flight','dest','year'], axis = 1)
df1.head()
```

Filtar valores duplicados

Devuelve aquellos elementos que tienen valores duplicados. Así, la tabla resultante contendrá valores únicos según los parámetros de la función.

```
# keep: If 'first', it considers first value as unique and rest of the same values as dupl
# keep: If 'last', it considers last value as unique and rest of the same values as duplic
# keep: If False, it consider all of the same values as duplicates
# inplace: Boolean values, removes rows with duplicates if True.
# ignore_indexbool, default False. If True, the resulting axis will be labeled 0, 1, …, n
```

```
df1 = df.drop_duplicates(subset = "origin",
                         keep = 'first',
                         inplace = False,
                         ignore_index = False)

df1.head()
```

Se pueden buscar valores duplicados en una combinación de varias columnas

```
# keep: If 'first', it considers first value as unique and rest of the same values as dupl
# keep: If 'last', it considers last value as unique and rest of the same values as duplic
# keep: If False, it consider all of the same values as duplicates
# inplace: Boolean values, removes rows with duplicates if True.
# ignore_indexbool, default False. If True, the resulting axis will be labeled 0, 1, …, n

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', 'Alejandro', 'Santiago'],
                   'Age'        : [23, 24, 24, 25],
                   'University' : ['AA', 'BB', 'BB', 'DD']})

df1 = df.drop_duplicates(subset = ["Age","University"],
                         keep = False,
                         inplace = False,
                         ignore_index = True)

df1.head()
```

Conservar **solamente** los valores duplicados

```
# keep{‘first’, ‘last’, False}, default ‘first’
# Determines which duplicates (if any) to mark.
# first : Mark duplicates as True except for the first occurrence.
# last : Mark duplicates as True except for the last occurrence.
# False : Mark all duplicates as True.

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', 'Alejandro', 'Santiago'],
                   'Age'        : [23, 24, 24, 25],
                   'University' : ['AA', 'BB', 'BB', 'BB']})

df1 = df[df[["Age","University"]].duplicated(keep = False)]
df1.head()
```

Filtrar valores faltantes

```
# La librería "numpy" proporciona una forma para generar un valor vacío
import numpy as np

# axis
# axis{0 or ‘index’, 1 or ‘columns’}, default 0
# Determine if rows or columns which contain missing values are removed.
# 0, or ‘index’ : Drop rows which contain missing values.
# 1, or ‘columns’ : Drop columns which contain missing value.

# how
# how: {‘any’, ‘all’}, default ‘any’
# Determine if row or column is removed from DataFrame, when we have at least one NA or al
# ‘any’ : If any NA values are present, drop that row or column.
# ‘all’ : If all values are NA, drop that row or column.

# subset
# subset: column label or sequence of labels, optional
# Labels along other axis to consider, e.g. if you are dropping rows these would be a list

# inplace
# inplace: bool, default False
# If True, do operation inplace and return None.

# La función dropna y notnull pueden identificar a los valores de np.nan, None y pd.NA

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Santiago', "Fernand0", "Ma
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})
#df1 = df.dropna(axis = 0, how = "all")
df1 = df.dropna(axis = 0, how = "any")
df1.head(10)
```

Cuenta de los valores no nulos en el data frame

```
column_notnull = df.notnull().sum()
column_notnull
```

Filtrar filas según una cadena de texto

```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina', "Fernand0", "Mar
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuya primera letra es 'C'
df1 = df[df['Name'].str[0] == 'C' ]
df1.head()
```

```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina', "Fernand0", "Mar
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos primeras letras es 'Ca'
df1 = df[df['Name'].str[0:2] == 'Ca' ]
df1.head()
```

```
df['Name'].str[0:5]
```

```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina', "Fernand0", "Mar
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos primeras letras es 'Ca'
df1 = df[df['Name'].str.startswith("Ca") == True]
df1.head()
```

```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina', "Fernand0", "Mar
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos últimas letras es 'os'
df1 = df[df['Name'].str.endswith("os") == True]
df1.head()
```

```
x = df["Name"]
x.str.startswith("Ca")
```