**CONTENIDO:**

Sesiones 1 y, 2. Introducción a Python y Spyder.

Sesión 3. Describiendo nuestros datos con representaciones gráficas y numéricas.

Sesión 4. Sub-setting.

Temas clave: pandas, sub-setting, 99's, special codes in quantitative variables.

Sesión 5. Recoding: Gestionando lo inesperado: Limpieza y formato en nuestros datos.

**PARTE 1 Y 2**

**Nuestro primer dataset**

**import pandas as pd**

**#Define variables.**

**name = ['Bianca','Pedro','Alberto']**

**gender =['Female','Male','Male']**

**age = [20, 35, 46]**

**#Create a dataframe**

**class2019 = pd.DataFrame({'name': name, 'gender’: gender, 'age':age})**

**class2019.shape class2019.head()**

**#QC OK**

**#Clean up**

**del (name, gender, age)**

**#Export dataframe to Excel**

**class2019.to\_excel("class2019.xlsx”)**

**Importar un dataset**

**import os**

**import pandas as pd**

**# Change working directory**

**os.chdir('C:\carp\_alb\EDEM\PEP\code\_and\_data')**

**os.getcwd()**

**#Reads data from CSV file and stores it in a dataframe called rentals\_2011**

**#Pay atention to the specific format of your CSV data (; , or , .)**

**rentals\_2011 = pd.read\_csv ("washington\_bike\_rentals\_2011.csv", sep=';', decimal=',')**

**rentals\_2011.shape**

**rentals\_2011.head()**

**#QC OK**

**How to merge two dataframes**

**# Merge the two dataframes(rentals & weather) into a new single dataframe**

**rentals\_weather\_2011 =pd.merge(weather\_2011, rentals\_2011, on="day")**

**rentals\_weather\_2011.shape rentals\_weather\_2011.head()**

**How we can merge two dataframes in a new one containing same variables but more rows.**

**rentals\_weather\_11\_12 = rentals\_weather\_2011.append(rentals\_weather\_2012, ignore\_index=True)**

Fijarse, el añadir se hace con ‘append’.

**PARTE 3**

**DESCRIBING NOMINAL VARIABLES (I)**

**FROM FREQUENCIES TO A TABLE OF PERCENTAGES**

**# a) obtain n**

**n=mytable.sum()**

**# b) divide by n in order to get proportions,and multiply by 100**

**mytable = (mytable/n)\*100**

**# Round to your pleasure**

**mytable3 = round(mytable2,1)**

**print (mytable3)**

**DESCRIBING NOMINAL VARIABLES (II)**

**BARCHART**

**#Barchart**

**bar\_list = ['Sunny', 'Cloudy', 'Rainy’]**

**plt.bar(bar\_list, mytable2,edgecolor='black’)**

**plt.ylabel('Percentage’)**

**plt.title('Figure 1. Percentage of weather situations')**

**#############################################################################**#Extra tip: Legend with sample size

#You need to have the sample size stored into n

props = dict(boxstyle='round', facecolor='white', lw=0.5)

textstr = '$\mathrm{n}=%.0f$'%(n) plt.text (2,60, textstr, bbox=props) **#############################################################################**

**DESCRIBING DESCRIPTIVE VARIABLES (I)**

**#Histogram Figure 1**

**plt.hist(x, bins=10, edgecolor='black’)**

**plt.xticks(np.arange(0, 10000, step=1000))**

**plt.title('Figure 1. Daily Bicycle rentals in Washington DC''\n' 'by Capital bikeshare.2011 - 2012’)**

**plt.ylabel('Frecuency’) plt.xlabel('Number of rented bicycles')**

**DESCRIBING DESCRIPTIVE VARIABLES (II)**

**#Histogram Figure 2**

**plt.hist(x, bins=10,edgecolor='black’)**

**plt.xticks(np.arange(0, 10000, step=1000))**

**plt.title('Figure 1. Daily Bicycle rentals in Washington DC''\n' 'by Capital bikeshare.2011 - 2012’)**

**plt.ylabel('Frecuency’) plt.xlabel('Number of rented bicycles’)**

**textstr = 'Mean = 4504\nS.D.= 1937 \nn = 731’**

**plt.text (6500,110, textstr)**

**# Add reference lines and store their names in label for later legend**

**plt.axvline(x=4504,linewidth=1, linestyle= 'solid', color="red", label='Mean')**

**DESCRIBING DESCRIPTIVE VARIABLES (III)**

**#histogram ver3**

**plt.hist(x, bins=10, edgecolor='black')**

**plt.xticks(np.arange(0, 10000, step=1000))**

**plt.title('Figure 3. Daily Bicycle rentals in Washington DC' '\n''by Capital bikeshare. 2011 - 2012')**

**plt.ylabel('Frecuency')**

**plt.xlabel('Number of rented bicycles’)**

**props = dict(boxstyle='round', facecolor='white’,lw=0.5)**

**textstr = '$\mathrm{Mean}=%.1f$\n$\mathrm{S.D.}=%.1f$\n$\mathrm{n}=%.0f$'%(m, sd, n) plt.text (6500,110, textstr , bbox=props)**

**plt.axvline(x=m, linewidth=1, linestyle= 'solid', color="red", label='Mean')**

**DESCRIBING DESCRIPTIVE VARIABLES (IV)**

**#histogram ver4**

**plt.hist(x, bins=10, edgecolor='black')**

**plt.xticks(np.arange(0, 10000, step=1000))**

**plt.title('Figure 1. Daily Bicycle rentals in Washington DC' '\n' 'by Capital bikeshare. 2011 - 2012')**

**plt.ylabel('Frecuency')**

**plt.xlabel('Number of rented bicycles’)**

**props = dict(boxstyle='round', facecolor='white', lw=0.5)**

**textstr = '$\mathrm{n}=%.0f$'%(n) plt.text (-50,128, textstr , bbox=props)**

**# Add reference lines and store their names in label for later legend**

**plt.axvline(x=m, linewidth=1, linestyle= 'solid', color="red", label='Mean')**

**plt.axvline(x=m-sd, linewidth=1, linestyle= 'dashed', color="green", label='- 1 S.D.')**

**plt.axvline(x=m + sd, linewidth=1, linestyle= 'dashed', color="green", label='+ 1 S.D.')**

**plt. legend(loc='upper left', bbox\_to\_anchor=(0.73, 0.98))**

**DESCRIBING DESCRIPTIVE VARIABLES (V)**

**#Boxplot**

**plt.boxplot(x,patch\_artist=True, vert=False, labels=['# rentals'])**

**plt.xticks(np.arange(0, 10000, step=1000))**

**plt.show()**

**PARTE 4**

**SUBSETTING (I)**

**# Subset year 0**

**wbr\_2011 = wbr[wbr.yr == 0]**

**# Subset year 1**

**wbr\_2012 = wbr[wbr.yr == 1]**

**SUBSETTING (II) – SELECTING VARIABLES**

**# Select variables, by column name**

**my\_vars=['temp\_celsius','cnt']**

**#Extract those varibles and save them into wbr\_minimal**

**wbr\_minimal= wbr[my\_vars]**

**wbr\_minimal.shape**

**SUBSETTING (I) – SELECTING CASES**

**# Select a subsample from our data**

**# Select cases only from 2011**

**# Create a new dataframe containing observations from 2011**

**#Explore years**

**mytable = wbr.groupby(['yr']).size()**

**print(mytable)**

**#Excursus to Operators**

**# Subset year 0**

**wbr\_2011 = wbr[wbr.yr == 0]**

**# Subset year 1**

**wbr\_2012 = wbr[wbr.yr == 1]**

**AVOIDING ARTEFACTS**

**# Remove cases with nan in any variable**

**# Create a new data frame where observations containing nan are removed**

**wbr\_ue2 = wbr\_ue.dropna()**

**PARTE 5**

**COMPUTING NEW COLUMNS**

**#Computing new columns**

**#Let’s compute the casual to registered rentals ratio**

**wbr['cs\_ratio']=(wbr.casual)/(wbr.registered)**

**wbr.cs\_ratio.describe()**

**#Note that for creation of new columns we use “robust” column specification with [“”] not attribute (.)call**

Esto último quiere decir que la otra forma de llamar variables, que sería en este caso wbr.cs\_ratio, no funcionaría para computar nuevas columnas, así que solo vale wbr[‘cs\_ratio’].

**RECODING (I)**

**#Recoding season into a string variable (season\_cat)**

**wbr.loc[(wbr['season']==1),"season\_cat"]= "Winter"**

**wbr.loc[(wbr['season']==2),"season\_cat"]= "Spring"**

**wbr.loc[(wbr['season']==3),"season\_cat"]= "Summer"**

**wbr.loc[ wbr['season']==4),"season\_cat"]= "Autum"**

**RECODING (II)**

**#Recode the number of rentals in Three Groups**

**wbr.loc[ (wbr['cnt']<2567.1) ,"cnt\_cat2"]= "1: Low rentals"**

**wbr.loc[ ((wbr['cnt']>2567.1) & (wbr['cnt']<6441.6)) ,"cnt\_cat2"]= "2: Average rentals"**

**wbr.loc[ (wbr['cnt']>6441.6) ,"cnt\_cat2"]= "3: High rentals"**

**Show up the plot in your desired order**

**#Import specific functionality from pandas.api.types**

**import CategoricalDtype**

**#Step 1: declare the ordered categories**

**my\_categories=["Low rentals", "Average rentals", "High rentals"]**

**#Step 2: define new data type**

**my\_rentals\_type = CategoricalDtype(categories=my\_categories, ordered=True)**

**#Then create a new categorical\_ordered variable using our specific data type wbr["cnt\_cat5"] = wbr.cnt\_cat2.astype(my\_rentals\_type)**

**PARTE 6**

**DESCRIBING QUANTITATIVE VARIABLES**

Why some days are rent more bikes than other days in Washington D.C.?

**H0.: µ rentals in working days = µ rentals in holidays**

**H1.: µ rentals in working days ≠ µ rentals in holidays**

1. Numeric Procedure ⇨ t test for independent simples
2. Graphic procedure ⇨ confidence interval plot

**Mean comparison (2 groups)**

1. Describe the two variables involved in the hypothesis
2. Perform the numeric test: t.test
3. Perform the graphic test: plot of the means
4. When posible: combine both numeric and graphic in same plot

**Step 1**

**Rentals**

**wbr.cnt.describe() plt.hist(wbr.cnt)**

**Working days**

**mytable = pd.crosstab(index=wbr["wd\_cat"], columns="count")**

**n=mytable.sum()**

**mytable2 = (mytable/n)\*100**

**plt.bar(mytable2.index, mytable2['count'])**

**Step 2**

**#Descriptive comparison:**

**wbr.groupby('wd\_cat').cnt.mean()**

**#Statistical comparison:**

**#Extract the two sub samples and store them in two objects**

**cnt\_wd=wbr.loc[wbr.wd\_cat=='Yes', "cnt"]**

**cnt\_nwd=wbr.loc[wbr.wd\_cat=='No', "cnt"]**

**#Perform a t test for mean comparison**

**#import scipy.stats as stats**

**stats.ttest\_ind(cnt\_wd, cnt\_nwd, equal\_var = False)**

**Step 3**

**#CI meanplot**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**plt.figure(figsize=(5,5))**

**ax = sns.pointplot(x="wd\_cat", y="cnt", data=wbr,ci=95, join=0)**

**plt.yticks(np.arange(3000, 7000, step=500))**

**plt.ylim(2800,6200)**

**plt.axhline(y=wbr.cnt.mean(), linewidth=1, linestyle= 'dashed', color="green")**

**props = dict(boxstyle='round’, facecolor='white', lw=0.5)**

**plt.text(0.85,5400,'Mean:4504.3''\n''n:731' '\n' 't:1.601' '\n' 'Pval.:0.110',** **bbox=props)**

**plt.xlabel('Working Day')**

**plt.title('Figure 6. Average rentals by Working Day.''\n')**

**DESCRIBING QUANTITATIVE VARIABLES**

**Mean comparison (>2 groups)**

**H0.: µ rentals sunny = µ rentals cloudy= µ rentals stormy.**

**H1.: µ rentals differ in at least 2 of the 3 groups compared**

1. Numeric Procedure ⇨ One-Way ANOVA
2. Graphic procedure ⇨ Confidence interval plot

**Mean comparison (>2 groups)**

1. Describe the two variables involved in the hypothesis
2. Perform the numeric test: One-Way ANOVA
3. Perform the graphic test: plot of the means
4. When posible: combine both numeric and graphic in same plot.

**Step 1 it’s easy, we skip it**

**Step 2**

**#Descriptive comparison**

**wbr.groupby('ws\_cat').cnt.mean()**

**#Statistical comparison**

**cnt\_sunny=wbr.loc[wbr.ws\_cat=='Sunny', "cnt"]**

**cnt\_cloudy=wbr.loc[wbr.ws\_cat=='Cloudy', "cnt"]**

**cnt\_rainy=wbr.loc[wbr.ws\_cat=='Rainy', "cnt"]**

**stats.f\_oneway(cnt\_sunny, cnt\_cloudy,cnt\_rainy )**

**Step 3**

**#Graphic comparison: confidence intervals for the means**

**plt.figure(figsize=(5,5))**

**ax = sns.pointplot(x="ws\_cat", y="cnt", data=wbr, capsize=0.05, ci=99.9, join=0)**

**ax.set\_ylabel('')**

**plt.yticks(np.arange(1000, 7000, step=500))**

**plt.ylim(800,6200)**

**plt.axhline(y=wbr.cnt.mean(), linewidth=1, linestyle= 'dashed', color="green")**

**props = dict(boxstyle='round', facecolor='white', lw=0.5)**

**plt.text(1.5, 5000, 'Mean: 4504.3''\n''n: 731' '\n' 'F: 40.06' '\n' 'Pval.: 0.000', bbox=props)**

**plt.xlabel('Weather Situation')**

**plt.title('Figure 8. Average rentals by Weather Situation.''\n')**