Cheatography

Data Science Cheat Sheet

by elhamsh via cheatography.com/31327/cs/13764/

Pandas

import pandas as pd	
df.iloc[:5,:]	return slice of data:all columns first 5 rows
type(df)	DataFrame
df.shape	(len, #ofcols)
df.columns	name of cols
df.index	return index column
df.head(3)	return first 3 rows
df.iloc[-5:,:]	return last 5 rows
df.tail()	return last 5 rows
df.info()	return index, column types, # of row, # of not null cols
type(df['-low'])	Series
type(df['-low'].values)	numpy.ndarray
np.log10(-df['low'])	return data frame
np.log10(-df['low'].va-lues)	return list of list

Each column in pandas is a Series.
You can run numpy on df or a col of df

Statistical Data Analysis

df.des-cribe()	count, mean,std,max, quartiles for each col of non-null rows
df['lo-w'].count()	return # of not null rows
df[cols].c-ount()	return a series
df['lo-w'].mean()	return mean ignoring nulls
df.std()	
df.median()	

Statistical Data Analysis (cont)

df.quantile(q)	q=.5:median q=[.25,.7-5]:IQrange
df['low'].min()	alphabetic order for non-numerics
df['low'].max()	alphabetic order for non-numerics
df.mean(axis='c-olumns')	mean of all columns for each row
df.low	df['low']

Time series

index_col='Date', parse_date=True	
df.loc['2-015-2']	return for all days
df.loc['2-015-2-20']	return all rows with this date
df.loc['2-015-2-20':'2015-3']	range
newD = pd.to_dat-eti-me('Date')	y-m-d h:m:s
df.reinde-x(newD)	reindexing with matching dates. if doesn't match,new rows w. null value
df.reinde-x(n-ewD,me-thod='ffill')	fill empty values forward fill:value of previous rows
method='b-fill'	backward fill: value of later rows
df.resamp-le('D').m-ean()	daily mean
'H', 'min', '2W'	hour, minute, 2 weekshour, minute, 2 weeks
'Y', 'Q', 'M', 'B'	year, quarter, month, business day

Time series (cont)

df.resamp-le('W').s-um().max()	max of weekly sum
df.resamp-le('4h').ffill()	every 4hours. fill nan w. previous valuesevery 4hours. fill nan w. previous values
df1+df2	
df['Temperat-ure']['2010--august']	select temp col of aug.
df['Temperat-ure']['2010-2']	select temp col of feb.
unsmooth.rol-ling(window=-24).mean()	moving average 24h
df['type'].str.u-pper()	return a column converted to uppercase
df['product'].s-tr.contai-ns('ware')	return boolean if substring 'ware' exists
True+True	2
False + False	0
df['product'].s-tr.contai-ns('wa-re').sum()	# of rows contains substring 'ware'
df['date'].d-t.hour	return hour of each row 0-23
df['date'].dt.t-z_localize('-US/Central')	set timezone
df['date'].dt.tz_convert('US/Eastern')	
df['date'].resa-mple('A').first()	yearly from the initial date in data (1960-12-31)
df['date'].resa-mple('A').first-().interpola-te('linear')	replace nan with interp-olation

By **elhamsh**
cheatography.com/elhamsh/

Not published yet.
Last updated 20th December, 2017.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Time series (cont)

df.columns.str.strip()	removes space from df.columns
df.set_index('Date', inplace=True)	
newD = pd.to_datetime('Date_list', format='%Y-%M-%D %H:%M)	
pd.Series(Columns_list, index=newD)	Construct a pandas Series c
ts2_interp = ts2.reindex(ts1.index).interpolate(how='linear')	Reset the index of ts2 to ts1, and then use linear interpolation to fill in the NaNs: ts2_interp
timezone.dt.tz_localize('US/Central')	localize the local time timezone to 'US/Central

Build DF

df=pd.read_csv("filepath", index_col=0)	add index column 0-len(inp)
index_col='nameofacolumn'	
df.index=['A', 'B', ...]	assign index to df. len(index)==len(df)
pd.DataFrame({'id':[1,2,3], 'gen':'M'})	key: columns, values: row
pd.DataFrame(dict_of_lists)	
zipped=list(zip(list_labels, list_values))	
pd.DataFrame(dict(zipped))	list_labels, list_values = list of list
pd.read_csv("filepath", header=None)	no header
pd.read_csv("filepath", options)	col_n:list of column names
hearder=0, names=col_n	rename the header

Build DF (cont)

header=None, names=col_n	no header in file & header is col_n
na_values='-1'	convert specific value (-1) to a nan
na_values={'colname':['-1', '']}	define a dic for each col
parse_dates=[[0,1,1]]	convert 3 columns of date to one col
parse_dates=True	convert column with date to dateformat
delimiter=' '	
header=3	header is in index 3
comment='#'	ignore all lines start with '#' in the input
index_col = 'dates'	set a column as index
df[cols]	take specific columns
df.to_csv('outputpath')	
df.to_excel('outputpath')	
pd.DataFrame({'smoothed':smoothed, 'unsmoothed':unsmoothed})	create df.if they have index, will merge based on index

categorical

df['type'].decribe()	count not null,# of unique,top item,freq. of top
df['type'].unique()	#of unique items
df.loc[df['type']==x,:]	df[df['type']==x]
del def['type']	delete a column

Numpy+Df

df.values	Create array of DataFrame values
df[colname]=0	create a columns with zero elements in df

Cleanning

df_dropped = df.drop(list_to_drop, axis='columns')	Remove the appropriate columns list_to_drop
df.set_index(colname)	Set colname as the index
pd.to_numeric()	It converts a Series of values to floating-point values. Furthermore, by specifying the keyword argument errors='coerce', you can force strings like 'M' to be interpreted as NaN.
df.reset_index()[colname]	Extract the colname column from df using .reset_index()
df.loc[df[colname]=='sth']	choose the rows in df for df[colname]='sth'
df.loc[df[colname].str.contain('sth')]	choose the rows in df where the column df[colname] contain 'sth'

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Plot

import matplotlib.pyplot as plt	
plt.plot(df['low'].values)	x axis= index of value
plt.show()	show the image
plt.plot(df['low'])	x axis is index of df (eg date)
df['low'].plot()	plot series directly. has also x label
df.plot()	show all columns in df with legend
plt.yscale('log')	log scale on vertical axis
df['low'].plot(color='b',style='.-', legend=True)	
plt.axis((minx, maxx,miny,maxy))	zoom
plt.title('title')	
plt.ylabel('label)	
plt.xlabel('xlabel')	
plt.savefig('a.pdf')	
plt.savefig('a.jpg')	
df.plot(subplots=True)	Draw each column in one subplot.
df.plot(x='colname',y='colname',kind='scatter')	plot 2 columns
kind = 'box'	box plot
kind = 'hist'	histogram
kind='area'	
bins=30	integer:#of bins
range=(4,8)	tuple (min,max)
normed=True	boolean. normalize to one for hist
cumulative=True	boolean for hist

Plot (cont)

alpha=0.3	visibility of several histograms
s=sizes	sizes= array of size of each circle in scatter plot
fig, axes=subplots(nrows=1,ncols=1)	
df['low'].plot(ax=axes[0], ...)	...: kind, bins, normed,cumulative
df.plot(y='colname',kind='box')	
style='k.-'	color,marker,line type
plt.clf()	clears the entire current figure with all its axes, but leaves the window opened, such that it may be reused for other plots

Indexing

df['colname']['rowname']	rowname is index_col
df.colname['rowname']	
df.loc['rowname','colname']	
df.loc['rownstart','rownend',:]	row names are inclusive.
df[['low']]	returns a single column data frame
df['low']	returns a series with index of df

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com