oscillator_source_code

November 20, 2017

```
In [1]: from __future__ import division
        import numpy as np

        def MA_create(df):
            MAlist = [[10,'10-MA','Adj Close'],[20,'20-MA','Adj Close'],
                      [50,'50-MA','Adj Close'],[100,'100-MA','Adj Close'],
                      [250,'250-MA','Adj Close']]
            for days,name,base in MAlist:
                df[name] = 'NaN'
                for i in range(len(df)):
                    df.ix[i,name] = MA_cal(df,days,base,i)
            return df

        def MA_cal(df,days,base,i):
            if i >= days-1: return (sum(df.ix[i-(days-1):i+1,base]) / days)
            elif 0 < i < days-1 : return (sum(df.ix[:i+1,base]) / (i+1))
            elif i == 0:  return (df.ix[0,base])

        def MA_update(df):
            MAlist = [[10,'10-MA','Adj Close'],[20,'20-MA','Adj Close'],
                      [50,'50-MA','Adj Close'],[100,'100-MA','Adj Close'],
                      [250,'250-MA','Adj Close']]
            for days,name,base in MAlist:
                for i in range(len(df)+(t+1),len(df)):
                    df.ix[i,name] = MA_cal(df,days,base,i)
            return df
        ######################################################################################
        def BB_create(df):
            df['BW'] = 'NaN'
            df['%b'] = 'NaN'
            for i in range(len(df)):
                df.ix[i,'BW'],df.ix[i,'%b'] = BB_cal(df,i)
            return df

        def BB_cal(df,i):
            if i >= 19: std = np.std(list(df.ix[i-19:i+1,'Adj Close']))
            elif 0 < i < 19: std = np.std(list(df.ix[0:i,'Adj Close']))
            elif i == 0: std = 0
```

```
    upper = df.ix[i,'20-MA'] + 2*std
    lower = df.ix[i,'20-MA'] - 2*std
    BW = ((upper - lower) / df.ix[i,'20-MA']) * 100
    try:
        b = (df.ix[i,'Adj Close'] - lower) / (upper - lower)
    except ZeroDivisionError as e:
        b = 0
    if b == float('-inf'): b = -99999
    elif b == float('inf'): b = 99999
    return BW,b

def BB_update(df,t):
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'BW'],df.ix[i,'%b'] = BB_cal(df,i)
    return df
######################################################################################
def EMA_create(df):
    EMAlist = [[12,'12-EMA','Adj Close'],[26,'26-EMA','Adj Close']]
    for days,name,base in EMAlist:
        df[name] = 'NaN'
        for i in range(len(df)):
            df.ix[i,name] = EMA_cal(df,days,name,base,i)
    return df

def EMA_cal(df,days,name,base,i):
    if i > days-1: return ((df.ix[i,base] - df.ix[i-1,name])*
                           (2/(days+1)) + df.ix[i-1,name])
    elif 0 < i < days-1: return ((df.ix[i,base] - df.ix[i-1,name])*
                                 (2/(i+2)) + df.ix[i-1,name])
    elif i == 0: return df.ix[0,base]
    elif i==days-1: return (sum((df.ix[:days,base]))/days)

def EMA_update(df,t):
    EMAlist = [[12,'12-EMA','Adj Close'],[26,'26-EMA','Adj Close']]
    for days,name,base in EMAlist:
        for i in range(len(df)+(t+1),len(df)):
            df.ix[i,name] = EMA_cal(df,days,name,base,i)
    return df
######################################################################################
def MACD_DEM_OSC_create(df):
    df['MACD'] ='NaN'
    df['DEM'] = 'NaN'
    df['OSC'] = 'NaN'
    for i in range(len(df)):
        df.ix[i,'MACD'] = MACD_cal(df,i)
    for i in range(len(df)):
        df.ix[i,'DEM'] = DEM_cal(df,i)
    for i in range(len(df)):
        df.ix[i,'OSC'] = OSC_cal(df,i)
    return df

def MACD_cal(df,i):
    return df.ix[i,'12-EMA'] - df.ix[i,'26-EMA']

def DEM_cal(df,i):
    return EMA_cal(df,9,'DEM','MACD',i)

def OSC_cal(df,i):
    return df.ix[i,'MACD'] - df.ix[i,'DEM']

def MACD_DEM_OSC_update(df,t):
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'MACD'] = MACD_cal(df,i)
        df.ix[i,'DEM'] = DEM_cal(df,i)
        df.ix[i,'OSC'] = OSC_cal(df,i)
    return df
######################################################################################
def RSI_SMA_create(df):
    RSI_list = [[9,'9-RSI'],[14,'14-RSI'],[21,'21-RSI']]
    RSI_SMA_list = [['9-9S-RSI','9-RSI'],['14-9S-RSI','14-RSI'],
                    ['21-9S-RSI','21-RSI']]
    for days,name in RSI_list:
        RSI = RSI_cal(df,days)
        for j in range(len(df)):
            df.ix[j,name] = RSI[j]
    for SMAname,SMAbase in RSI_SMA_list:
        for i in range(len(df)):
            df.ix[i,SMAname] = MA_cal(df,9,SMAbase,i)
    return df

def RSI_cal(df,days):
    U=[0,]
    D=[0,]
    for i in range(len(df)-1):
        if df.ix[i,'Adj Close'] < df.ix[i+1,'Adj Close']:
            U.append(df.ix[i+1,'Adj Close'] - df.ix[i,'Adj Close'])
            D.append(0)
        elif df.ix[i,'Adj Close'] > df.ix[i+1,'Adj Close']:
            U.append(0)
            D.append(df.ix[i,'Adj Close'] - df.ix[i+1,'Adj Close'])
        elif df.ix[i,'Adj Close'] == df.ix[i+1,'Adj Close']:
            U.append(0)
            D.append(0)
    avg_U=[0,sum(U[1:days+1])/days,]
    avg_D=[0,sum(D[1:days+1])/days,]
    for i in range(1,days):
        avg_U.insert(i,(avg_U[i-1]*(i+1-1)+U[i])/i+1)
        avg_D.insert(i,(avg_D[i-1]*(i+1-1)+D[i])/i+1)
    for i in range(days+1,len(df)):
        avg_U.append((avg_U[i-1]*(days-1)+U[i])/days)
        avg_D.append((avg_D[i-1]*(days-1)+D[i])/days)

    RSI = [50,]
    for i in range(1,len(df)):
        try:
            RSI.append((1 - (1 / (1+((avg_U[i]) / (avg_D[i]))))) * 100)
        except:
            if avg_U == 0 and avg_D == 0: RSI.append(0)
            elif avg_D == 0 and avg_U != 0: RSI.append(100)
    return RSI
########################################################################################
def RSV_cal(df,days,i):
    H = max([x for x in df.ix[i+1-days:i+1,'High']])
    L = min([x for x in df.ix[i+1-days:i+1,'Low']])
    try:
        RSV = ((df.ix[i,'Adj Close'] - L)/(H-L)) * 100
    except:
        RSV = 50
    return RSV

def KD_create(df):
    KDlist = [[9,'9-FK'],[14,'14-FK'],[18,'18-FK']]
    avg_KDlist = [[3,'{}-FD(3)'],[5,'{}-FD(5)']]
    SDlist = [[3,'{}-SD3'],[5,'{}-SD5']]
    for days,name in KDlist:
        df[name] = 'NaN'
        for i in range((len(df))-1,days-2,-1):
            df.ix[i,name] = RSV_cal(df,days,i)
        for i in range(days-1):
            df.ix[i,name] = RSV_cal(df,i+1,i)
        for avg_days,avg_name in avg_KDlist:
            df[avg_name.format(str(days))] = 'NaN'
            df.ix[0,avg_name.format(str(days))] = 50
            for i in range(len(df)):
                df.ix[i,avg_name.format(str(days))] = MA_cal(df,avg_days,name,i)
            for avg_avg_days,avg_avg_name in SDlist:
                df[avg_avg_name.format(avg_name.format(str(days)))] = 'NaN'
                for i in range(len(df)):
                    tmpval = MA_cal(df,avg_avg_days,avg_name.format(str(days)),i)
                    df.ix[i,avg_avg_name.format(avg_name.format(str(days)))] = tmpval
    return df

def KD_update(df,t):
    KDlist = [[9,'9-FK'],[14,'14-FK'],[18,'18-FK']]
    avg_KDlist = [[3,'{}-FD(3)'],[5,'{}-FD(5)']]
    SDlist = [[3,'{}-SD3'],[5,'{}-SD5']]
    for days,name in KDlist:
        for i in range(len(df)+(t+1),len(df)):
            df.ix[i,name] = RSV_cal(df,days,i)
        for avg_days,avg_name in avg_KDlist:
            for i in range(len(df)+(t+1),len(df)):
                df.ix[i,avg_name.format(str(days))] = MA_cal(df,avg_days,name,i)
            for avg_avg_days,avg_avg_name in SDlist:
                for i in range(len(df)+(t+1),len(df)):
                    tmpval2 = MA_cal(df,avg_avg_days,avg_name.format(str(days)),i)
                    df.ix[i,avg_avg_name.format(avg_name.format(str(days)))] = tmpval2
    return df
######################################################################################
def pdi_ndi_adx_cal(df):
    tr_list = [max(df.ix[0,'High'] - df.ix[0,'Low'],
                   abs(df.ix[0,'High'] - df.ix[0,'Adj Close']),
                   abs(df.ix[0,'Adj Close'] - df.ix[0,'Low'])),]
    for i in range(1,len(df)):
        tr_list.append(max(df.ix[i,'High'] - df.ix[i,'Low'],
                           abs(df.ix[i,'High'] - df.ix[i-1,'Adj Close']),
                           abs(df.ix[i-1,'Adj Close'] - df.ix[i,'Low'])))
    tr14_list = [0,sum(tr_list[1:15])/14,]
    for i in range(1,14):
        tr14_list.insert(i,tr14_list[i-1]-(tr14_list[i-1]/14) + tr_list[i])
    for i in range(15,len(df)):
        tr14_list.append(tr14_list[i-1]-(tr14_list[i-1]/14) + tr_list[i])

    pdm_list = [0,]
    ndm_list = [0,]
    for i in range(1,len(df)):
        pdm = df.ix[i,'High'] - df.ix[i-1,'High']
        ndm = df.ix[i-1,'Low'] - df.ix[i,'Low']
        if pdm > ndm and pdm > 0: pass
        else: pdm = 0
        if ndm > pdm and ndm >0: pass
        else: ndm = 0
        pdm_list.append(pdm)
        ndm_list.append(ndm)
    pdm14_list = [0,sum(pdm_list[1:15])/14,]
    ndm14_list = [0,sum(ndm_list[1:15])/14,]
    for i in range(1,14):
        pdm14_list.insert(i,pdm14_list[i-1]-(pdm14_list[i-1]/14) + pdm_list[i])
        ndm14_list.insert(i,ndm14_list[i-1]-(ndm14_list[i-1]/14) + ndm_list[i])
    for i in range(15,len(df)):
        pdm14_list.append(pdm14_list[i-1]-(pdm14_list[i-1]/14) + pdm_list[i])
        ndm14_list.append(ndm14_list[i-1]-(ndm14_list[i-1]/14) + ndm_list[i])
    pdi_ndi_list = [[0,0],]
    for i in range(1,len(df)):
        pdi = pdm14_list[i]/tr14_list[i]*100
        ndi = ndm14_list[i]/tr14_list[i]*100
        pdi_ndi_list.append([pdi,ndi])


    dx_list = [0,]
    for i in range(1,len(pdi_ndi_list)):
        try:
            dx_list.append(abs(pdi_ndi_list[i][0] - pdi_ndi_list[i][1])/
                           abs(pdi_ndi_list[i][0] + pdi_ndi_list[i][1])*100)
        except ZeroDivisionError as e:
            dx_list.append(0)
    adx_list = [0,sum(dx_list[14:28])/14,]
    for i in range(1,27):
        adx_list.insert(i,(adx_list[i-1]*13 + dx_list[i])/14)
    for i in range(28,len(dx_list)):
        adx_list.append((adx_list[i-1]*13 + dx_list[i])/14)
    return pdi_ndi_list,adx_list

def pdi_ndi_adx_create(df):
    pdi_ndi_list,adx_list = pdi_ndi_adx_cal(df)
    df['pos_di'] = 'NaN'
    df['neg_di'] = 'NaN'
    df['ADX'] = 'NaN'

    for i in range(len(df)):
        df.ix[i,'pos_di'] = pdi_ndi_list[i][0]
        df.ix[i,'neg_di'] = pdi_ndi_list[i][1]
        df.ix[i,'ADX'] = adx_list[i]

def pdi_ndi_adx_update(df,t):
    pdi_ndi_list,adx_list = pdi_ndi_adx_cal(df)
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'pos_di'] = pdi_ndi_list[i][0]
        df.ix[i,'neg_di'] = pdi_ndi_list[i][1]
        df.ix[i,'ADX'] = adx_list[i]
######################################################################################
def obv_cal(df,i):
    if df.ix[i,'Adj Close'] > df.ix[i-1,'Adj Close']:
        return (df.ix[i-1,'OBV'] + df.ix[i,'Volume'])
    elif df.ix[i,'Adj Close'] < df.ix[i-1,'Adj Close']:
        return (df.ix[i-1,'OBV'] - df.ix[i,'Volume'])
    elif df.ix[i,'Adj Close'] == df.ix[i-1,'Adj Close']:
        return (df.ix[i-1,'OBV'])

def VWMA_cal(df,days,i):
    try:
        return (df.ix[i,'Volume']*df.ix[i,'Adj Close']
                +df.ix[i-1,'Volume']*df.ix[i-1,'Adj Close']
                +df.ix[i-2,'Volume']*df.ix[i-2,'Adj Close'])/
                (df.ix[i,'Volume']+df.ix[i-1,'Volume']+df.ix[i-2,'Volume'])
    except ZeroDivisionError as e:
        return 0

def obv_create(df):
    df['OBV'] = 'NaN'
    df['VWMA'] = 'NaN'
    df['3DSMA'] = 'NaN'
    df.ix[0,'OBV'] = df.ix[0,'Volume']
    df.ix[1,'OBV'] = df.ix[1,'Volume']
    df.ix[0,'VWMA'] = df.ix[0,'Adj Close']
    try:
        temp = (df.ix[0,'Volume']*df.ix[0,'Adj Close'] +df.ix[1,'Volume']*
                df.ix[1,'Adj Close'])/(df.ix[0,'Volume']+df.ix[1,'Volume'])
    except ZeroDivisionError as e:
        temp = 0
    df.ix[1,'VWMA'] = temp
    for i in range(2,len(df)):
        df.ix[i,'OBV'] = obv_cal(df,i)
    for i in range(len(df)):
        df.ix[i,'3DSMA'] = MA_cal(df,3,'Adj Close',i)
    for i in range(2,len(df)):
        df.ix[i,'VWMA'] = VWMA_cal(df,3,i)
    return df

def obv_update(df,t):
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'OBV'] = obv_cal(df,i)
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'3DSMA'] = MA_cal(df,3,'Adj Close',i)
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'VWMA'] = VWMA_cal(df,3,i)
    return df
#######################################################################################
def ROC_cal(df,days,i):
    if i >= days: result = ((df.ix[i,'Adj Close'] -
                            df.ix[i-days,'Adj Close'])/
                            df.ix[i-days,'Adj Close'] *100)
    elif 0 < i < days : result = ((df.ix[i,'Adj Close'] -
                                   df.ix[i-1,'Adj Close'])/
                                   df.ix[i-1,'Adj Close']*100)
    elif i == 0: result = 0
    return result
```

```
def ROC_create(df):
    df['ROC(12)'] = 'NaN'
    for i in range(len(df)):
        df.ix[i,'ROC(12)'] = ROC_cal(df,12,i)

def ROC_update(df,t):
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'ROC(12)'] = ROC_cal(df,12,i)
######################################################################################
def MFI_cal(df):
    TP_V_list = []
    for i in range(len(df)):
        TP_V_list.append([(df.ix[i,'High']+df.ix[i,'Low']+
                           df.ix[i,'Adj Close'])/3,
                           df.ix[i,'Volume']])
    MFI = []
    for j in range(len(df)-14):
        TPV_temp_list = TP_V_list[j:j+15][:]
        RMF_pos = []
        RMF_neg = []
        for i in range(1,15):
            if TPV_temp_list[i][0] > TPV_temp_list[i-1][0]:
                RMF_pos.append(TPV_temp_list[i][0]*TPV_temp_list[i][1])
            elif TPV_temp_list[i][0] < TPV_temp_list[i-1][0]:
                RMF_neg.append(TPV_temp_list[i][0]*TPV_temp_list[i][1])
        try:
            MFI.append((100 - 100/(1 + sum(RMF_pos)/sum(RMF_neg))))
        except ZeroDivisionError as e:
            MFI.append(100)
    for i in range(14):
        MFI.insert(0,float(50))
    return MFI

def MFI_create(df):
    df['MFI'] = 'NaN'
    MFI = MFI_cal(df)
    for i in range(len(df)):
        df.ix[i,'MFI'] = MFI[i]
    return df

def MFI_update(df,t):
    MFI = MFI_cal(df)
    for i in range(len(df)+(t+1),len(df)):
        df.ix[i,'MFI'] = MFI[i]
    return df
```