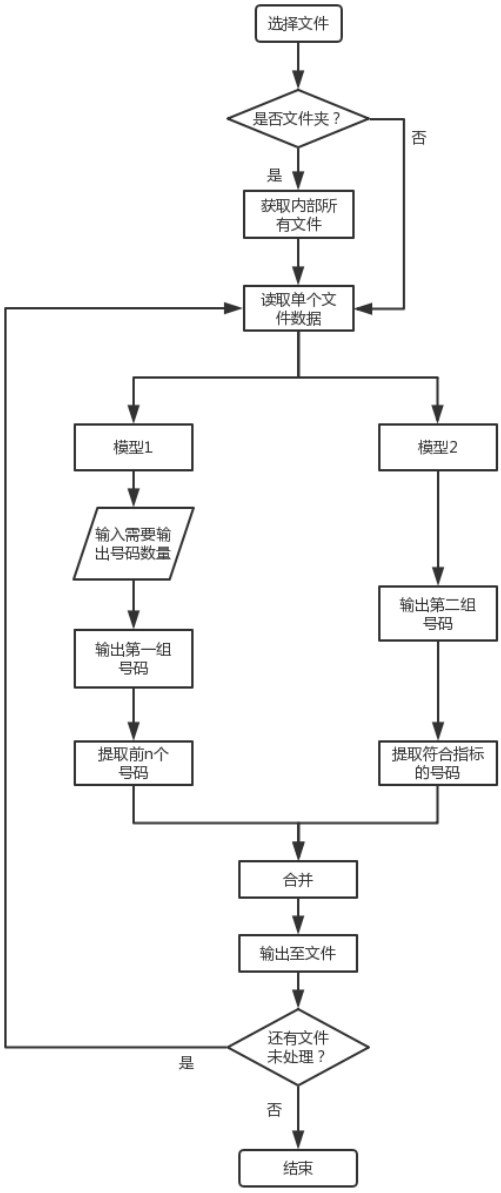
亲密度分析代码文档

1 程序结构：

1.1 文件结构：

```
|____code -r:6
| |____文档.docx
| |____ana_old_tmp.py
| |____ana_new_tmp.py
| |____phone_info_v3.py
| |____ana.py
| |____function.py
```

1.2 流程图：



1.3 文件说明：

1.3.1 ana.py:

主函数入口，包括提供图形界面。

1.3.2 phone_info_v3.py:

读取文件函数接口，提取通话记录数据，包括通话日期、通话类型、号码、去区号号码，返回类型为 json。

1.3.3 function.py:

主要提供 8 个亲密度指标的计算函数，返回类型为 json。

1.3.4 ana_old_tmp.py:

旧模型的接口，返回找到的号码、阈值、通话次数，类型为数组。

1.3.5 ana_new_tmp.py:

新模型的接口，返回号码及各个亲密度指标，类型为数组。

2 各模块功能：

2.1 ana.py:

```
def getFileList(path):     #获取指定目录下的所有文件，参数为目录地址，返回文件夹
                               里的文件列表，返回形式为数组。

def select(address):   #主计算函数，参数为文件地址。
    try:
        if num_Get: #输出号码数量，下文定义，若没有输入，则默认为 5 个。
            num_get=int(num_Get) #  若有定义输入，则用定义。
    except:
        num_get=5
    file_name=address.split('/')[-1].split('.')[0].split('/')[-1]   #拿到文件名

    #调取旧模型接口函数，参数为文件路径，需要输出号码个数：
    #返回号码、阈值、通话次数，类型均为数组：
    list_num_old,list_ratio,list_counts=get_function_old(address,num_get)

#调取新模型接口函数，参数为文件路径，需要输出号码个数：
```

```
#返回号码数组及 8 个指标(json 格式)：
list_num_new,list_1,list_2,list_3,list_4,list_5,list_6,list_7,list_8=get_function_new(address,num_get)

    phone_recover(list_num_old,phone_with_id) #恢复区号
    phone_recover(list_num_new,phone_with_id) #恢复区号

    list_num_tmp=list_num_old[:5]    #截取 list_num_old 前 5 个号码。

    #获取 list_num_1 中 list_7 中为 10.18 的号码：
    list_num_1_tmp=[]
    for i in range(len(list_num_new)):
        if list_7[i]==10.184982602:
            list_num_1_tmp.append(list_num_new[i])
    #在代码的当前文件夹写入 txt 文件，每处理一个文件，写入一个文件。
    with open('output.txt','a+') as f:
        f.write(str(file_name)+":"+"\n")
        k=1
        try:
                for i in range(len(list_num_tmp)):
                    f.write(str(k)+"-:"+str(list_num_tmp[i])+'\n')   #写入旧模型号码
                    k+=1

                for i in range(len(list_num_1_tmp)):   #写入新模型号码
                    if list_num_1_tmp[i] not in list_num_tmp:
                        f.write(str(k)+":"+str(list_num_1_tmp[i])+'\n')
                        k+=1
                f.write("\n")

        except Exception as e:
            f.write("encounter error"+str(e)+'\n') #出错，写入出错信息。

#入口函数，参数为文件或文件夹路径：
def get_function(path):
    global filelist #定义全局存储文件名的列表，定义在调用函数内部最前面。
    global filelength #全局变量，存放每个文件识别出的号码长度
    global judgelist   #标记数组，存放每个文件夹内的文件在处理的时候是否出错，
                       若出错，置 0，否则置 1.
    filelength=[]
    filelist=[] #存放文件名
    judgelist=[]
    if '.' in path:   #判断 addressPath 中是文件夹还是单个文件
        filelist.append(path) #单个文件
    else:
```

```
        getFileList(path) #文件夹，调用获取文件列表函数
    for item in filelist:
        global other_cell_phone#定义全局数组，前后调用；for 下面定义，每次调用
                                一个文件，都会产生新数组，防止不同文件相互叠加
        global phone_with_id   #带区号号码；
        global start_time #日期。
        global init_type #通话类型。
        init_type=[]
        phone_with_id=[]
        other_cell_phone=[]
        start_time=[]
        try:
            result=read_file(item) #调用读取文件函数,返回类型为 json。
            if len(result['calls'])==0 or result['calls']=='error': #判断是否读取失败。
                judgelist.append(0) #处理出错，置 0
            else:
                for it in result['calls']:
                    start_time.append(it['st']) # start_time
                    init_type.append(it['it'])   #主被叫
                    other_cell_phone.append(it['phone'][1]) #不带区号
                    phone_with_id.append(it['phone'][0])   #带区号
                select(item) #拿到数据后开始进行分析，参数为单个文件地址
        except Exception as e:
            traceback.print_exc() #打印出错信息
            judgelist.append(0) #处理出错，置 0

#GUI 界面：
class MyFrame(Frame):
    def __init__(self):
        Frame.__init__(self,None,-1,title="通话记录分析
                        ",pos=(100,100),size=(800,600))
        panel=Panel(self,-1)
        self.button1=Button(panel,-1,"打开文件",pos=(370,100))
        self.button2=Button(panel,-1,"RUN",pos=(370,200),size=(100,40))
        #self.button3=Button(panel,-1,"写入文件",pos=(370,320),size=(100,40))   #隐
            藏写入文件按钮
        self.button4=Button(panel,-1,"确定",pos=(300,150),size=(60,20))

        cb1=RadioButton(panel,-1,label="Sigle file",pos=(70,60)) #定义文件读取方
        式，单个;
        self.Bind(EVT_RADIOBUTTON,self.onClick,cb1)

        cb2=RadioButton(panel,-1,label="Folder File",pos=(150,60)) #定义文件夹读取
        方式，文件夹形式
```

```
        self.Bind(EVT_RADIOBUTTON,self.onClick,cb2)

        self.button1.Bind(EVT_BUTTON,self.getMyPath)

        StaticText(panel,-1,"文件",pos=(70,100))
        text=TextCtrl(panel,-1,pos=(100,100),size=(250,20))
        self.__TextBox1=text
        StaticText(panel,-1,"请输入想要输出的号码个数：\n(默认为 5-6
                    个)",pos=(70,150))
        text_input=TextCtrl(panel,-1,pos=(230,150),size=(60,20))
        self.__TextBox3=text_input
        self.button4.Bind(EVT_BUTTON,self.getInput)
        self.button2.Bind(EVT_BUTTON,self.run_file)
        StaticText(panel,-1,"输出",pos=(70,200))
        text_output=TextCtrl(panel,-1,pos=(100,200),style=TE_MULTILINE |
                                TE_READONLY,size=(250,200),) #多行显示&只读
        self.__TextBox2=text_output
        #self.button3.Bind(EVT_BUTTON,self.get_write_path)
        StaticText(panel,-1,"Date:2018-4-17",pos=(680,500))
        self.InitUI()

def onClick(self,event): #文件 or 文件夹,用变量 cb 表示。
        global cb
        cb=event.GetEventObject().GetLabel()
        if str(event.GetEventObject().GetLabel())=="Folder File":
                cb=2
        else:
                cb=1
def getInput(self,event): #得到输入的号码个数，存入变量 num_get.
        self.__TextBox2.Clear()
        global num_Get
        num_Get=self.__TextBox3.GetValue()
def run_file(self,event):   #运行代码
        get_function(Path)      #调用入口函数
        dial=MessageDialog(None,"处理完成！")
        dial.ShowModal()

def InitUI(self):       #自定义函数,完成菜单的设置
        menubar = MenuBar()             #生成菜单栏
        filemenu = Menu()               #生成一个菜单
        qmi1 = MenuItem(filemenu,1, "help")        #生成一个 help 菜单项
        qmi2 = MenuItem(filemenu,2, "Quit")   #quit 项，id 设为 2，在 bind 中调用
        filemenu.AppendItem(qmi1)                  #把菜单项加入到菜单中
        filemenu.AppendItem(qmi2)
```

```
        menubar.Append(filemenu, "&File")              #把菜单加入到菜单栏中
        self.SetMenuBar(menubar)                       #把菜单栏加入到 Frame 框架中
        self.Bind(EVT_MENU, self.OnQuit, id=2)       #给菜单项加入事件处理，id=2
        self.Bind(EVT_MENU, self.help_window, id=1)   #help 窗口
        self.Show(True)              #显示框架

    def OnQuit(self, e):       #自定义函数   响应菜单项
        self.Close()

    def help_window(self,event): #定义 help 窗口
        dial=MessageDialog(None,"路径及文件名称不要出现中文.\ntxt、csv、xls 文
        件请尽量使用如下数据格式：\n 第一行的列名：\n 通话日期:start_time,\n 通
        话 location:place,"\
        +"\n 通话类型:init_type,\n 对方号码：other_cell_phone,\n 通话持续时
        间:use_time\n 数据格式："\
        +"\n 通话日期:****/**/** **:**:**,\n 对方号码:**********,"\
        +"\n 通话持续时长:**:**:**\n"\
        +"json 文件格式:\n[\n   {\n     'start_time':****/**/** **:**:**\n
        'other_cell_phone':**********\n      'use_time':**:**:**\n"\
        +"     ......\n     }\n     ......\n]\n 不一样的格式可能导致无法读取！\n 如果确认
        格式没有问题，请确认文件中是否出现了乱码的空行。",pos=(10,10)) #测试用

        dial.ShowModal()

    def get_write_path(self,event): #写入文件，已取消，可看源码。
        ...
    def getMyPath(self,event): #选择文件或文件夹
        self.__TextBox2.Clear()   #清除文本框
        global Path #定义全局变量 Path，文件夹路径
        if cb==2: #代表前面 onClick()函数选择的是文件夹，打开文件夹选择框。
            dlg=DirDialog(self,"选择文件夹",style=DD_DEFAULT_STYLE)
            if dlg.ShowModal() == ID_OK:
                Path=dlg.GetPath()
                self.__TextBox1.SetLabel(Path) #设置 textbox 内容为文件内容
                global fileNum
                fileNum=0 #变量，存文件数量，全局变量，后面会用到。
                for lists in os.listdir(Path):                    # 统计文件数量
                    sub_path = os.path.join(Path, lists)
                    if os.path.isfile(sub_path):
                        fileNum = fileNum+1
                return Path #返回文件夹路径
            dlg.Destroy()
        else:   #打开文件选择框。
            dlg = FileDialog(self,u"选择文件",style=DD_DEFAULT_STYLE)
```

```
            if dlg.ShowModal() == ID_OK:
                Path=dlg.GetPath()
                self.__TextBox1.SetLabel(Path) #设置 textbox 为文件内容
                return Path
            dlg.Destroy()
if __name__ == "__main__":
    app = App()     #创建应用的对象
    myframe = MyFrame()     #创建一个自定义出来的窗口
    #myframe.Center()#正中间显示
    myframe.Show()     #这两句一定要在 MainLoop 开始之前就执行
    app.MainLoop()
```

2.2 phone_info_v3.py:

```
from datetime import datetime
import pandas as pd
import json
import re
import chardet

#号码标准化，同时删除非法数据
def re_phone(phone):
    quhao= ['010', '020', '021', '022', '023', '024', '025', '026', '027', '028', '029', '0310',
'0311', '0312', '0313', '0314', '0315', '0316', '0317', '0318', '0319', '0335', '0349', '0350',
'0351', '0352', '0353', '0354', '0355', '0356', '0357', '0358', '0359', '0370', '0371', '0372',
'0373', '0374', '0375', '0376', '0377', '0378', '0379', '0391', '0392', '0393', '0394', '0395',
'0396', '0398', '0410', '0411', '0412', '0413', '0414', '0415', '0416', '0417', '0418', '0419',
'0421', '0427', '0429', '0431', '0432', '0433', '0434', '0435', '0436', '0437', '0438', '0439',
'0451', '0452', '0453', '0454', '0455', '0456', '0457', '0458', '0459', '0464', '0467', '0468',
'0469', '0470', '0471', '0472', '0473', '0474',  '0475', '0476', '0477', '0478', '0479', '0482',
'0483', '0510', '0511', '0512', '0513', '0514', '0515', '0516', '0517', '0518', '0519', '0523',
'0527', '0530', '0531', '0532', '0533', '0534', '0535', '0536', '0537', '0538', '0539', '0541',
'0542', '0543', '0544', '0545', '0546', '0547', '0550', '0551', '0552', '0553', '0554', '0555',
'0556', '0557', '0558', '0559', '0561', '0562', '0563', '0564', '0565', '0566', '0571', '0572',
'0573', '0574', '0575', '0576', '0577', '0578', '0579', '0580', '0591', '0592', '0593', '0594',
'0595', '0596', '0597', '0598', '0599', '0610', '0611', '0612', '0613', '0614', '0615', '0616',
'0617', '0618', '0619', '0624', '0730', '0731', '0732', '0733', '0734', '0735', '0736', '0737',
'0738', '0739', '0743', '0744', '0745', '0746', '0750', '0751', '0752', '0753', '0754', '0755',
'0756', '0757', '0758', '0759', '0760', '0761', '0762', '0763', '0764', '0765', '0766', '0767',
'0768', '0769', '0770', '0771', '0771', '0772', '0773', '0774', '0774', '0775', '0775', '0776',
'0777', '0778', '0779', '0789', '0790', '0791', '0792', '0793', '0794', '0795', '0796', '0797',
'0798',  '0799',  '0812',  '0813',  '0816',  '0817',  '0818',  '0825',  '0826',  '0827',  '0830',
'0831','0832', '0832', '0833', '0833', '0834', '0835', '0836', '0838', '0839', '0851', '0852',
'0853', '0854', '0855', '0856', '0857', '0858', '0859', '0869', '0870', '0871', '0872', '0873',
'0874', '0875', '0876', '0877', '0878', '0879', '0883','0886', '0887', '0888', '0889', '0890',
```

```
'0891', '0892', '0893', '0894', '0895', '0896', '0897', '0897', '0898', '0898', '0898', '0898',
'0898', '0899', '0899', '0901', '0902', '0903', '0906', '0908', '0909', '0910', '0911', '0912',
'0913', '0914', '0915', '0916', '0917', '0919', '0930', '0931', '0932', '0933', '0934', '0935',
'0936', '0937', '0937', '0937', '0938', '0939', '0941', '0943', '0951', '0952', '0953', '0954',
'0970', '0971', '0972', '0973', '0974', '0975', '0976',  '0977', '0990', '0991', '0992', '0993',
'0994', '0995', '0996', '0997', '0998', '0999']
    #调整区号
    phone=str(phone).strip()
    if phone.startswith('400'):
        return [phone,'error']
    if phone.startswith('+'):
        phone='0'+phone[1:]
    if phone.startswith('00'):
        phone=phone[1:]
    if phone.startswith('86'):
        phone='0'+phone
    if not phone.startswith('0') and len(phone) in [9,10]:
        phone='0'+phone
    if len(phone)==11 and phone[:1] not in ['0','1']:
        phone='0'+phone
    #截取区号输出
    if not phone.replace('*','0').isdigit():
        return [phone,'error']
    l=len(phone)
    if l==14:
        return [phone,phone[3:]]
    if l==12:
        return [phone,phone[4:]]
    if l==11:
        if phone[:3] in quhao:
            return [phone,phone[3:]]
        elif phone[:4] in quhao:
            return [phone,phone[4:]]
        else:
            return[phone,phone]
    if l==10:
        return [phone,phone[3:]]
    if l==7 or l==8:
        return [phone,phone]
    return [phone,'error']
#日期标准化，返回标准化后的日期或返回 error
def re_date(data):
    data=str(data).strip()
    r=re.compile(r'(\d{4}[-/])?\d{1,2}[-/]\d{1,2}\s\d{2}:\d{2}')
```

```
        data=re.search(r,data)
        if data==None:
            return 'error'
        year=datetime.now().year   #获取当前年份
        month=datetime.now().month #获取当前月份
        data=data.group().replace('/','-')
        if data.count('-')==1:
            if int(data[0:2])<=month:
                data=str(year)+"-"+data
            else:
                data=str(year-1)+"-"+data
        return data
#主被叫标准化
def re_it(data):
        keys=['被叫','主叫','无条件呼转']
        data=str(data).strip()
        for i in keys:
            if i in data:
                return i
        return 'error'
#通话时长标准化
def re_ut(data):
        data=str(data).strip()
#       if data=="552":
#           return 'error'
        if ':' in data:
            if data.replace(':','0').isdigit():
                return data
        elif '秒' in data:
            data=data.replace('秒','')
            data=data.replace('分',':')
            data=data.replace('时',':')
            return re_ut(data)
        elif data.isdigit():
            data=int(data)
            if data>50000:
                return 'error'
            hour=data//3600
            minute=data%3600//60
            second=data%60
            res=''
            if hour:
                res=str(hour)+':'
            if res!='' or minute!=0:
```

```
            res=res+str(minute)+':'
        res=res+str(second)
        return res
    else:
        return 'error'

#探测键名
def find_keys(data,row=0):
    phone_index=[]
    st_index=[]
    it_index=[]
    ut_index=[]
    i_index=None
    p_index=None
    s_index=None
    u_index=None
    for i in data.columns:
        tem=str(data[i][row]).strip()
        if type(tem)==None:
            continue
        if re_date(tem)!='error':
            st_index.append(i)
        elif re_it(tem)!='error':
            it_index.append(i)
        elif re_ut(tem)!='error':
            ut_index.append(i)
        else:
            if tem[0]=='+':
                tem=tem[1:]
            if tem.replace('*','0').isdigit() and len(tem)>5:
                phone_index.append(i)
    if len(phone_index)>1:
        for j in range(row+1,len(data)):
            for i in phone_index:
                if data[i][j]!=data[i][row]:
                    p_index=i
                    break
            if p_index!=None:
                break
    elif len(phone_index)==1:
        p_index=phone_index[0]
    if len(st_index)>1:
        for j in range(row+1,len(data)):
            for i in st_index:
```

```
                    if data[i][j]!=data[i][row]:
                        s_index=i
                        break
                if s_index!=None:
                    break
        elif len(st_index)==1:
            s_index=st_index[0]
        if len(it_index)>1:
            for j in range(row+1,len(data)):
                for i in it_index:
                    if re_it(data[i][j])!='error':
                        i_index=i
                        break
                if i_index!=None:
                    break
        elif len(it_index)==1:
            i_index=it_index[0]
        if len(ut_index)>1:
            for j in range(row+1,len(data)):
                for i in ut_index:
                    if re_ut(data[i][j])!='error':
                        u_index=i
                        break
                if u_index!=None:
                    break
        elif len(ut_index)==1:
            u_index=ut_index[0]
        if p_index==None or s_index==None:
            if row<len(data)-1 and row<20:
                return find_keys(data,row+1)
        if 'use_time' in data.columns:
            u_index='use_time'
        if 'other_cell_phone' in data.columns:
            p_index='other_cell_phone'
        if 'init_type' in data.columns:
            i_index='init_type'
        if 'start_time' in data.columns:
            s_index='start_time'
        return [p_index,s_index,i_index,u_index]
#从 json 中自动寻找 calls 列表
def find_calls(info):
    if type(info)==dict:
        for i in info:
            calls=find_calls(info[i])
```

```
                if calls!=None:
                    return calls
        if type(info)==list:
            if len(info)>0 and type(info[0])==dict:
                for i in info[0]:
                    if type(info[0][i])==str and re_it(info[0][i])!='error':
                        return info
            for i in info:
                calls=find_calls(i)
                if calls!=None:
                    return calls
        return None
#将数据转为 DataFrame 格式后统一解析
def read_df(info):
        ocp_key=''
        st_key=''
        it_key=''
        ut_key=''
        #自动探测可能字段
        auto_key=find_keys(info)
        #print(auto_key)
        if auto_key[0]!=None:
            ocp_key=auto_key[0]
        else:
            return 'error_ocpKeyNotFound'
        if auto_key[1]!=None:
            st_key=auto_key[1]
        else:
            return 'error_stKeyNotFound'
        it_key=auto_key[2]
        ut_key=auto_key[3]
        call_list=[]
        for i in range(len(info)):
            tem={}
            phone=re_phone(info[ocp_key][i])
            st=re_date(info[st_key][i])
            if it_key==None:
                it='error'
            else:
                it=re_it(info[it_key][i])
            if ut_key==None:
                ut='error'
            else:
                ut=re_ut(info[ut_key][i])
```

```
            if phone[1]=='error' or st=='error':
                continue
            else:
                tem['phone']=phone
                tem['st']=st
                tem['it']=it
                tem['ut']=ut
                call_list.append(tem)
    return call_list
def read_json(address,encoding):
    info = pd.read_json(address,encoding=encoding)
    calls=find_calls(info.to_dict())
    calls = pd.DataFrame(calls)
    return read_df(calls)
def read_csv(address,encoding):
    info =
pd.read_csv(address,engine='python',encoding=encoding,skip_blank_lines=True,dtype=
str)
    return read_df(info)
def read_excel(address,encoding):
    info = pd.read_excel(address,encoding=encoding,skip_blank_lines=True,dtype=str)
    return read_df(info)
#根据文件名读取单个文件
def read_file(path):
    filename=path.split('/')[-1]
    user=filename.split('.')[0]
    ftype=filename.split('.')[1]
    calls=''
    with open(path,'rb') as f:
        encoding=chardet.detect(f.read())['encoding']
    if ftype=='csv':
        calls=read_csv(path,encoding)
    elif ftype=='txt':
        pass
    elif ftype=='json':
        calls=read_json(path,encoding)
    elif ftype=='xls' or ftype=='xlsx':
        calls=read_excel(path,encoding)
    else:
        pass
    data_json={}
    data_json['user']=user
    data_json['calls']=calls
    return data_json
```

2.3 function.py:

```
#导入需要的包：
from datetime import datetime,date
import numpy as np
from dateutil.parser import parse
from collections import Counter
import traceback

def log(mess): #写入日志函数，在当前文件夹下写入，日志名为当前日期。
    m=datetime.now().month
    d=datetime.now().day
    n='ana_pro_'+str(m)+"-"+str(d)
    with open(str(n)+'.log','a+',encoding='utf8') as f:
        f.write(str(mess)+str(datetime.now())+'\n')
#恢复号码区号：
def phone_recover(phone_before,phone_after):
    for i in range(len(phone_before)):
        if len(str(phone_before[i]))<11: #手机号不处理
            for item in phone_after:
                if str(phone_before[i]) in str(item) and
len(str(phone_before[i]))<len(str(item)):
                    phone_before[i]=item
#初始化字典，传入参数：号码数组，返回 value 为 0 的字典，key 为号码：
def ini_dic(phonelist):
    dic_tmp={}
    for item in phonelist:
        dic_tmp[item]=0
    return dic_tmp

#转换通话时间，格式可为 时:分:秒 或 分:秒 或 秒。
def time2se(t): #转换时间
    if str(t)=='error':
        return 0
    elif ":" not in str(t):
        return int(t)
    elif str(t).count(':')==1:
        m,s = t.strip().split(":")
        return int(m) * 60 + int(s)
    elif str(t).count(':')==2:
        h,m,s = t.strip().split(":")
        return int(h)*3600+int(m) * 60 + int(s)
    else:
        return 0
```

```
#传入通话日期的数组，返回所有日期的第一天和最后一天的间隔天数：
def get_gap(start_time):
    start_time2 = []
    for i in start_time:
        start_time2.append(i + ':00')
    result=sorted(start_time2,key=lambda date: datetime.strptime(date, \
    "%Y-%m-%d %H:%M:%S"), reverse=True)  #统一格式化并按日期排序。
    lastday = result[0]
    firstday = result[len(result) - 1]
    year = lastday.split('-')[0]
    month = lastday.split('-')[1]
    day =  lastday.split('-')[2][0:2]
    year1 = firstday.split('-')[0]
    month1 = firstday.split('-')[1]
    day1 =  firstday.split('-')[2][0:2]
    day_cha = (datetime(int(year),int(month),int(day)) - \
    datetime(int(year1),int(month1),int(day1))).days
    return day_cha
def get_desktop(): #获得 windows 桌面路径
    key = win32api.RegOpenKey(win32con.HKEY_CURRENT_USER,\
    r'Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders',\
    0,win32con.KEY_READ)
    return win32api.RegQueryValueEx(key,'Desktop')[0]

#下面是亲密度 8 个指标的计算函数，传入参数各不同：

'''
获取每个号码在六个月内的通话的最大间隔天数, phonelist 即 phone_list 的去重数组。
因为通话记录文件中，最长日期为 6 个月，所以这里不用判断是否满足 6 个月的条件。
'''
def function_A(start_time , phonelist , phone_list ):
    phone_dic={}  #key : value，号码：gap
    dic_ret={}  #key : value , 号码 : 权重
    try:
        for i in range(len(phonelist)):
            list_tmp=[]  #M 每个号码的日期列表
            for j  in range(len(start_time)):
                if phone_list[j]==phonelist[i]:
                    list_tmp.append(start_time[j]) #拿到该号码的通话日期
            phone_dic[phonelist[i]]=get_gap(list_tmp) #调用函数获得最大日期间隔。

        #按照不同的日期间隔天数，赋给不同的权重：
        for item in phonelist:
```

```
                if phone_dic[item]<=27:
                    dic_ret[item]=5.424888044
                elif phone_dic[item]>27 and phone_dic[item]<=33:
                    dic_ret[item]=2.11547407
                elif phone_dic[item]>33 and phone_dic[item]<=44:
                    dic_ret[item]=-0.036586964
                elif phone_dic[item]>44 and phone_dic[item]<=62:
                    dic_ret[item]=-0.801349703
                elif phone_dic[item]>62 and phone_dic[item]<=110:
                    dic_ret[item]=-3.61036265
                elif phone_dic[item]>110 and phone_dic[item]<=136:
                    dic_ret[item]=-3.979029711
                elif phone_dic[item]>136:
                    dic_ret[item]=-5.772336523
            return dic_ret   #返回权重，形式为 json。
        except Exception as e:
            traceback.print_exc()
            return 'error' #处理出错，返回 error。
'''
申请日前 5 个月内 日通话记录 最小条数（只统计有记录天数），因为要求 5 个月，所以需要判断通话记录中，是否满足 5 个月的条件，若超过 5 个月，需要截取申请日前 5 个月的通话记录，参数中 dur_month 为间隔月数，在 ana_new_tmp 文件中计算。
'''
def function_B(start_date,phonelist,phone_list,dur_month):
    phone_dic={} #号码：num
    dic_ret={}
    try:
        if float(dur_month)<5: #通话记录不足 5 个月按照全部记录来计算。
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(start_date)):
                    if phone_list[j]==phonelist[i]:
                        list_tmp.append(start_date[j])
                tmp=[]
                dic = Counter(list_tmp)   #统计数组中个元素个数，即通话次数，返回形式为字典
                for tt in dic.values():
                    tmp.append(tt)
                phone_dic[phonelist[i]]=min(tmp) #获取最小条数
        #如超过 5 个月，需要截取出 5 个月的通话记录：
        else:
            #类似 1 2 3 4 5 6 这种，要去掉 1：
            if int(max(start_date)-min(start_date))==5:
                tmp_date=min(start_date)+1
```

```
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(start_date)):
                    if float(start_date[j])>=float(tmp_date):
                        if phone_list[j]==phonelist[i]:
                            list_tmp.append(start_date[j])
                tmp=[]
                dic = Counter(list_tmp)   #统计数组中个元素个数，返回形式为
                                            字典
                for tt in dic.values():
                    tmp.append(tt)
                if len(tmp)!=0: #tmp 为 0 说明近 5 个月没有通话。
                    phone_dic[phonelist[i]]=min(tmp) #获取最小条数
                else:
                    phone_dic[phonelist[i]]=0
        #类似 11 12 1 2 3 4 这种。要去掉 11.
        else:
            tmp1=[]
            for item in start_date:
                d=str(item).split('.')[0]
                if len(d)==1:
                    tmp1.append(d)
            tmp_date=int(max(tmp1))+7 #例子中为单数月为 4.需要去掉 4+7 即
                                        11 月。
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(start_date)):
                    #不计算去掉的月份。
                    if str(start_date[j]).split('.')[0] != str(tmp_date):
                        if phone_list[j]==phonelist[i]:
                            list_tmp.append(start_date[j])
                tmp=[]
                dic = Counter(list_tmp)   #统计数组中个元素个数，返回形式为
                                            字典
                for tt in dic.values():
                    tmp.append(tt)
                if len(tmp)!=0:    #tmp 为 0 说明近 5 个月没有通话。
                    phone_dic[phonelist[i]]=min(tmp) #获取最小条数
                else:
                    phone_dic[phonelist[i]]=0
#赋权重：
for item in phonelist:
    if phone_dic[item]<=1:
        dic_ret[item]=1.1818106913
```

```
            elif phone_dic[item]>1 and phone_dic[item]<=2:
                dic_ret[item]=-11.48609917
            elif phone_dic[item]>2:
                dic_ret[item]=-18.65522439
        return dic_ret    #返回 json。
    except Exception as e:
        traceback.print_exc()
        return 'error'

#申请日前 5 个月内通话总时长，use_time 为通话时长数组。
def function_C(start_date , phonelist , phone_list , dur_month , use_time):
    dic_ret={} #定义返回的字典。
    phone_dic={}
    #print("use_time:",use_time)
    try:
        if float(dur_month)<5:    #判断是否超过 5 个二阅。
            for i in range(len(phonelist)):
                tmp=0
                for j in range(len(use_time)):
                    if phone_list[j]==phonelist[i]:
                        tmp+=time2se(use_time[j])
                phone_dic[phonelist[i]]=tmp
        else:
            #类似 1 2 3 4 5 6 这种，要去掉 1
            if int(max(start_date)-min(start_date))==5:
                tmp_date=min(start_date)+1
                for i in range(len(phonelist)):
                    tmp=0
                    for j in range(len(use_time)):
                        if float(start_date[j])>=float(tmp_date):
                            if phone_list[j]==phonelist[i]:
                                tmp+=time2se(use_time[j])
                    phone_dic[phonelist[i]]=tmp
            else:
                tmp1=[]
                for item in start_date:
                    d=str(item).split('.')[0]
                    if len(d)==1:
                        tmp1.append(d)
                tmp_date=int(max(tmp1))+7
                for i in range(len(phonelist)):
                    tmp=0
                    for j in range(len(use_time)):
                        if str(start_date[j]).split('.')[0]!=str(tmp_date):
```

```
                                if phone_list[j]==phonelist[i]:
                                    tmp+=time2se(use_time[j])
                        phone_dic[phonelist[i]]=tmp
            #赋权重：
            for item in phonelist:
                if int(phone_dic[item])<=238:
                    dic_ret[item]=-10.5303
                elif int(phone_dic[item])>238 and int(phone_dic[item])<=342:
                    dic_ret[item]=-9.588832248
                elif int(phone_dic[item])>342 and int(phone_dic[item])<=521:
                    dic_ret[item]=-7.804042147
                elif int(phone_dic[item])>521 and int(phone_dic[item])<=942:
                    dic_ret[item]=-5.164131972
                elif int(phone_dic[item])>942 and int(phone_dic[item])<=1294:
                    dic_ret[item]=-1.113717626
                elif int(phone_dic[item])>1294 and int(phone_dic[item])<=1529:
                    dic_ret[item]=2.8652344528
                elif int(phone_dic[item])>1529 and int(phone_dic[item])<=2720:
                    dic_ret[item]=3.1871679739
                elif int(phone_dic[item])>2720 and int(phone_dic[item])<=6285:
                    dic_ret[item]=5.638299171
                elif int(phone_dic[item])>6285 and int(phone_dic[item])<=8581:
                    dic_ret[item]=7.5679954566
                elif int(phone_dic[item])>8581 and int(phone_dic[item])<=11998:
                    dic_ret[item]=8.3371979762
                else:
                    dic_ret[item]=13.26397752
            return dic_ret
        except Exception as e:
            traceback.print_exc()
            return 'error'

#申请日前 2 个月内休息日通话时长的最小值，start_weekend 为将日期转换为星期的数组，start_date 为日期，形式为 month.day/100,比如 6.01 为 6 月 1 号。
def function_D(start_weekend , start_date , phonelist , phone_list , dur_month , use_time ):
    phone_dic={}
    dic_ret={}
    try:
        if float(dur_month)<2: #判断是否超过两个月。
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(use_time)):
                    if start_weekend[j]>=6 and  start_weekend[j]<=7: #休息日
                        if phone_list[j]==phonelist[i]:
```

```
                        #将时长转换为秒数，并存入数组。
                        list_tmp.append(time2se(use_time[j]))
            if len(list_tmp)!=0: #有通话，通过 min 函数找出最小时长。
                phone_dic[phonelist[i]]=min(list_tmp)
            else:
                phone_dic[phonelist[i]]=0
        else:
            #类似前面几个函数，通过最大最小月份判断截取方式。
            if int(max(start_date)-min(start_date))<6:  #没有 12 月出现。
                tmp_date=max(start_date)-2
                for i in range(len(phonelist)):
                    list_tmp=[]
                    for j in range(len(use_time)):
                        if float(start_date[j])>=float(tmp_date): #过滤去掉月份
                            if start_weekend[j]>=6 and start_weekend[j]<=7:
                                if phone_list[j]==phonelist[i]:
                                    list_tmp.append(time2se(use_time[j]))
                    if len(list_tmp)!=0:
                        phone_dic[phonelist[i]]=min(list_tmp)
                    else:
                        phone_dic[phonelist[i]]=0
            else:    #else 分支已经代表 12 月也在通话日期中。
                tmp1=[]
                for item in start_date:
                    d=str(item).split('.')[0] #提取出月份
                    # 通过 if 拿到除 11 和 12 月的月份，tmp1 存储除 11 和 12 外的
                      月份，比如 [1,2,3,4]等。
                    if len(d)==1 and (d not in tmp1) and int(d) <7:
                        tmp1.append(d)
                for i in range(len(phonelist)):
                    list_tmp=[]
                    for j in range(len(start_date)):
                        #len(tmp1)为 1，说明上面只拿到了 1 月。
                        if len(tmp1)==1:#只保留 12 月 和 1 月
                            if str(start_date[j]).split('.')[0]=='1' or
str(start_date[j]).split('.')[0]=='12': #不计算去掉的月份。
                                if phone_list[j]==phonelist[i]:
                                    if start_weekend[j]>=6 and
start_weekend[j]<=7:
                                        list_tmp.append(time2se(use_time[j]))
                        # 拿到了 1 月和 2 月。
                        elif len(tmp1)==2: #1 2
                            if str(start_date[j]).split('.')[0]=='1' or
str(start_date[j]).split('.')[0]=='2':
```

```
                                        if phone_list[j]==phonelist[i]:
                                            if start_weekend[j]>=6 and start_weekend[j]<=7: #工作日
                                                list_tmp.append(time2se(use_time[j]))
                                    #拿到了 1 月、2 月...超过两个月。
                                    else: #12 1 2 3 4 5 这类，只要与 5 的差的绝对值在 2 以内，就是 2 个月内。
                                        if abs(int(str(start_date[j]).split('.')[0]) - int(max(tmp1)))<2:
                                            if phone_list[j]==phonelist[i]:
                                                if start_weekend[j]>=6 and start_weekend[j]<=7: #工作日
                                                    list_tmp.append(time2se(use_time[j]))
                            if len(list_tmp)!=0:
                                phone_dic[phonelist[i]]=min(list_tmp)
                            else:
                                phone_dic[phonelist[i]]=0
                for item in phonelist:
                    if int(phone_dic[item])<=37:
                        dic_ret[item]=5.9927480344
                    elif int(phone_dic[item])>37 and int(phone_dic[item])<=63:
                        dic_ret[item]=3.691018281
                    elif int(phone_dic[item])>63 and int(phone_dic[item])<=387:
                        dic_ret[item]=1.0544188718
                    elif int(phone_dic[item])>387:
                        dic_ret[item]=-8.568019606
                return dic_ret
            except Exception as e:
                traceback.print_exc()
                return 'error'

#申请日前 5 个月内在 6 点到 11 点之间 呼入记录 数占所有通话记录数的比例（不区分呼入呼出），start_hour 为通话日期中的小时，24 小时制。
def function_E ( start_date , start_hour , phonelist , phone_list , dur_month ):
    phone_dic1={}
    phone_dic={}
    dic_ret={}
    try:
        for i in range(len(phonelist)): #初始化
            phone_dic1[phonelist[i]]=0
        if float(dur_month)<5: #通话记录不足 5 个月
            for i in range(len(phone_list)):
                if int(start_hour[i])>=6 and int(start_hour[i])<11:
                    phone_dic1[phone_list[i]]=int(phone_dic1[phone_list[i]])+1
```

```
            leng=len(phone_list)
            for i in range(len(phonelist)):
                phone_dic[phonelist[i]]='%.4f'%(phone_dic1[phonelist[i]]/leng)
        else:
            if int(max(start_date)-min(start_date))==5: #类似 1 2 3 4 5 6 这种，要去掉
1
                tmp_date=min(start_date)+1
                for i in range(len(phone_list)):
                    if float(start_date[i])>=float(tmp_date):
                        if int(start_hour[i])>=6 and int(start_hour[i])<11:
                            phone_dic1[phone_list[i]]=int
( phone_dic1[phone_list[i]] )+1
                leng=len(phone_list) #数组长度即是通话记录条数。
                for i in range(len(phonelist)):
                    phone_dic[phonelist[i]]='%.4f'%(phone_dic1[phonelist[i]]/leng)
            else:
                tmp1=[]
                for item in start_date:
                    d=str(item).split('.')[0]
                    if len(d)==1:
                        tmp1.append(d)
                tmp_date=int(max(tmp1))+7 #例子中为单数月为 4.需要去掉 4+7 即
11 月。
                for i in range(len(phone_list)):
                    if str(start_date[i]).split('.')[0]!=str(tmp_date): #不计算去掉的月份。
                        if int(start_hour[i])>=6 and int(start_hour[i])<11:
                            phone_dic1[phone_list[i]]=int( phone_dic1[phone_list[i]] )+1
                leng=len(phone_list)
                for i in range(len(phonelist)):
                    phone_dic[phonelist[i]]='%.4f'%(phone_dic1[phonelist[i]]/leng)
        for item in phonelist:
            if float(phone_dic[item])<=0.0262:
                dic_ret[item]=6.575846258
            elif float(phone_dic[item])>0.0262 and float(phone_dic[item])<= 0.1244 :
                dic_ret[item]=3.4150940198
            elif float(phone_dic[item])>0.1244 and float(phone_dic[item])<= 0.166 :
                dic_ret[item]=1.7550101573
            elif float(phone_dic[item])> 0.166   and float(phone_dic[item])<= 0.1995 :
                dic_ret[item]=0.4110557047
            elif float(phone_dic[item])> 0.1995   and float(phone_dic[item])<=0.325:
                dic_ret[item]=-1.560838162
            elif float(phone_dic[item])>0.325:
                dic_ret[item]=-5.056723647
        return dic_ret
```

```
    except Exception as e:
        traceback.print_exc()
        return 'error'

#申请日前 6 个月内在 21 点到 01 点之间 呼出记录 数占 所有通话记录数的比例:
def function_F( init_type , start_hour , phonelist , phone_list ):
    phone_dic={}
    phone_dic1={} #字典，value 为号码的呼出次数
    dic_ret={}
    try:
        for i in range(len(phonelist)): #初始化
            phone_dic1[phonelist[i]]=0
        for i in range(len(phone_list)):
            if int(start_hour[i])>=21 or int(start_hour[i])<=1: #过滤 21 到 1 点的通话
                if init_type[i]=='主叫': #判断是否主叫
                    phone_dic1[phone_list[i]]=int(phone_dic1[phone_list[i]])+1
        leng=len(phone_list)
        for i in range(len(phonelist)):
            phone_dic[phonelist[i]]='%.4f'%(phone_dic1[phonelist[i]]/leng) #计算比例

        for item in phonelist:
            if float(phone_dic[item])<=0.016:
                dic_ret[item]=8.4820253906
            elif float(phone_dic[item])>0.016 and float(phone_dic[item])<=0.0383 :
                dic_ret[item]=7.4622653527
            elif float(phone_dic[item])>0.0383   and float(phone_dic[item])<=0.0513 :
                dic_ret[item]=7.036194014
            elif float(phone_dic[item])>0.0513   and float(phone_dic[item])<=0.2305 :
                dic_ret[item]=5.9889821926
            elif float(phone_dic[item])>0.2305 :
                dic_ret[item]=-5.655053362
        return dic_ret
    except Exception as e:
        traceback.print_exc()
        return 'error'

#申请日前 6 个月内最近一次呼出距离申请日天数。
def function_G(init_type , start_date , phonelist , phone_list):
    phone_dic={}
    phone_dic1={}
    dic_ret={}
    list_tmp0=[]
    try:
        for item in start_date:
```

```
            list_tmp0.append(str(item) + ':00')
        #对所有日期排序，通过索引，拿到最新的一个日期，即申请日日期。
         tmp_date=sorted(list_tmp0, key=lambda date: datetime.strptime( date ,     \
        "%Y-%m-%d %H:%M:%S" ) ,  reverse=True)[0]
         for i in range(len(phonelist)):
            list_tmp=[]
            for j in range(len(start_date)):
                if phonelist[i]==phone_list[j]:

                    if str(init_type[j])=='主叫': #可能有些号码没有主叫，那么 list_tmp
数组会为空。
                        list_tmp.append(start_date[j])
                #start_time2 存储每个号码的通话主叫日期，若没有主叫记录，则为空
                start_time2 = []
                if len(list_tmp)!=0:
                    for item in list_tmp:
                        start_time2.append(str(item) + ':00')
                #list_tmp 为空，给这个号码赋一个很早的日期，便于最后赋权重为 0
                else:
                    start_time2.append("2010-01-01 00:00:00")
                #排序，拿到该号码的最新呼出日期。
                list_tmp2 = sorted( start_time2, key=lambda date: datetime.strptime(date,
                            "%Y-%m-%d %H:%M:%S" ) , reverse=True) [0]
                date_most_near=list_tmp2 #最后一个日期
                a1 = parse(tmp_date)    #datetime  模块处理时间
                a2 = parse(date_most_near)
                day_gap=(a1-a2).days    #两个日期的间隔天数。
                if day_gap>1000:
                    phone_dic[phonelist[i]]=0.1 #0.1 可以赋给权重 0，所以赋个 0.1 数值。
                else:
                    phone_dic[phonelist[i]]=day_gap

         for item in phonelist:
            if phone_dic[item]==0.1:
                dic_ret[item]=0
            elif phone_dic[item]<=2:
                dic_ret[item]=10.184982602
            elif phone_dic[item]>2 and phone_dic[item]<=4:
                dic_ret[item]=8.1155618579
            elif phone_dic[item]>4 and phone_dic[item]<=5:
                dic_ret[item]=6.9613270106
            elif phone_dic[item]>5 and phone_dic[item]<=6:
                dic_ret[item]=4.5959367779
            elif phone_dic[item]>6 and phone_dic[item]<=16:
```

```
                dic_ret[item]=2.6226433698
            elif phone_dic[item]>16 and phone_dic[item]<=34:
                dic_ret[item]=1.1170475316
            elif phone_dic[item]>34 and phone_dic[item]<=44:
                dic_ret[item]=-1.384859674
            elif phone_dic[item]>44 and phone_dic[item]<=66:
                dic_ret[item]=-4.010604391
            elif phone_dic[item]>66 and phone_dic[item]<=90:
                dic_ret[item]=-5.424703276
            elif phone_dic[item]>90:
                dic_ret[item]=-9.074354137
        return dic_ret
    except Exception as e:
        traceback.print_exc()
        return 'error'
#申请日前 3 个月内 工作日 日呼出 最小条数（只统计有记录天数).
def function_H(init_type , start_weekend , start_date , phonelist , phone_list , dur_month):
    phone_dic={} #号码：num
    dic_ret={}
    try:
        if float(dur_month)<3: #通话记录不足 3 个月按照全部记录来计算。
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(start_date)):
                    if start_weekend[j]>=1 and  start_weekend[j]<=5: #工作日
                        if phone_list[j]==phonelist[i]:
                            if init_type[j]=='主叫':
                                #list_tmp 存储该号码的通话日期：
                                list_tmp.append(start_date[j])
                tmp=[]
                dic = Counter(list_tmp)  #统计数组中个元素个数，返回形式为字典
                for tt in dic.values():
                    tmp.append(tt) #tmp 存储每个日期的通话条数。
                if len(tmp)!=0: #tmp 为 0 说明近 5 个月没有通话。
                        phone_dic[phonelist[i]]=min(tmp) #获取最小条数
                else:
                    phone_dic[phonelist[i]]=0
        else:
            if int(max(start_date)-min(start_date))<6: #类似 1 2 3 4 5 6 这种，要去掉 1
2 3 月。
                tmp_date=max(start_date)-3
                for i in range(len(phonelist)):
                    list_tmp=[]
                    for j in range(len(start_date)):
```

```
                    if phone_list[j]==phonelist[i]:
                        if float(start_date[j])>=float(tmp_date): #过滤去掉月份
                            if start_weekend[j]>=1 and start_weekend[j]<=5:
                                if init_type[j]=='主叫':
                                    list_tmp.append(start_date[j])
                tmp=[]
                dic = Counter(list_tmp)  #统计数组中个元素个数，返回 json
                for tt in dic.values():
                    tmp.append(tt)
                if len(tmp)!=0: #tmp 为 0 说明近 5 个月没有通话。
                    phone_dic[phonelist[i]]=min(tmp) #获取最小条数
                else:
                    phone_dic[phonelist[i]]=0
        else: #类似 11 12 1 2 3 4 这种。要去掉 11 月 12 月和 1 月.
            tmp1=[]
            for item in start_date:
                d=str(item).split('.')[0]
                if len(d)==1 and (d not in tmp1) and int(d) <7:
                    tmp1.append(d)
            for i in range(len(phonelist)):
                list_tmp=[]
                for j in range(len(start_date)):
                    if len(tmp1)==1:#只保留 11 12 1 月
                        if phone_list[j]==phonelist[i]:
                            if str(start_date[j]).split('.')[0]=='1' or
str(start_date[j]).split('.')[0]=='11' or str(start_date[j]).split('.')[0]=='12':
                                if init_type[j]=='主叫':
                                    list_tmp.append(start_date[j])
                    elif len(tmp1)==2:#只保留 12 1 2 月
                        if phone_list[j]==phonelist[i]:
                            if str(start_date[j]).split('.')[0]=='1' or
str(start_date[j]).split('.')[0]=='2' or str(start_date[j]).split('.')[0]=='12': #不计算去掉的月份。
                                if init_type[j]=='主叫':
                                    list_tmp.append(start_date[j])
                    elif len(tmp1)==3: #1 2 3。
                        if phone_list[j]==phonelist[i]:
                            if str(start_date[j]).split('.')[0] in tmp1:
                                if init_type[j]=='主叫':
                                    list_tmp.append(start_date[j])
                    else: #12 1 2 3 4 5 这类，只要与 5 的差的绝对值在 3 以内，
就是 3 个月内。
                        if phone_list[j]==phonelist[i]:
                            if abs(int(str(start_date[j]).split('.')[0]) -
int(max(tmp1)))<3:
```

```
                                        if init_type[j]=='主叫':
                                            list_tmp.append(start_date[j])
                        tmp=[]
                        dic = Counter(list_tmp)  #统计数组中个元素个数，返回形式为
字典
                        for tt in dic.values():
                            tmp.append(tt)
                        if len(tmp)!=0: #tmp 为 0 说明近 5 个月没有通话。
                            phone_dic[phonelist[i]]=min(tmp) #获取最小条数
                        else:
                            phone_dic[phonelist[i]]=0
            for item in phonelist:
                if phone_dic[item]<=1:
                    dic_ret[item]=3.2530185824
                elif phone_dic[item]>1:
                    dic_ret[item]=-9.377878727
            return dic_ret
        except Exception as e:
            traceback.print_exc()
            return 'error'
```

2.4 ana_old_tmp.py:

```
import sys
import os
import time
import datetime
from datetime import datetime,date
from phone_info_v3 import read_file
import traceback
from function import get_gap

def select(address,num_get):
    num_Get=num_get
    try:
        if num_Get:
            num_get=int(num_Get) #若有定义输入，则用定义，否则默认为 5 个
    except:
        num_get=5
    #定义各个需要的数组。
    start_hour=[]  #存放日期中的小时
    start_date=[]  #存放日-day
    start_date1=[] #存放月日,比如 10 月 2 号，以 10.02 数字存放
    start_weekday=[] #存放星期几
    phonelist=[] #存放号码，每个号码唯一；
```

```
    bj=0 #被叫数量 ， 后面代买已注释。
    zj=0 #主叫数量
    year=datetime.now().year   #获取当前年份
    month=datetime.now().month #获取当前月份
    #下面的 for 处理日期，拿到转换后的星期，小时，日期，形式均为 list。
    for item in start_time:
        if str(item).count('-')==2 and str(item).count(':')==2:
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M:%S").weekday()+1) #将日期转换为星期
            item=time.strptime(item, "%Y-%m-%d %H:%M:%S")  #将日期字符串转换为
日期格式
            start_hour.append(item.tm_hour)  #提取出时间中的小时
            start_date.append(item.tm_mday) #提取日期
            start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
        elif str(item).count('-')==2 and str(item).count(':')==1:
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M").weekday()+1)
            item=time.strptime(item, "%Y-%m-%d %H:%M")
            start_hour.append(item.tm_hour)
            start_date.append(item.tm_mday)
            start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
        elif str(item).count('-')==1 and str(item).count(':')==2:
            if int(item[0:2])<=int(month):
                item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于当
前实际月份，则添加当前年份，否则添加去年年份
            else:
                item=str(int(year)-1)+"-"+str(item)
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M:%S").weekday()+1)
            item=time.strptime(item, "%Y-%m-%d %H:%M:%S")
            start_hour.append(item.tm_hour)
            start_date.append(item.tm_mday)
            start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
        elif str(item).count('-')==1 and str(item).count(':')==1:
            if int(item[0:2])<=int(month):
                item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于当
前实际月份，则添加当前年份，否则添加去年年份
            else:
                item=str(int(year)-1)+"-"+str(item)
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M").weekday()+1)
            item=time.strptime(item, "%Y-%m-%d %H:%M")
            start_hour.append(item.tm_hour)
            start_date.append(item.tm_mday)
```

```
            start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
        else:
            string=str(address)+"time 数据格式有误，请参照 File-help！"
            dial=MessageDialog(None,string)
            dial.ShowModal()
            exit()

#从所有号码中拿出唯一的号码存进 phonelist.
for item in other_cell_phone:#添加进每个文件的号码
        if item in phonelist:
            pass
        else:
            phonelist.append(item)

#用 day_count 统计记录中出现的日期，每个日期只记录一次，记录形式为 list。
day_count=[]
for item in start_date1:
        if(item in day_count):
            pass
        else:
            day_count.append(item)

dic={}  #存放 号码:通话次数
for item in other_cell_phone:
        if(item in dic.keys()):
            dic[item]+=1
        else:
            dic[item]=1
#一个号码若每天都打电话，则标记为 1，先进行初始化。
dic_phone_day={}
dic_phone_work={}#统计工作日天数
dic_phone_fev={}#节假日
#先全部置一。
for item in other_cell_phone:
        dic_phone_work[item]=1
        dic_phone_fev[item]=1
        dic_phone_day[item]=1

#统计每个号码的工作日和周六周末通话数
for item in dic_phone_work.keys():
        i=0
        j=0
        f=0
        for call in other_cell_phone:
```

```
            if (item==call):
                if(start_weekday[i]>=1 and start_weekday[i]<=5):
                    j+=1
                elif(start_weekday[i]>=6 and start_weekday[i]<=7):
                    f+=1
            i+=1
        dic_phone_work[item]=j #工作日通话次数
        dic_phone_fev[item]=f #周六周日通话次数

    dic_phone_ratio={} #存放通话通话比例阈值，计算公式：(节假日-工作日)/(节假日+工作日)
    for item in dic_phone_day.keys():   #item 为号码
        if item in dic_phone_ratio.keys():
            pass
        else:
            if len(item)>=8: #过滤小于 8 位的服务号码。
                dic_phone_ratio[item]=float(1.0*(dic_phone_fev[item]-dic_phone_work[item])/(dic_phone_fev[item]+dic_phone_work[item]))

    #根据阈值筛初步选出号码：
    dic_phone_select1={}
    for day_Gap in range(10,35,5): #为保证找出指定个数的号码，这里调整通话天数，先考虑平均至少 10 天通话一次，15 天次之，然后最差 35 天一次
        for item in dic_phone_ratio.keys():
            if dic[item]>=(len(day_count)/day_Gap):
                dic_phone_select1[item]=dic_phone_ratio[item]
        if len(dic_phone_select1)>num_get: #拿到符合要求数量的号码，跳出循环。
            break

    #然后根据阈值排序从大到小筛选出号码：
    dic_phone_select={}
    i=0
    先排序，然后取前 num_get 个
    for item in sorted(dic_phone_select1.items(), key = lambda asd:asd[1], reverse=True):
        if i<num_get:
            dic_phone_select[item[0]]=dic_phone_ratio[item[0]]
            i+=1
        else:
            break

    #如下代码作用：统计出所有号码通话天数，即若某天有通话(无论一天中通话多少次)，则加一，
    for item in dic_phone_day.keys():
        j=0
```

```
            tmp=[]
            for call in other_cell_phone:
                if (item!=call):
                    pass
                    j+=1
                else:
                    if(start_date1[j] in tmp):
                        pass
                    else:
                        tmp.append(start_date1[j])
                    j+=1
            dic_phone_day[item]=len(tmp) #tmp 的长度即是通话天数。
        #按照测试，通话天数最多的号码很亲密，所以拿出来。
        most_related_num=max(dic_phone_day.items(),key=lambda x:x[1])[0]
        tmparray=[] #存储通过基本筛选条件的号码
        for i in range(len(other_cell_phone)):
            #下面的 if 条件是该算法中最基本的条件！！！相当于对第一次筛选的号码进行
            #第二遍的基本条件的筛选。
            if(start_hour[i]>=16    and start_weekday[i]>=5 and start_weekday[i]<=7 and
len(other_cell_phone[i])>=8):
                #要在第一次筛选的号码中。
                if(other_cell_phone[i] in dic_phone_select.keys()):
                    if(other_cell_phone[i] in tmparray):
                        pass
                    else:
                        tmparray.append(other_cell_phone[i]) #符合条件，加入 list。
        #如果最长通话天数的号码在已经筛选的号码字典中，通过阈值大小排序，筛选。
        if most_related_num in dic_phone_select:
            i=0
            for item in sorted(dic_phone_select, key = lambda asd:asd[1], reverse=True):
                if i>=num_get: #注意这里是  >=，下面 else 分支是 >。
                    break
                elif item in tmparray:
                    final_select_tmp.append(item)
                    i+=1
        else:
            i=0 #通过 i 控制最终找到的号码数量。
            final_select_tmp.append(most_related_num) #将这个特殊号码加入最终数组。
            final_select_counts.append(dic[most_related_num]) #加入通话次数。

            #最后一次筛选：
            for item in sorted(dic_phone_select, key = lambda asd:asd[1], reverse=True):
                if i>num_get:   #数量已够，挑出循环。
                    break
```

```
                elif item in tmparray:
                    final_select_tmp.append(item)
                    i+=1   #拿到一个，数量加一。

        for item in final_select_tmp:
            #这两个全局数组是最后需要返回的数组。
            final_select_tmp_ratio.append(dic_phone_ratio[item]) 阈值数组。
            final_select_tmp_num.append(dic[item]) #通话次数数组。

        phone_recover(final_select_tmp,phone_with_id) #恢复区号；

#根据 address 的文件名，调用对应解析函数
def get_function_old(path,num_get):
        global other_cell_phone#定义全局数组，这样每次调用一个文件，都会产生新数组，
            防止不同文件相互叠加
        global phone_with_id #带区号号码。
        global start_time #通话日期
        global final_select_tmp #返回的号码数组
        global final_select_tmp_ratio #返回的号码 ratio 数组
        global final_select_tmp_num #返回的号码通话次数数组
        final_select_tmp=[]
        final_select_tmp_ratio=[]
        final_select_tmp_num=[]
        phone_with_id=[]
        other_cell_phone=[]
        start_time=[]
        try:
            result=read_file(path)
            if len(result['calls'])==0 or result['calls']=='error':
                return "None"
            else:
                for it in result['calls']:
                    start_time.append(it['st']) #拿到通话日期
                    other_cell_phone.append(it['phone'][1]) #处理区号的号码。
                    phone_with_id.append(it['phone'][0]) #未处理的号码。

                #调用 select 函数进行处理，参数为路径和输出数量
                select(path,num_get)

                return  final_select_tmp,final_select_tmp_ratio,final_select_tmp_num #返回
号码，阈值，通话次数
        except:
            traceback.print_exc()
            return "error" #出错，返回 error。
```

2.5 ana_new_tmp.py:

```
import sys
import os
import pandas as pd
import numpy as np
import time
import datetime
from datetime import datetime,date
from wx import *
from function import *
from phone_info_v3 import read_file
import traceback

def select ( address , num_get ):
    num_Get=num_get
    try:
        if num_Get:
            num_get=int(num_Get) #若有定义输入，则用定义，否则默认为 5 个
    except:
        num_get=5
    phonelist=[]

    start_hour=[]  #存放日期中的小时
    start_date=[]  #存放日-day
    start_date1=[] #存放月日,比如 10 月 2 号，以 10.02 数字存放
    start_weekday=[] #存放星期几
    year=datetime.now().year  #获取当前年份
    month=datetime.now().month #获取当前月份
    for item in start_time:
        if str(item).count('-')==2 and str(item).count(':')==2:
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M:%S").weekday()+1) #将日期转换为星期
            item=time.strptime(item, "%Y-%m-%d %H:%M:%S")  #将日期字符串转换为
日期格式
            start_hour.append(item.tm_hour)  #提取出时间中的小时
            start_date.append(item.tm_mday) #提取日期
            start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
        elif str(item).count('-')==2 and str(item).count(':')==1:
            start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M").weekday()+1) #将日期转换为星期
            item=time.strptime(item, "%Y-%m-%d %H:%M")  #将日期字符串转换为日
期格式
            start_hour.append(item.tm_hour)
```

```
                start_date.append(item.tm_mday)
                start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
            elif str(item).count('-')==1 and str(item).count(':')==2:
                if int(item[0:2])<=int(month):
                    item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于
当前实际月份，则添加当前年份，否则添加去年年份
                else:
                    item=str(int(year)-1)+"-"+str(item)
                start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M:%S").weekday()+1)
                item=time.strptime(item, "%Y-%m-%d %H:%M:%S")
                start_hour.append(item.tm_hour)
                start_date.append(item.tm_mday)
                start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
            elif str(item).count('-')==1 and str(item).count(':')==1:
                if int(item[0:2])<=int(month):
                    item=str(year)+"-"+str(item)
                else:
                    item=str(int(year)-1)+"-"+str(item)
                start_weekday.append(datetime.strptime(item,
"%Y-%m-%d %H:%M").weekday()+1)
                item=time.strptime(item, "%Y-%m-%d %H:%M")
                start_hour.append(item.tm_hour)
                start_date.append(item.tm_mday)
                start_date1.append(float(item.tm_mon)+float(1.0*(item.tm_mday)/100))
            else:
                string=str(address)+"time 数据格式有误，请参照 File-help！"
                dial=MessageDialog(None,string)
                dial.ShowModal()
                exit()

        for item in other_cell_phone:#添加进每个文件的号码
            if item in phonelist:
                pass
            else:
                phonelist.append(item)

        dur_date=get_gap(start_time) #获取通话记录的第一天和最后一天间隔天数。
        dur_month='%.1f'%(dur_date/30) # 用间隔天数获得月数，形式为小数，如 5.5。

        #下面八个函数，分别调用亲密度计算函数，若计算失败，则调用 ini_dic 赋一个
value 全为 0 的字典。
        try:
            dic_A=function_A(start_time,phonelist,other_cell_phone)
```

```
        if dic_A=='error':
            dic_A=ini_dic(phonelist)
    except Exception as e:
        log("function_A") #写入日志，为了该文档过于繁杂，该文档中很多地方的写入日志已被删除，源码中没有删除。
        log(e)

    try:
        dic_B=function_B(start_date1,phonelist,other_cell_phone,dur_month)
        if dic_B=='error':
            dic_B=ini_dic(phonelist)
    except Exception as e:
        log("function_C")
        log(e)

    try:
        dic_C=function_C(start_date1,phonelist,other_cell_phone,dur_month,use_time)
        if dic_C=='error':
            dic_C=ini_dic(phonelist)
    except Exception as e:
        log("function_C")
        log(e)

    try:

dic_D=function_D(start_weekday,start_date1,phonelist,other_cell_phone,dur_month,use_time)
        if dic_D=='error':
            dic_D=ini_dic(phonelist)
    except Exception as e:
        log("function_D")
        log(e)

    try:
        dic_E=function_E(start_date1,start_hour,phonelist,other_cell_phone,dur_month)
        if dic_E=='error':
            dic_E=ini_dic(phonelist)
    except Exception as e:
        log("function_E")
        log(e)

    try:
        dic_F=function_F(init_type,start_hour,phonelist,other_cell_phone)
        if dic_F=='error':
```

```
            dic_F=ini_dic(phonelist)
    except Exception as e:
        log("function_F")
        log(e)

    try:
        dic_G=function_G(init_type,start_time,phonelist,other_cell_phone)
        if dic_G=='error':
            dic_G=ini_dic(phonelist)
    except Exception as e:
        log("function_G")
        log(e)

    try:
        dic_H=function_H( init_type , start_weekday , start_date1 , phonelist ,
                          other_cell_phone , dur_month )
        if dic_H=='error':
            dic_H=ini_dic(phonelist)
    except Exception as e:
        log("function_H")
        log(e)

    dic_ratio_tmp={}
    for item in phonelist:
        dic_ratio_tmp[item]=float
        ( dic_A[item] )+float( dic_B[item] )+float( dic_C[item] )+float( dic_D[item] )+
        float(dic _E[item])+float(dic_F[item])+float(dic_G[item])+float(dic_H[item])

    i=1
    for item in sorted(dic_ratio_tmp.items(), key = lambda asd:asd[1], reverse=True):
        #通过号码特征去除服务号码:
        if (len(item[0]) <= 9 and str(item[0]).startswith('0')) or str(item[0]).startswith('9')
or len(str(item[0])) <= 3 or (len(item[0]) <=9 and str(item[0]).startswith('1')):
            pass #跳过该号码。
        else:
            final_select_tmp.append(item[0]) #加入列表。
            i+=1
            if i>num_get: #数量达到要求，可以跳出。
                break

    for item in final_select_tmp:
        final_select_dicA.append(dic_A[item]) #需要返回的 8 个指标字典。
        final_select_dicB.append(dic_B[item])
        final_select_dicC.append(dic_C[item])
```

```
            final_select_dicD.append(dic_D[item])
            final_select_dicE.append(dic_E[item])
            final_select_dicF.append(dic_F[item])
            final_select_dicG.append(dic_G[item])
            final_select_dicH.append(dic_H[item])

        phone_recover(final_select_tmp,phone_with_id) #恢复区号。

#根据 address 的文件名，调用对应解析函数
def get_function_new(path,num_get):
        global phone_with_id
        global other_cell_phone
        global start_time
        global final_select_tmp
        global final_select_dicA
        global final_select_dicB
        global final_select_dicC
        global final_select_dicD
        global final_select_dicE
        global final_select_dicF
        global final_select_dicG
        global final_select_dicH
        global init_type
        global use_time

        log("get_function start..")
        log(str(path))

        use_time=[]
        init_type=[]
        final_select_tmp=[]
        final_select_dicA=[]
        final_select_dicB=[]
        final_select_dicC=[]
        final_select_dicD=[]
        final_select_dicE=[]
        final_select_dicF=[]
        final_select_dicG=[]
        final_select_dicH=[]
        other_cell_phone=[]
        start_time=[]
        phone_with_id=[]
        final_select=[]
        try:
```

```
        result=read_file(path)
        if len(result['calls'])==0 or result['calls']=='error':
            log("获取 calls 失败...")
            return "None"
        else:
            log("获取 calls 成功...")
            for it in result['calls']:
                start_time.append(it['st']) #start_time
                init_type.append(it['it']) #主被叫
                use_time.append(it['ut']) #通话时长
                other_cell_phone.append(it['phone'][1]) #处理过的号码
                phone_with_id.append(it['phone'][0])  #带区号的号码。

            #调用 select 函数，参数为路径，号码数量。
            select( path , num_get )

            #返回号码数组，8 个形式为字典的指标：
            return final_select_tmp , final_select_dicA , final_select_dicB ,
                final_select_dicC , final_select_dicD , final_select_dicE ,
                final_select_dicF , final_select_dicG , final_select_dicH
    except Exception as e:
        log(str(e))
        traceback.print_exc()
        return "error" #处理出错，返回 error。
```