**第2章 数据预处理与统计**

**2.1任务1.1-1.5 数据预处理**

**2.1.1各工序工作时长处理**

由于案卷的处理时间和操作人员的工作时长应去掉非工作时间，且工作时间为周一至周六上午8:30-12:00和下午13:00-18:00，所以我们应初步筛选出非工作时间的数据，将这些数据归到工作时间范围内以去除，方便后续对各工序消耗时长的计算。具体代码如下

from datetime import datetime

def wb(given\_datetime):

    # 判断是否星期日（星期天），0代表星期一，6代表星期日

    if given\_datetime.weekday() == 6:

        return False

    else:

        return True

判断是否星期日（星期天），0代表星期一，6代表星期日，清理结果无在周日工作，7月也无法定节假日。

对开始与结束工序事件进行清理，如果开始工序事件在上班之前均归类为最近的上班事件，即为8：30或13：00，如果为下班后均归类为下班时间，即12：00和18：00。具体代码如下：

from datetime import datetime, time

def Anti\_scroll\_dog\_start(given\_timestamp):

    # 提取给定时间戳的年月日

    year, month, day = given\_timestamp.year, given\_timestamp.month, given\_timestamp.day

    # 定义每天的时间段范围

    start\_time = pd.Timestamp(datetime(year, month, day, 8, 30, 0))

    end\_time = pd.Timestamp(datetime(year, month, day, 12, 0, 0))

    work\_start\_pm = pd.Timestamp(datetime(year, month, day, 13, 0, 0))

    work\_end\_pm = pd.Timestamp(datetime(year, month, day, 18, 0, 0))

    # 判断给定 Timestamp 是否在时间段范围内

    if start\_time <= given\_timestamp <= end\_time or work\_start\_pm <= given\_timestamp <=work\_end\_pm:

        return given\_timestamp

    else:

        if given\_timestamp <= start\_time:

            return start\_time

        if end\_time <= given\_timestamp <= work\_start\_pm:

            return work\_start\_pm

        if given\_timestamp >= work\_end\_pm:

            return work\_end\_pm

由于PDF处理无需计算耗时，对于data.xlsx所给数据中进行PDF处理缺失的工序结束时间直接填充与开始时间相同的值，即为0。

**2.2任务1.1-1.5 数据统计**

本问主要是统计分析档案数字化流程的耗时和进度，操作人员的工作量和工作效率情况。需要注意的是各工序的耗时是该工序的开始时间至结束时间的时长，但不包括非工作时长。针对任务1.1，需要根据data.xlsx给出加工处理各工序的数据，将数据分别按照工序和案卷号分类，统计出四道工序最终的案卷数量及各工序各案卷的耗时；针对任务1.2，需要对返工的案卷进行汇总并计算占比；针对任务1.3，需要先筛选出自检全检的工序的操作人员，再计算他们每个人的返工案卷数及其工作量在该工序的占比；针对任务1.4，需要统计每个工序的完成案卷数，总耗时和平均耗时；针对任务1.5，需要统计每个操作人员每个工序的的工作时长，完成案卷数和每个案卷的平均耗时。

**2.2.1任务1.1：统计各工序案卷数**

对于汇总各案卷各工序的开始时间及各案卷的完成时长，要使用groupby函数按照案卷编号和工序进行分组，然后使用first函数获取每个组的第一个记录，即开始时间和完成时长。重要代码如下：

for group\_name, group\_data in grouped\_data:

    print(f"工序: {group\_name}")

    arch\_group = group\_data.groupby('sARCH\_ID')

    for arch\_id, arch\_group\_data in arch\_group:

        print(arch\_id)

        min\_start\_time = np.min(arch\_group\_data['dUPDATE\_TIME'])

        max\_end\_time = np.max(arch\_group\_data['dNODE\_TIME'])

        cost = time\_consuming(min\_start\_time,max\_end\_time)

        if group\_name == '图像处理':

            tuxiang.append([min\_start\_time,max\_end\_time])

            cost\_one.append(cost)

        if group\_name == '扫描':

            shaomiao.append([min\_start\_time,max\_end\_time])

            cost\_one.append(cost)

        if group\_name == '自检全检':

            zijian.append([min\_start\_time,max\_end\_time])

            cost\_one.append(cost)

        if group\_name == 'PDF处理':

            pdf.append([min\_start\_time,max\_end\_time])

得出案卷完成时长最长的三个案卷结果如下表1所示：

**表1**

|  |  |
| --- | --- |
| 案卷号 | 完成时长 |
| 托630239-册一 | 478306.083 |
| 托629637-册一 | 278808.66 |
| 托684645-册一 | 211512.173 |