交通运输数据技术 作业三

——数据质量评价与预处理

2251140 范程

一、 总述

本次作业是基于第 6 和第 7 讲——数据质量评价与预处理的任务，分为故障数据检测、故障数据修复和数据平滑处理三个任务。本次作业的目的在于了解故障数据的判别方法、数据修补的方法和数据平滑的方法。在数据的处理和可视化方面，我使用了 R 语言作为工具。以下部分我对已经提交的代码作一些必要的说明。

二、 相关数据分析

（一） 任务一：基于数据一的故障数据检测

任务一的数据来源于 tmp001~tmp005，在进行故障数据处理之前要先读取数据，这需要导入 readxl 库，同样地，在本次作业中还需要用到的库还有 tidyverse, dplyr, ggplot2, zoo, VIM, Metrics 和 forecast 这些，在导入之前需要先通过 Console 安装。

在对导入的 NHNX39(1)检测器数据进行整理之后，我们层层递进地采用独立判断法、联合判断法和平均有效车长法对故障数据进行识别。

(1) 独立判断法

在进行独立判断法需要我们对相关参数按照其特征进行阈值设定。在本数据中，通过设定限速为 80km/h，占有率不超过 95%，流量不超过 17veh/20s 三个阈值来识别故障数据：

```
speed_limitation <- 80*1.3 # 限速 80
occupation_limitation <- 95 # 占有率阈值 95
volume_limitation <- 17 # 流量阈值 17
data11 <- filter(data,FINT_SPEED > speed_limitation | FINT_OCCUPY > occupation_limitation | FINT_VOLUME > volume_limitation)
data1 <- anti_join(data,data11)
m1 <- nrow(data) - nrow(data1) # 无效记录数,m2,m3 也是
m1
```

(2) 联合判断法

联合判断法的要求是根据多个数据之间的关联性来判定数据的合理性。在本任务中，很明显故障的数据有这些类型：

流量 volume	*速度 speed*	*占有率 occupy*
+	**0**	**any**
0	+	**any**
+	+	**0**
0	**0**	+

因此需要用 filter()函数基于(1)中的 data1 进行进一步的筛选，如流量和速度都为正但占有率为 0 的情况，可以通过以下代码实现：

```
data24 <- filter(data1,FINT_VOLUME == 0 & FINT_SPEED == 0 & FINT_OCCUPY > 0)
```

之后需要对四个数据进行抗卷积处理，用到 anti_join()函数，就是在 data1 中去掉这些数据，才能得到 m2 和 data2。

(3) 平均有效车长法

平均有效车长可以通过以下公式来计算：$average\_length = \frac{v \times o}{q}$，也就是以下代码：

```
avg_length <- (data2$FINT_SPEED / 3.6) *(data2$FINT_OCCUPY/100)/(data2$FINT_VOLUME / 20)
```

之后采用和之前类似的方法来过滤无效数据。

(4) 数据统计和时序图绘制

经过上面三步过滤筛选得到了有效数据 data3，在有效性计算的时候要把数据按日期和时间进行统计，以小时有效性为例，可以用以下代码实现：

```
data3$FDT_TIME <- as.POSIXct(data3$FDT_TIME, format = "%Y-%m-%d %H:%M:%S")
hour_counts <- data3 %>%
  group_by(date_hour = format(FDT_TIME, "%Y-%m-%d %H", trim = FALSE)) %>%
  summarise(count = n())# 计算每小时的数据量 hour_counts
hour_counts1 <- hour_counts$count/180
h_validity <- sum(hour_counts1)/24/7
```

先把 data3 中的数据的时间日期格式统一，再按照日期和小时分组后把每日每小时的统计出来。根据数据有效性的定义：$valid = \frac{n_{valid}}{n_{total}} \times 100\%$，可以分别计算出数据的小时有效性和日有效性：

```
> h_validity
[1] 0.9789352
> d_validity
[1] 0.9789352
```

不同方法统计出的无效记录数：

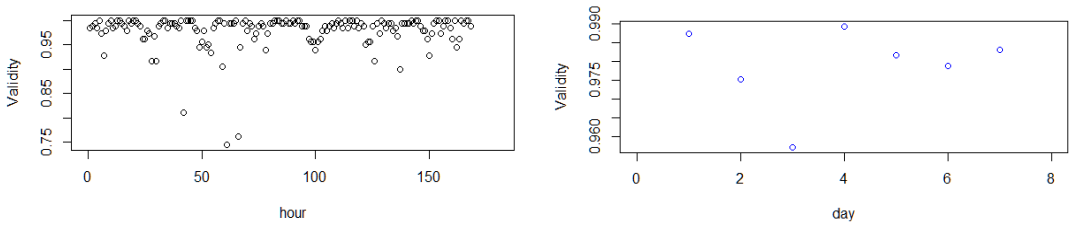
独立判断法(m1)

联合判断法(m2)

平均有效车长法(m3)

```
> m1
[1] 217
>
> m2
[1] 97
> m3
[1] 267
```

画出时序图：（用的是散点图，日期 1-7 是从 4-17 至 4-24，小时 0-168 是一周的 168 小时）



可以看出小时有效性图中有几个比较低的离群值，在 60h 也就是第三天左右，也对该日的日有效性产生了一定的影响。但总体来说数据有效性还是比较高的。

（二） 任务二：基于数据二的故障数据修复

任务二的要求是在 NHNX(39)的数据中随机选择 100 行数据赋为空值，用分别用 K 近邻法和相邻时间平均值法修复数据。对于随机选择的原数据，可以用以下方法：

```
selected_rows <- sample(1:3597, 100) # 先保存被选中的行
origin_values <- imputation_data[selected_rows, "speed_391_23"] # 原值另存
origin_values <- unlist(origin_values)
imputation_data[selected_rows, "speed_391_23"] <- NA # 赋为空值
```

(1) K 近邻法

用 VIM 库中的函数 knn()就可以根据定好的 k 值来进行数据修复。以 k=3 为例：

```
imputation_3 <- kNN(imputation_data, variable = 'speed_391_23', k = 3)
repaired_3 <- imputation_3[selected_rows, "speed_391_23"] # 提取被修复的值
```

在计算完毕之后，调用 Metrics 库中的 rmse()和 mape()函数来计算均方根误差和平均绝对百分比误差。得到 k=3 和 k=5 的 RMSE 和 MAPE 值如下：

```
> rmse_3
[1] 17.79663
> rmse_5
[1] 16.91301
```

```
> mape_3
[1] 0.5398529
> mape_5
[1] 0.5187597
```

说明 k=5 的时候的修复效果略好。

(2) 相邻时间平均值法

采用离线处理的方法调用自定义的 fill_missing_values()函数，用不同的窗口取值（3 个和 5 个）来修复数据。也就是：$y_{avg} = \frac{\sum_{i=1}^{n} y(k-i)+y(k+i)}{n}$.由于是离线处理，所以参考的值是向前向后的 3 个或 5 个的均值。最后计算得到 RMSE 和 MAPE 值分别为：

```
> rmse3
[1] 9.485445
> rmse5
[1] 8.994242
> mape3
[1] 0.2565404
> mape5
[1] 0.2443155
```

通过两种方法的比较，发现在数据齐全的情况下，参考了前后时刻的离线处理的相邻时间平均法效果更好。

（三） 任务三：基于数据一的数据平滑处理

提取检测器 NHNX(39)在 4-24-16:00~18:00 的数据：

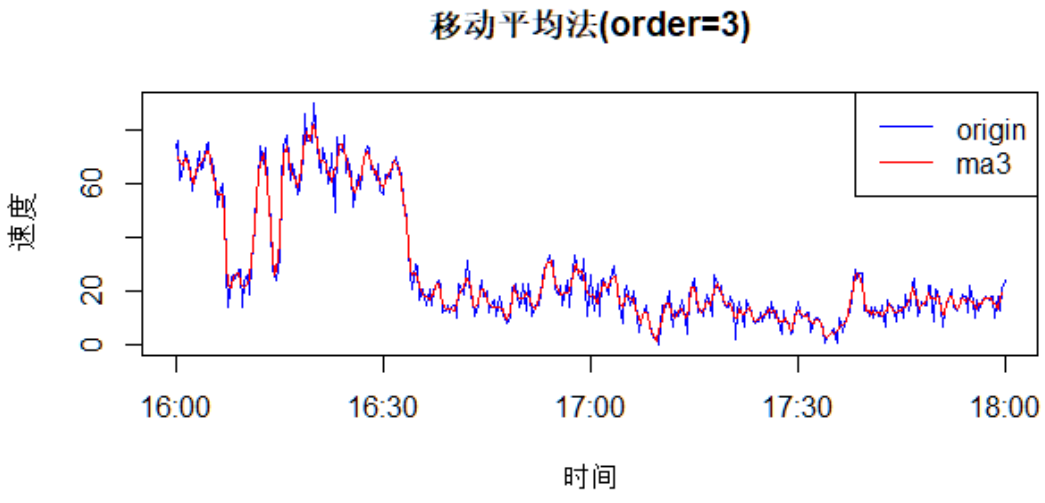
```
NHNX39_speed<data[data$FSTR_LOOPGROUPID=='NHNX39(1)'&as.Date(data$FDT_TIME)==as.Date('2010-04-24') & format(data$FDT_TIME,"%H:%M:%S") >= "16:00:00" & format(data$FDT_TIME,"%H:%M:%S") <= "18:00:00", c("FDT_TIME","FINT_SPEED")]
```

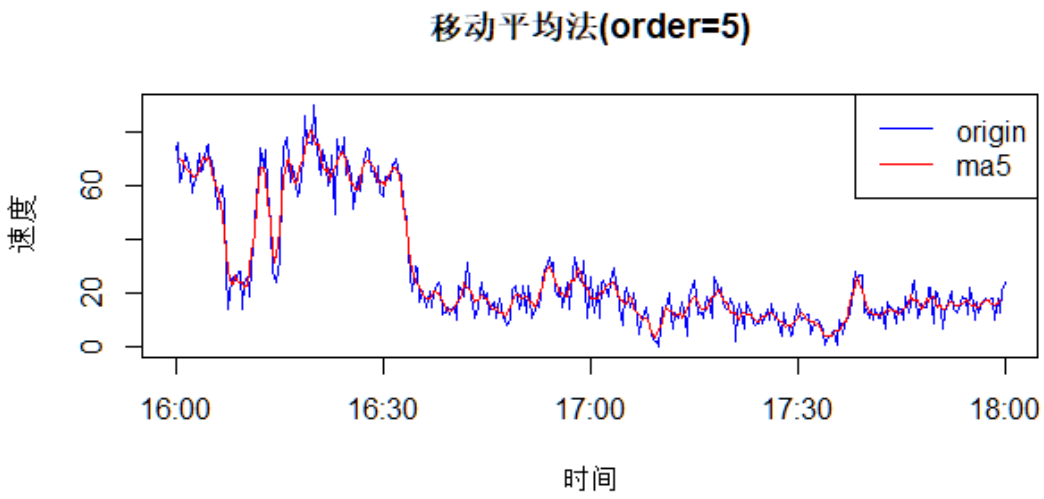
(1) 移动平均法

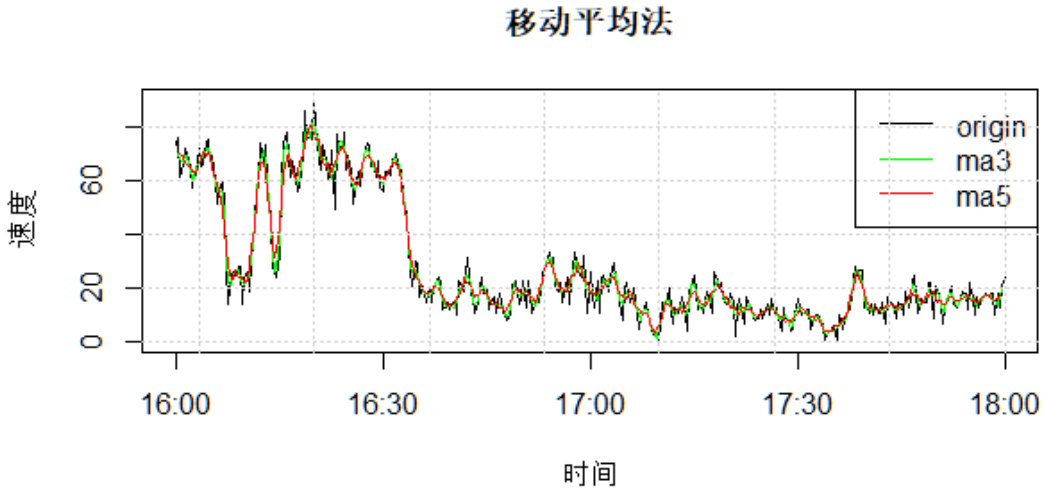
调用 forecast 库中的 ma()函数，之后给定窗口大小取值来进行数据平滑。以窗口大小取值=3 为例：

```
ma_NHNX39_3 <- forecast::ma(NHNX39_speed$FINT_SPEED,order = 3)
```

得到的 ma_NHNX39_3 就是平滑之后的数据。分别取 order=3 和 order=5，得到平滑数据之后分别与原来的数据作时序图进行比较来观察平滑效果：







发现在平滑效果上两种方法都比原数据效果要好很多，让很多可能是不可导的点变得更加平滑，且窗口取值为 5 时效果好于窗口取值为 3 时。可能是由于窗口取值越大，每一点被平均后的值的波动越小。

(2) 简单指数平滑法

给定平滑参数$\alpha \in (0,1), z(k) = \alpha y(k) + (1-\alpha)z(k-1)$，则可以通过调用自定义的函数 myfunction(data,alpha)，在函数中让k遍历所有的取值：

```
for(i in 2:n){smoothed_speed_data[i]<-alpha*data[i]+(1-alpha)*smoothed_speed_data[i-1]}
```

将第一个元素赋值为输入数据的第一个元素，作为初始值。接下来的 for 循环从第二个元素开始遍历输入数据，对每个元素进行指数平滑处理，并将结果保存在 smoothed_speed_data 中。

分别取$\alpha = 0.3, \alpha = 0.6$，平滑并绘图：

能够很明显地看出在$\alpha = 0.6$时，图像的尖锐部分明显变得平滑了，且从平滑意义上效果好于$\alpha = 0.3$时。但是当我在把α的取值调整成 0.7 甚至更高的时候效果就没有这么改变了，说明对于同一数据α的取值有一个阈值，但总体来看时取值大平滑效果好，但可能对数据的真实性造成一些影响。

三、 反思与总结

本次任务调用了很多库和函数，分别采用不同的方法进行了数据质量评价和处理。在处理大规模的交通数据时，数据的有效性、完整性和不同的修复方法都会影响数据处理的结果，而且相关参数取值不是越高越好，需要具体问题具体分析。