前言

基于上一期《【干货】手把手带你抓“网上购物”类APP信息（含代码）》的文章，相信各位已经有了1000多种APP的下载信息了吧。本期，我们就针对此数据，再给大家分享一下如何完成数据的清洗和可视化，在探索过程中，我们将回答这几个问题：

- 各类APP中，下载量前5的都是哪些应用？
- 各类APP中，好评率最差的5个APP又是哪些？
- 评论人数与好评率是否存在某种关系呢？

在回答这5个问题之前，需要对数据做一些清洗和处理，包括重复观测的检查及删除、异常值的处理、缺失值的处理、数据类型的转换等。接下来我们就以爬虫获得的数据开始探索数据的面貌。

数据窥探

先来作数据的一些简单的摸索，如数据集各变量的类型、数据集是否存在重复等。

```
# ========== Python3.X Jupyter ==========

# 导入第三方包
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 读取外部数据源
app_info = pd.read_excel(r'E:\Jupyter\apps.xlsx')

# 数据集的观测数量及变量数
app_info.shape

# 窥探数据前5行信息
app_info.head()

# 查看数据集各变量的类型
app_info.dtypes

# 检查数据是否有重复（一般对于爬虫数据都需要检查）
any(app_info.duplicated())

# 数值变量的描述性分析
app_info.describe()
```

从上面代码的运行结果来看，数据集中，除了评论人数comments变量是数值型的，其他变量均为字符型；通过重复性检查，数据集确实存在重复观察，需要排重；通过评论人数的描述性统计来看，尽然最小值为-1，很明显，这就是一个异常数据，需要处理。

数据清洗

- 重复观测的剔除和异常值的剔除

```
# 剔除重复观测
app_info.drop_duplicates(inplace=True)
app_info.shape

# 删除评论人数为-1的观测（因为只有2条记录）
app_info = app_info.loc[app_info.comments != -1,]
```

经过排重后，尽然发现有98条重复数据，这个比较容易理解，因为某一个APP可能被划分成多各类（如“每日优鲜”就被划分为了“商城”、“优惠”、“团购”和“快递”四类），那这个APP就容易被重复抓取几次，从而导致数据集的重复。

```
# 离散变量的统计描述
app_info.describe(include = ['object'])
```

从离散变量的描述性统计来看，变量company存在缺失值；在电商类APP中，快递类APP尽然最多，高达210个，可想而知，电商带动了多少人的就业啊~；从电商类的公司来看，淘宝绝对是独占鳌头啊，至少有8类APP都是淘宝门下的；我们还发现，好评率love一列存在642个"暂无"，即缺失值。

- 数据类型变换

从上面的数据变量类型的返回结果看，只有评论人数comments变量是数值型的，其他变量均为字符型，很明显，对于APP的下载人数、好评率和软件大小均可以转换为数值型，同时，软件的更新时间也可以转换为真正的日期型。探索过程中，细心的你一定会发现，APP的安装人数单位不一致，有的以万为单位，有的以亿为单位，有的就没有单位；同样，软件的大小，有的以MB为单位，有的则以KB为单位，所以在数据类型转换前，一定要处理好量纲。

```
# 自定义函数，处理安装人数的单位
def func(x):
    if x.find('亿') != -1:
        y = float(x[:-1])*10000
    elif x.find('万') != -1:
        y = float(x[:-1])
    else:
        y = float(x)/10000
    return(y)
# 安装人数变量的类型转换
app_info['install_new'] = app_info.install.apply(func)

# 自定义匿名函数
y = lambda x : float(x[:-2]) if x.find('MB') != -1 else float(x[:-2])/1024
# 软件大小变量的类型转换
app_info['size_new'] = app_info['size'].apply(y)

# 自定义匿名函数，将“暂无”设置为缺失值
y = lambda x : np.nan if x == '暂无' else float(x[:-1])/100
app_info['love_new'] = app_info['love'].apply(y)

# 用中位数对好评率进行填补
app_info['love_new'] = app_info.love_new.fillna(app_info.love_new.median())

# 日期类型的转换
app_info['update_new'] = pd.to_datetime(app_info['update'], format = '%Y年%m月%d日')
```

OK，数据清洗这块的任务就完成了，代码中有很多细节的地方值得注意哦~希望这些数据清洗的技巧对各位网友有所帮助。接下来，我们再来看看数据的描述性统计结果：

```
# 数值变量的描述性统计
app_info.describe()

# 删除不必要的变量
app_info.drop(['install','size','love','update'], axis = 1, inplace=True)
app_info.head()
```

数据的可视化分析

首先需要回答的第一个问题是，各类APP中，下载量前5的都是哪些应用？

```
# 各类应用安装量最多的前5个APP（产生绘图数据）
ls = []

categories = ['商城','团购','优惠','快递','全球导购']
for cate in categories:
    sub = app_info.loc[app_info.appcategory.apply(lambda x : x.find(cate) != -1),['appname','install_new']]
```

```
    # 取前5的安装量
    sub = sub.sort_values(by = ['install_new'],ascending=False)[:5]
    sub['type'] = cate
    ls.append(sub)
# 合并数据集
app_install_cat = pd.concat(ls)


# 设置绘图风格
plt.style.use('ggplot')
# 中文处理
plt.rcParams['font.sans-serif'] = 'Microsoft YaHei'

# 为了让多张子图在一张图中完成，设置子图的位置
ax1 = plt.subplot2grid((3,2),(0,0))
ax2 = plt.subplot2grid((3,2),(0,1))
ax3 = plt.subplot2grid((3,2),(1,0))
ax4 = plt.subplot2grid((3,2),(1,1))
ax5 = plt.subplot2grid((3,2),(2,0), colspan=2) # colspan指定跨过的列数

# 将图框存放起来，用于循环使用
axes = [ax1,ax2,ax3,ax4,ax5]
types = app_install_cat.type.unique()

# 循环的方式完成5张图的绘制
for i in range(5):
    # 准备绘图数据
    data = app_install_cat.loc[app_install_cat.type == types[i]]
    # 绘制条形图
    axes[i].bar(range(5), data.install_new, color = 'steelblue', alpha = 0.7)
    # 设置图框大小
    gcf = plt.gcf()
    gcf.set_size_inches(8, 6)
    # 添加标题
    axes[i].set_title(types[i]+'类APP下载量前5的应用', size = 9)
    # 设置刻度位置
    axes[i].set_xticks(np.arange(5) + 0.4)
    # 为刻度添加标签值
    axes[i].set_xticklabels(data.appname, fontdict={'fontsize':7},  color = 'red')
    # 删除各子图上、右和下的边界刻度标记
    axes[i].tick_params(top = 'off', bottom = 'off', right = 'off')

# 调整子图之间的水平间距和高度间距
plt.subplots_adjust(hspace=0.6, wspace=0.3)

# 显示图形
plt.show()
```

由上图可知，从商城类和优惠类的应用中，唯品会APP在豌豆荚市场上的下载量都跃居前五；对于团购类和快递类，美团和菜鸟的下载量还是占绝对优势的；全球导购类下载量前三的APP并没有太大差异，且下载量并不大，个人觉得主流商城APP（如天猫，京东，唯品会）都有海淘业务，基本能满足消费者需求。

再来回答第二个问题，各类APP中，好评率最差的5个APP又是哪些？下面的这段代码与上面的代码基本一致，只是换了数据，所以，读者要确保能够读懂第一段绘图语句。

```
# 各类应用好评率最低的前5个APP（产生绘图数据）
ls = []
categories = ['商城','团购','优惠','快递','全球导购']
for cate in categories:
    sub = app_info.loc[app_info.appcategory.apply(lambda x : x.find(cate) != -1),['appname','love_new']]
    # 取前5的安装量
    sub = sub.sort_values(by = ['love_new'])[:5]
    sub['type'] = cate
    ls.append(sub)
app_love_cat = pd.concat(ls)

# 为了让多张子图在一张图中完成，设置子图的位置
ax1 = plt.subplot2grid((3,2),(0,0))
```

```
ax2 = plt.subplot2grid((3,2),(0,1))
ax3 = plt.subplot2grid((3,2),(1,0))
ax4 = plt.subplot2grid((3,2),(1,1))
ax5 = plt.subplot2grid((3,2),(2,0), colspan=2) # colspan指定跨过的列数

# 将图框存放起来，用于循环使用
axes = [ax1,ax2,ax3,ax4,ax5]
types = app_love_cat.type.unique()

# 循环的方式完成5张图的绘制
for i in range(5):
    # 准备绘图数据
    data = app_love_cat.loc[app_love_cat.type == types[i]]
    # 绘制条形图
    axes[i].bar(range(5), data.love_new, color = 'steelblue', alpha = 0.7)
    # 设置图框大小
    gcf = plt.gcf()
    gcf.set_size_inches(8, 6)
    # 添加标题
    axes[i].set_title(types[i]+'类APP好评率后5的应用', size = 9)
    # 设置x轴刻度位置
    axes[i].set_xticks(np.arange(5) + 0.4)
    # 为x轴刻度添加标签值
    axes[i].set_xticklabels(data.appname, fontdict={'fontsize':7},  color = 'red')
    # 设置y轴刻度位置
    axes[i].set_yticks(np.arange(0,0.6,0.15))
    # 为y轴刻度添加标签值
    axes[i].set_yticklabels([str(i*100) + '%' for i in np.arange(0,0.6,0.15)])
    # 删除各子图上、右和下的边界刻度标记
    axes[i].tick_params(top = 'off', bottom = 'off', right = 'off')


# 调整子图之间的水平间距和高度间距
plt.subplots_adjust(hspace=0.6, wspace=0.3)
plt.show()
```

上图反映的是，各类APP好评率最差的5个应用，有几个是我比较出乎意料，如商城类APP中，熟知的沃尔玛、千牛和建行善融商城也在里面；聚划算居然都出现在了团购类和优惠类APP中；对于快递类的应用来说，中国邮政口碑最差，其次是京东众包（大概了解了一下京东众包，这是一个全民快递服务APP，可以理解成C2C的服务）。

最后，再来回答第三个问题，评论人数与好评率是否存在某种关系呢？

```
# 导入第三方模块
from sklearn.linear_model import LinearRegression

# 评价人数与好评率是否存在关系呢?
# 散点图
plt.scatter(app_info.comments, # 评价人数
            app_info.love_new, # 好评率
            s = 30, # 设置点的大小
            c = 'black', # 设置点的颜色
            marker = 'o', # 设置点的形状
            alpha = 0.9, # 设置点的透明度
            linewidths = 0.3, # 设置散点边界的粗细
            label = '观测点'
            )

# 建模
reg = LinearRegression().fit(app_info.comments.reshape(-1,1), app_info.love_new)
# 回归预测值
pred = reg.predict(app_info.comments.reshape(-1,1))

# 绘制回归线
plt.plot(app_info.comments, pred, linewidth = 2, label = '回归线')
plt.legend(loc = 'lower right')

# 添加轴标签和标题
plt.title('评论人数与好评率的关系')
```

```
plt.xlabel('评论人数')
plt.ylabel('好评率')

# 去除图边框的顶部刻度和右边刻度
plt.tick_params(top = 'off', right = 'off')
# 显示图形
plt.show()
```

从这1100个APP的样本来看，似乎评论人数与好评率存在正向关系，但也不好说，毕竟好评率有太多的确实，这里只是用来中位数来替补的。如果感兴趣的话，你们也可以试试把确实的那些样本删除之后，再看看两者的关系。我们再来看一下"评论人数"数据的分布情况：

```
# 评论人数的描述统计
app_info.comments.describe(percentiles=np.arange(0,1.2,0.2))
```

有8成的APP，其评论人数不超过53人，数据太过偏态了。这里先筛选出评论人数不超过55人的app，然后，对其研究"评论人数"与"好评率"的关系。

```
# 散点图
sub_data = app_info.loc[app_info.comments <= 55,]
# sub_data = app_info.loc[app_info.comments > 55,]
plt.scatter(sub_data.comments, # 评价人数
            sub_data.love_new, # 好评率
            s = 30, # 设置点的大小
            c = 'black', # 设置点的颜色
            marker = 'o', # 设置点的形状
            alpha = 0.9, # 设置点的透明度
            linewidths = 0.3, # 设置散点边界的粗细
            label = '观测点'
            )

# 建模
reg = LinearRegression().fit(sub_data.comments.reshape(-1,1), sub_data.love_new)
# 回归预测值
pred = reg.predict(sub_data.comments.reshape(-1,1))

# 绘制回归线
plt.plot(sub_data.comments, pred, linewidth = 2, label = '回归线')
plt.legend(loc = 'lower right')

# 添加轴标签和标题
plt.title('评论人数与好评率的关系')
plt.xlabel('评论人数')
plt.ylabel('好评率')

# 显示图形
plt.show()
```

奇怪，经过筛选后，发现两者的关系又呈现负向关系了。反之，如果只看评论人数超过55人的那些APP，绘图结果，又呈现正向关系。

结语

OK，这期关于豌豆荚市场APP的探索性分析就讲到这里，希望对从事数据分析的网友有所帮助，主要是如何从清洗数据到实现数据的可视化。如果你有问题，欢迎在公众号的留言区域表达你的疑问。同时，也欢迎各位朋友继续转发与分享文中的内容，让跟多的人学习和操作。最后，本文相关的Python脚本和PDF版本已存放到百度云盘，可以通过下面的链接获取：

链接: https://pan.baidu.com/s/1geDRddT 密码: cud2