实战案例2：客户消费数据分析

作者： Robin 日期： 2018/05 数据集来源： UCI Machine Learning Repository

1. 案例描述

这是一个包含跨国销售记录的数据集，其中包含从2010年12月1日至2011年12月09日期间的某英国在线零售公司的所有交易记录。该公司销售业务主要为礼品，并且该公司的许多客户都是批发商。

2. 数据集描述

- UCI Machine Learning Repository提供的客户在线消费记录数据集。每行数据表示一条消费记录。
- 数据字典
  - **InvoiceNo**: 6位代码的发票单号（如果以字母`c`开头，表示该消费记录被取消）
  - **StockCode**: 5位产品编号
  - **Description**: 产品描述
  - **Quantity**: 每笔消费记录购买的产品数量
  - **InvoiceDate**: 消费记录产生的时间
  - **UnitPrice**: 产品单价
  - **CustomerID**: 5位客户编号
  - **Country**: 客户所在国家

3. 任务描述

- 比较各国家的客户数
- 比较各国家的成交额
- 统计各国家交易记录的趋势

4. 主要代码解释

- 代码结构

```
lect02_proj
├── data
    ├── online_retail.xlsx    # 数据文件
├── output
    ├── *.csv    # 生成的数据文件
    ├── *.png    # 生成的图片文件
├── main.py      # 主程序
├── lect02_proj_readme.pdf  # 案例讲解文档
```

- **main.py**

数据清洗后，保存清洗结果时使用了`utf-8`的编码方式，因为数据中有特殊字符。

```
def clean_data(data_df):
    ...
    # 保存清洗结果
    cln_data_df.to_csv(CLN_DATA_FILE, index=False, encoding='utf-8')
    ...
```

- **main.py**

`value_counts()`操作可以用于统计指定列中每个类别（这里是`Country`）的个数。注意，产生的结果是`Series`类型的数据，所以在进行后续可视化操作前，要使用`to_frame()`将`Series`类型转换为`DataFrame`类型。另外，为了使用`Seaborn`绘制柱状图，需要保证`DataFrame`的数据每列是一个统计值，所以又做了转置操作（`T`）。

```
def show_customer_stats(data_df):
    ...
    customer_per_country_df = \
        customer_per_country[customer_per_country.index != 'United Kingdom'].to_frame().T
    ...
```

- **main.py**

当使用多个条件对数据进行过滤时，可以使用`~`，`&`，`|`表示非，且，或操作。

```
def show_total_cost_stats(data_df):
    ...
    # 过滤掉"取消"的交易记录，以及'United Kingdom'的数据
    cond1 = ~data_df['InvoiceNo'].str.startswith('C')
    cond2 = data_df['Country'] != 'United Kingdom'
    valid_data_df = data_df[cond1 & cond2].copy()
    ...
```

- **main.py**

对日期数据类型进行操作时，先使用`pd.to_datetime()`将列转换为日期类型，然后可使用`dt.year`获取日期的年份，`dt.month`获取日期的月份等。

```
def show_trend_by_country(data_df):
    ...
    data_df['InvoiceDate'] = pd.to_datetime(data_df['InvoiceDate'])
    data_df['InvoiceYear'] = data_df['InvoiceDate'].dt.year.astype(str)
    data_df['InvoiceMonth'] = data_df['InvoiceDate'].dt.month.astype(str)
    ...
```

- **main.py**

可通过`str.cat()`对多列按字符串进行拼接，通过设定`sep`参数指定拼接的字符。

```
def show_trend_by_country(data_df):
    ...
    data_df['InvoiceYearMonth'] = data_df['InvoiceYear'].str.cat(data_df['InvoiceMonth'], sep='-
')
    ...
```

- **main.py**

可以使用`groupby`完成多列的分组操作，生成的结果是多重索引的数据。如果需要将多重索引拆分到行和列，可使用`unstack()`完成操作。

```
def show_trend_by_country(data_df):
    ...
    month_country_count = data_df.groupby(['InvoiceYearMonth', 'Country'])['StockCode'].count()
    month_country_count_df = month_country_count.unstack()
    ...
```

- **main.py**

Pandas中的日期类型默认是含有时间的，可以通过设置`Period`将datetime类型进行格式化。比如，如果只保留到月，可使用`to_period('M')`。

```
def show_trend_by_country(data_df):
    ...
    month_country_count_df.index = pd.to_datetime(month_country_count_df.index).to_period('M')
    ...
```

5. 案例总结

- 该项目通过使用Pandas完成了对各国客户网上交易记录的数据分析及可视化，巩固并应用了以下知识点:
    - Pandas的常用操作，包括分组、排序及统计
    - Pandas的可视化操作
    - Seaborn的可视化操作
    - Pandas对于日期、字符串类型数据的操作

6. 课后练习

- 尝试使用Pandas中其他的可视化工具（折线图、饼状图等）对该数据集进行深入分析。

参考资料

1. Pandas官方教程
2. 10分钟入门Pandas
3. Seaborn官方教程
4. Kaggle的Seaborn例子
5. DataCamp的Seaborn例子