Wine Terroir Data Processing

Andrew Lemke

This notebook transforms the raw wine terroir data into a usable form for causal inference.

The data

In this example, we have historical data from many wines produced the same way over a series of decades in California. The grapes and regions are recorded along with a selection of common wine attributes. This data is combined with historical weather data distilled to three weather indexes.

Our dataset contains 5 regions. The northern regions all have very similar climate while the climate is hotter in the southern regions.

The wine data

Feature Name	Feature Explanation	Units	If Used
Table.No.	Number of table in the originial report where data were extracted.	-	
Cultivar	Name of cultivar	-	x
Color	Skin color of each cultivar	-	
Recommend	Recommend types: particularly recommended, limited recommended, not recommended	-	x
RecReg	Recommended planting regions for the cultivar	-	
Reg	Actual planting regions for each cultivar	-	x
Harvest date	Grape harvest date of each cultivar in different regions	yy/mm/dd	x
DCY Day of	year for grape harvest date d	Days	
Must.Brix	Total soluble solids of each cultivar in different regions	oBrix	x
Must.Tacid	Must total acid of each cultivar in different regions	g/100cc	x
Must.pH	Must pH of each cultivar in different regions	-	x
Wine.Alcohol	Wine alcohol of each cultivar in different regions	% (v/v)	x
Wine.Facid	Wine fixed acid of each cultivar in different regions (wine fixed acid equals to wine total acid minus volatile acid)	g/100cc	x
Wine.Extract	Wine extract of each cultivar in different regions	g/100cc	
Wine.Tannin	Wine tannin of each cultivar in different regions	g/100cc	x
Wine.Tasting	Wine tasting notes (evaluate the cultivars based on the merits and defects of wine)		x

The Weather Data

We have one weather dataset for each of the 5 wine growing regions in California. Each dataset has the following features.

Feature Name	Feature Explanation	Units	If Used
GST	Growing season average temperature from April to October	deg C	x
WI	Winkler index	deg C * d	x
HI	Huglin index	deg C * d	x

Imports

Pandas and a regex library are needed to import the data. A regex library is used to extract usable data from descriptions of the wines.

In [1]:
```
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
```

The data is loaded from an online source. The data origionally comes from here.

Here, we load all of the regional weather data and perform some processing.

In [2]:
```
region_dataframes = []

region_nums = [1, 2, 3, 4, 5]

for region_num in region_nums:
    region_name = 'Region_' + str(region_num)
    region_data = pd.read_excel(
        r'https://gitlab.com/Lemke/colab-data/-/raw/02bbb076dc364989414edc1a1161d8b92a3a7a08/WineData/Subset3.xls',
        sheet_name = region_name,
        usecols = [8, 9, 10, 11]
    )
    region_data.dropna(axis=0, inplace=True)  # drop empty rows, an artifact of import
    region_data.rename(columns={'Year.1': 'Year'}, inplace=True)  # fixing another artifact of import
    region_data = region_data.astype({'Year': 'int'})  # make years ints

    region_data['Region'] = [region_num] * len(region_data)  # add region number for concating them all later

    region_dataframes.append(region_data)
```

In [3]:
```
# combine the list of dataframes into one dataframe that has all the info
all_regions = pd.concat(
    region_dataframes,
    axis=0,
    ignore_index=True,
)
```

In [4]:
```
all_regions.head()
```

Out[4]:

	Year	GST	WI	HI	Region
0	1911	15.073364	1099.050000	967.254052	1
1	1912	15.752336	1246.550000	1146.450272	1
2	1913	16.722430	1446.300000	1276.779404	1
3	1914	15.678579	1217.966001	1103.276910	1
4	1915	15.950196	1277.291993	1133.617078	1

Wine Data

The wine data requires a bit more work as it is more complex. Processing will be done on this dataframe, the weather and wine dataframes will be merged, then more processing will occur.

In [5]:
```
wine_data = pd.read_excel(
    r'https://gitlab.com/Lemke/colab-data/-/raw/02bbb076dc364989414edc1a1161d8b92a3a7a08/WineData/Subset1.xls',
    sheet_name = 'data',
)
```

In [6]:
```
# drop cols we don't use
wine_data = wine_data.drop(columns=['RecReg', 'Wine.Extract', 'DOY'])
# drop rows where data is missing
wine_data = wine_data.dropna()
# extract the year from the date feild
wine_data['year'] = wine_data['Harvest date'].dt.year
```

In [7]:
```
wine_data.head()
```

Out[7]:

	Table.No.	Cultivar	Color	Recommend	Reg	Harvest date	Must.Brix	Must.Tacid	Must.pH	Wine.Alcohol	Wine.Facid	Wine.Tannin	Wine.Tasting	year
0	9	Chardony	White	Particularly recommended	3	1936-09-08	27.2	0.54	3.88	15.6	0.42	0.03	Very distinct; good, but alcoholic	1936
2	9	Chardony	White	Particularly recommended	1	1937-10-01	22.1	0.72	3.38	12.0	0.52	0.03	Light body; fruity; distinct; good quality	1937
3	9	Chardony	White	Particularly recommended	3	1937-09-11	24.6	0.63	3.69	12.8	0.49	0.03	Flat; tends to oxidize; but distinct	1937
4	9	Chardony	White	Particularly recommended	1	1938-09-14	19.8	0.78	3.19	10.5	0.68	0.07	Tart, but palatable; distinct; picked too early	1938
5	9	Chardony	White	Particularly recommended	3	1938-09-25	25.2	0.62	3.34	14.8	0.59	0.06	Soft; very rich; distinct; very good	1938

As the weather data is per region per year, we will use this to make a key for merging the tables.

In [8]:
```
wine_data['RegionYearKey'] = [''.join(str(i)) for i in zip(wine_data['Reg'], wine_data['year'])]
all_regions['RegionYearKey'] = [''.join(str(i)) for i in zip(all_regions['Region'], all_regions['Year'])]
```

In [9]:
```
data = wine_data.merge(
    right=all_regions,
    how='left',
    on='RegionYearKey'
)
```

In [10]:
```
data[data['Reg'] != data['Region']].head()
```

Out[10]:

	Table.No.	Cultivar	Color	Recommend	Reg	Harvest date	Must.Brix	Must.Tacid	Must.pH	Wine.Alcohol	Wine.Facid	Wine.Tannin	Wine.Tasting	year	RegionYearKey	Year	GST	WI	HI	Region
42	10	Cabernet Sauvignon	Red	Particularly recommended	5	1937-08-27	22.9	0.65	3.48	11.2	0.49	0.11	Flat; little varietal aroma	1937	(5, 1937)	NaN	NaN	NaN	NaN	NaN
124	11	Semillon	White	Particularly recommended	5	1937-08-27	23.2	0.68	3.89	11.7	0.50	0.03	Distinct flavor; thin; not balanced	1937	(5, 1937)	NaN	NaN	NaN	NaN	NaN
165	12	Petite Sirah	Red	Particularly recommended	5	1936-08-23	23.6	0.61	3.99	11.9	0.45	0.21	Rich, but flat; note very early harvest. Desse...	1936	(5, 1936)	NaN	NaN	NaN	NaN	NaN
170	12	Petite Sirah	Red	Particularly recommended	5	1937-09-16	23.1	0.52	3.64	12.6	0.52	0.23	Fruity; astringent; heavy and coarse	1937	(5, 1937)	NaN	NaN	NaN	NaN	NaN
207	13	Muscat Canelli	White	Particularly recommended	5	1936-08-13	25.1	0.54	3.92	16.8	0.32	0.05	Distinct muscat; rich, luscious flavor; good	1936	(5, 1936)	NaN	NaN	NaN	NaN	NaN

Region 5 is missing some data, so we will drop these rows and then reset the default integer index for the dataframe to clean up.

In [11]:
```
data = data.dropna()
data.reset_index(
    drop=True,
    inplace=True
)
data.shape
```

Out[11]:
```
(1135, 20)
```

In [12]:
```
# verify all the rest of the merged rows match the origional tables
no_error = True
for i in data.index:
    check_data = data.loc[i, ['year', 'Reg', 'GST', 'WI', 'HI']]
    yr, rg = check_data[0:2]
    #print('merged:\n{}'.format(check_data[2:]))
    #print()

    #print(all_regions)
    reg_data = all_regions.loc[(all_regions['Region'] == rg) & (all_regions['Year'] == yr), ['GST', 'WI', 'HI']]
    if (reg_data == check_data[2:]).sum().sum() != 3:
        print('error')
        no_error = False

if no_error:
    print('check complete')
```

```
check complete
```

In [13]:
```
# merging gave us duplicate region and year columns, remove them
data.drop(
    columns=['year', 'Region'],
    inplace=True
)
```

Volatile acid in wine is the fixed acid minus the total acid. We will compute this feature.

In [14]:
```
data['Vacid'] = data['Must.Tacid'] - data['Wine.Facid']
```

We have two outcomes of importance, the wine quality and the "flatness" of the wine. The first of these is provided, the second we will extract from a regex of the tasting notes. Generally winemakers want to avoid flat

wines.

In [15]:
```
flat_regex = re.compile(r'[Ff]lat')
# 1 for flat 0 for not flat
data['Flat'] = data['Wine.Tasting'].apply(lambda x: bool(flat_regex.search(x))).astype(int)
```

In [16]:
```
# remove entries without our outcome of interest
data = data[(data['Recommend'] != 'Not fully tested')]

# transfer outcome from text tables to numerical catagories
data['RecommendNum'] = data['Recommend'].replace(
    to_replace={
        'Particularly recommended': 3,
        'Limited recommendation': 2,
        'Not recommended': 1,
        'Table grape': 0
    },
    inplace=False
)

# ignore warning since the replace operation is only used to get its return value, not change entries
```

In [17]:
```
# simplify column names
data = data.rename(
    mapper={
        'Wine.Alcohol': 'Alcohol',
        'Wine.Facid': 'Facid',
        'Must.Brix': 'Brix',
        'Must.pH': 'pH',
        'Wine.Tannin': 'Tannin',
    },
    axis='columns',

)
```

In [18]:
```
# add a column denoting the hotter region.

data['Outsider'] = (data['Reg'] == 4).astype(int)
# ignore warning since we are not using the right had side to set values, just get a return value

# get regions both as integers and strings
data['RegNum'] = data['Reg'].copy()
data['Reg'] = data['Reg'].map(str)
```

In [20]:
```
data.reset_index(inplace=True, drop=True)
data.to_pickle(r'../data/processed_wine_terroir.pickle')
```

In [ ]:
```

```