Objective here is to understand and learn some effective strategies for visualizing data especially when the number of dimensions start to increase

A quick refresher on visualization

Structured data typically consists of data observations represented by rows and features or data attributes represented by columns. Each column can also be called as a specific dimension of the dataset. Most common data types include continuous, numeric data and discrete, categorical data. Hence any data visualization will basically depict one or more data attributes in an easy to understand visual like a scatter plot, histogram, box-plot and so on. We will cover both *univariate* (one-dimension) and *multivariate* (multi-dimensional) data visualization strategies. We will be using the Python machine learning eco-system here and we recommend you to check out frameworks for data analysis and visualization including `pandas`, `matplotlib`, `seaborn`,

plotly and bokeh . Besides this, knowing about D3.js is also a must if you are interested in crafting beautiful and meaningful visualizations with data. Interested readers are recommended to read '*The Visual Display of Quantitative Information' by* Edward Tufte.

Talk is cheap, show me the visualizations (and code)!

Let's get cracking instead of me droning on about theory and concepts. We will use the Wine Quality Data Set available from the UCI Machine Learning Repository. This data actually consists of two datasets depicting various attributes of red and white variants of the Portuguese *"Vinho Verde"* wine. All the analyses in this article is available in my GitHub repository as a Jupyter Notebook for those of you itching to try it out yourself!

We'll start by loading up the following necessary dependencies for our analyses.

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib as mpl
import numpy as np
import seaborn as sns

%matplotlib inline
```

We will mainly be using matplotlib and seaborn as our visualization frameworks here but you are free to check out and try the same with any other framework of your choice. Let's take a look at the data after some basic data pre-processing steps.

```
1   white_wine = pd.read_csv('winequality-white.csv', sep=';')
2   red_wine = pd.read_csv('winequality-red.csv', sep=';')
3
4   # store wine type as an attribute
5   red_wine['wine_type'] = 'red'
6   white_wine['wine_type'] = 'white'
7
8   # bucket wine quality scores into qualitative quality labels
9   red_wine['quality_label'] = red_wine['quality'].apply(lambda value: 'low'
10                                                  if value <= 5 else 'medium'
11                                                  if value <= 7 else 'high')
12  red_wine['quality_label'] = pd.Categorical(red_wine['quality_label'],
13                                     categories=['low', 'medium', 'high'])
14  white_wine['quality_label'] = white_wine['quality'].apply(lambda value: 'low'
15                                                  if value <= 5 else 'medium'
```

```
16                                          if value <= 7 else 'high')
17    white_wine['quality_label'] = pd.Categorical(white_wine['quality_label'],
18                                       categories=['low', 'medium', 'high'])
19
20    # merge red and white wine datasets
21    wines = pd.concat([red_wine, white_wine])
22
23    # re-shuffle records just to randomize data points
24    wines = wines.sample(frac=1, random_state=42).reset_index(drop=True)
```

effective data viz 1 py hosted with   by **GitHub**   view raw

We create a single data frame wines by merging both the datasets pertaining to red and white wine samples. We also create a new categorical variable quality_label based on the quality attribute of wine samples. Let's take a peek at the data now.

```
wines.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	wine_type	quality_label
0	7.0	0.17	0.74	12.8	0.045	24.0	126.0	0.99420	3.26	0.38	12.2	8	white	high
1	7.7	0.64	0.21	2.2	0.077	32.0	133.0	0.99560	3.27	0.45	9.9	5	red	low
2	6.8	0.39	0.34	7.4	0.020	38.0	133.0	0.99212	3.18	0.44	12.0	7	white	medium
3	6.3	0.28	0.47	11.2	0.040	61.0	183.0	0.99592	3.12	0.51	9.5	6	white	medium
4	7.4	0.35	0.20	13.9	0.054	63.0	229.0	0.99888	3.11	0.50	8.9	6	white	medium

The wine quality dataset

It is quite evident that we have several numeric and categorical attributes for wine samples. Each observation belongs to a red or white wine sample and the attributes are specific attributes or properties measured and obtained from physicochemical tests. You can check out the Jupyter notebook if you want to understand the detailed explanation of each attribute but the names are pretty self-explanatory. Let's do a quick basic descriptive summary statistics on some of these attributes of interest.

```
1    subset_attributes = ['residual sugar', 'total sulfur dioxide', 'sulphates',
2                          'alcohol', 'volatile acidity', 'quality']
3    rs = round(red_wine[subset_attributes].describe(),2)
4    ws = round(white_wine[subset_attributes].describe(),2)
5
6    pd.concat([rs, ws], axis=1, keys=['Red Wine Statistics', 'White Wine Statistics'])
```

effective_data_viz_2.py hosted with 🖤 by **GitHub**   view raw

Red Wine Statistics

White Wine Statistics

	residual sugar	total sulfur dioxide	sulphates	alcohol	volatile acidity	quality	residual sugar	total sulfur dioxide	sulphates	alcohol	volatile acidity	quality
count	1599.00	1599.00	1599.00	1599.00	1599.00	1599.00	4898.00	4898.00	4898.00	4898.00	4898.00	4898.00
mean	2.54	46.47	0.66	10.42	0.53	5.64	6.39	138.36	0.49	10.51	0.28	5.88
std	1.41	32.90	0.17	1.07	0.18	0.81	5.07	42.50	0.11	1.23	0.10	0.89
min	0.90	6.00	0.33	8.40	0.12	3.00	0.60	9.00	0.22	8.00	0.08	3.00
25%	1.90	22.00	0.55	9.50	0.39	5.00	1.70	108.00	0.41	9.50	0.21	5.00
50%	2.20	38.00	0.62	10.20	0.52	6.00	5.20	134.00	0.47	10.40	0.26	6.00
75%	2.60	62.00	0.73	11.10	0.64	6.00	9.90	167.00	0.55	11.40	0.32	6.00
max	15.50	289.00	2.00	14.90	1.58	8.00	65.80	440.00	1.08	14.20	1.10	9.00

Basic descriptive statistics by wine type

It's quite easy to contrast and compare these statistical measures for the different types of wine samples. Notice the stark difference in some of the attributes. We will emphasize those in some of our visualizations later on.

Univariate Analysis

Univariate analysis is basically the simplest form of data analysis or visualization where we are only concerned with analyzing one data attribute or variable and visualizing the same (one dimension).
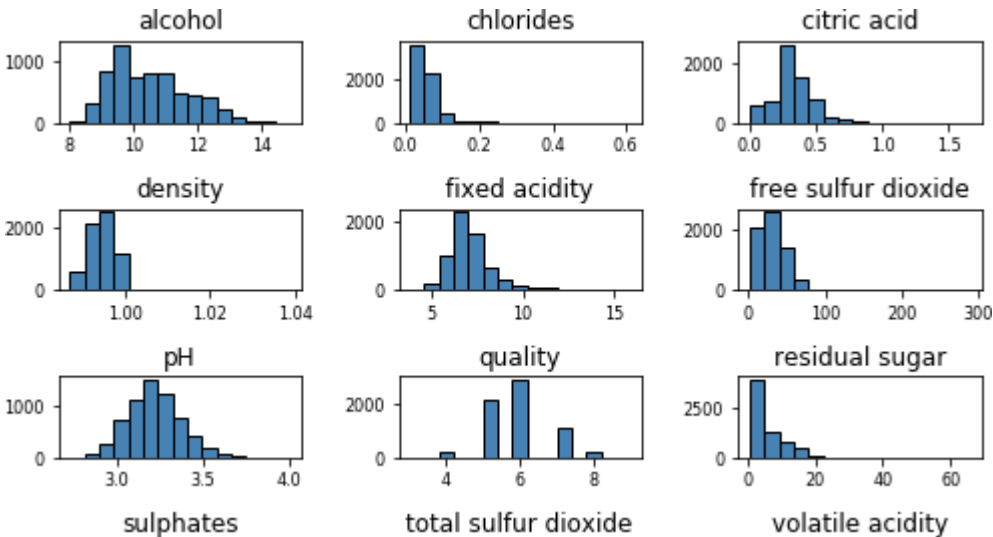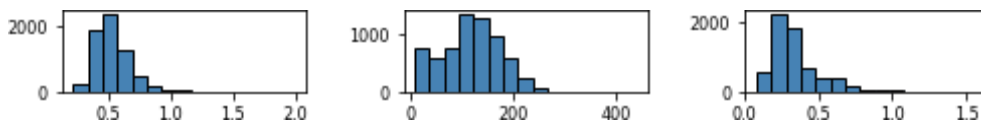
Visualizing data in One Dimension (1-D)

One of the quickest and most effective ways to visualize all numeric data and their distributions, is to leverage *histograms* using `pandas`

```
1    wines.hist(bins=15, color='steelblue', edgecolor='black', linewidth=1.0,
2            xlabelsize=8, ylabelsize=8, grid=False)
3    plt.tight_layout(rect=(0, 0, 1.2, 1.2))
```

effective_data_viz_3.py hosted with 🖤 by **GitHub**   view raw

Visualizing attributes as one-dimensional data

The plots above give a good idea about the basic data distribution of any of the attributes.

Let's drill down to *visualizing one of the continuous, numeric attributes.* Essentially a *histogram* or a *density plot* works quite well in understanding how the data is distributed for that attribute.
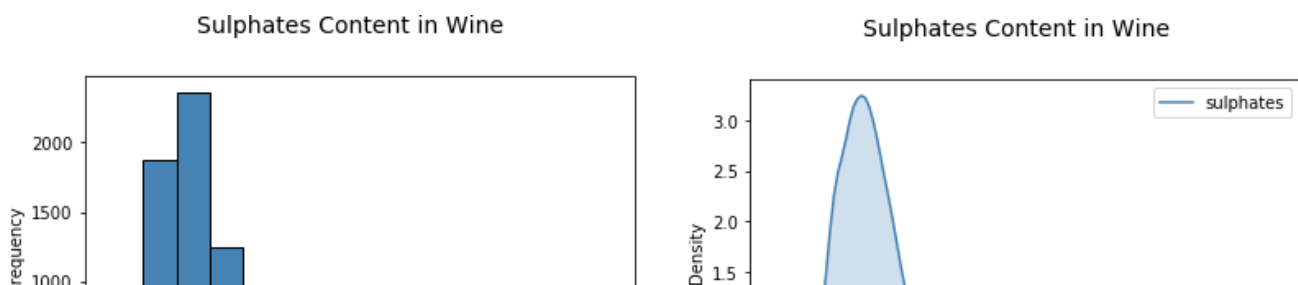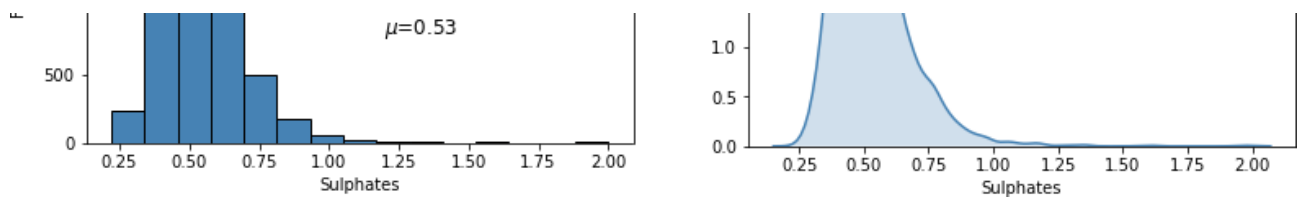
```python
1    # Histogram
2    fig = plt.figure(figsize = (6,4))
3    title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
4    fig.subplots_adjust(top=0.85, wspace=0.3)
5
6    ax = fig.add_subplot(1,1, 1)
7    ax.set_xlabel("Sulphates")
8    ax.set_ylabel("Frequency")
9    ax.text(1.2, 800, r'$\mu$='+str(round(wines['sulphates'].mean(),2)),
10            fontsize=12)
11   freq, bins, patches = ax.hist(wines['sulphates'], color='steelblue', bins=15,
12                                  edgecolor='black', linewidth=1)
13
14
15   # Density Plot
16   fig = plt.figure(figsize = (6, 4))
17   title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
18   fig.subplots_adjust(top=0.85, wspace=0.3)
19
20   ax1 = fig.add_subplot(1,1, 1)
21   ax1.set_xlabel("Sulphates")
22   ax1.set_ylabel("Frequency")
23   sns.kdeplot(wines['sulphates'], ax=ax1, shade=True, color='steelblue')
24
```

effective_data_viz_4.py hosted with   by **GitHub**   view raw

Visualizing one-dimensional continuous, numeric data

It is quite evident from the above plot that there is a definite right skew in the distribution for wine `sulphates` .

Visualizing a discrete, categorical data attribute is slightly different and *bar plots* are one of the most effective ways to do the same. You can use *pie-charts* also but in general try avoiding them altogether, especially when the number of distinct categories is more than three.

```python
1    # Bar Plot
2    fig = plt.figure(figsize = (6, 4))
3    title = fig.suptitle("Wine Quality Frequency", fontsize=14)
4    fig.subplots_adjust(top=0.85, wspace=0.3)
5
6    ax = fig.add_subplot(1,1, 1)
7    ax.set_xlabel("Quality")
8    ax.set_ylabel("Frequency")
9    w_q = wines['quality'].value_counts()
10   w_q = (list(w_q.index), list(w_q.values))
11   ax.tick_params(axis='both', which='major', labelsize=8.5)
12   bar = ax.bar(w_q[0], w_q[1], color='steelblue',
13            edgecolor='black', linewidth=1)
```

effective_data_viz_5.py hosted with 💚 by GitHub   view raw
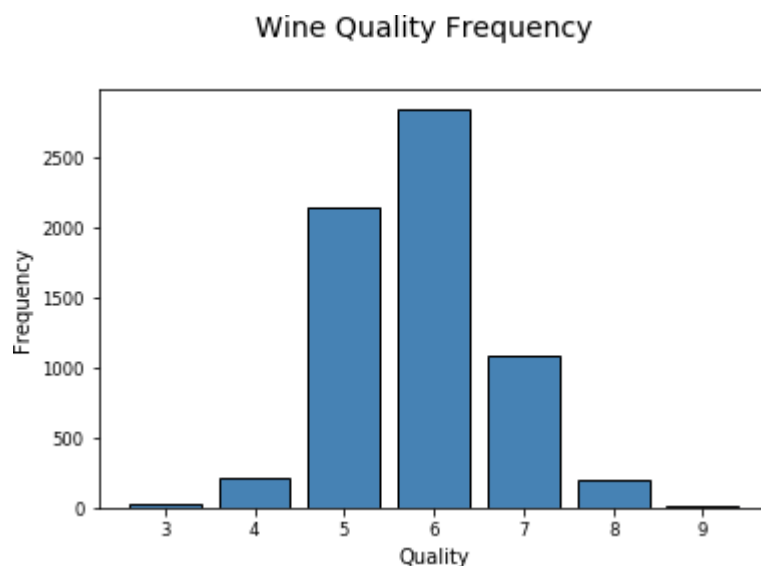
Visualizing one-dimensional discrete, categorical data

Let's move on to looking at higher dimensional data now.

Multivariate Analysis

Multivariate analysis is where the fun as well as the complexity begins. Here we analyze multiple data dimensions or attributes (2 or more). Multivariate analysis not only involves just checking out distributions but also potential relationships, patterns and correlations amongst these attributes. You can also leverage inferential statistics and hypothesis testing if necessary based on the problem to be solved at hand to check out statistical significance for different attributes, groups and so on.

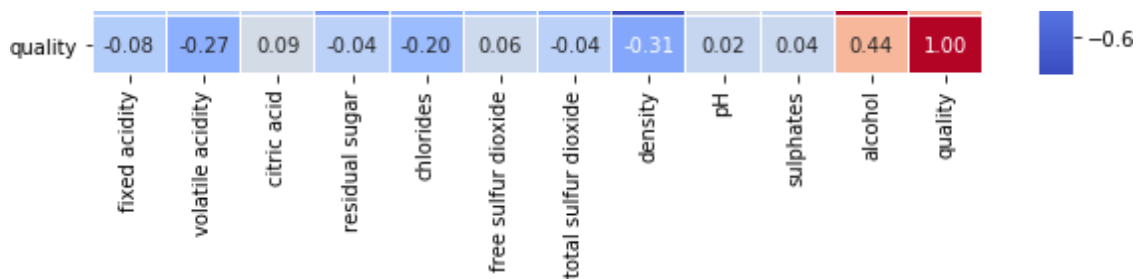## Visualizing data in Two Dimensions (2-D)

One of the best ways to check out potential relationships or correlations amongst the different data attributes is to leverage a *pair-wise correlation matrix* and depict it as a *heatmap*.

```python
# Correlation Matrix Heatmap
f, ax = plt.subplots(figsize=(10, 6))
corr = wines.corr()
hm = sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                 linewidths=.05)
f.subplots_adjust(top=0.93)
t= f.suptitle('Wine Attributes Correlation Heatmap', fontsize=14)
```

effective_data_viz_6.py hosted with 💚 by **GitHub**   view raw

Wine Attributes Correlation Heatmap
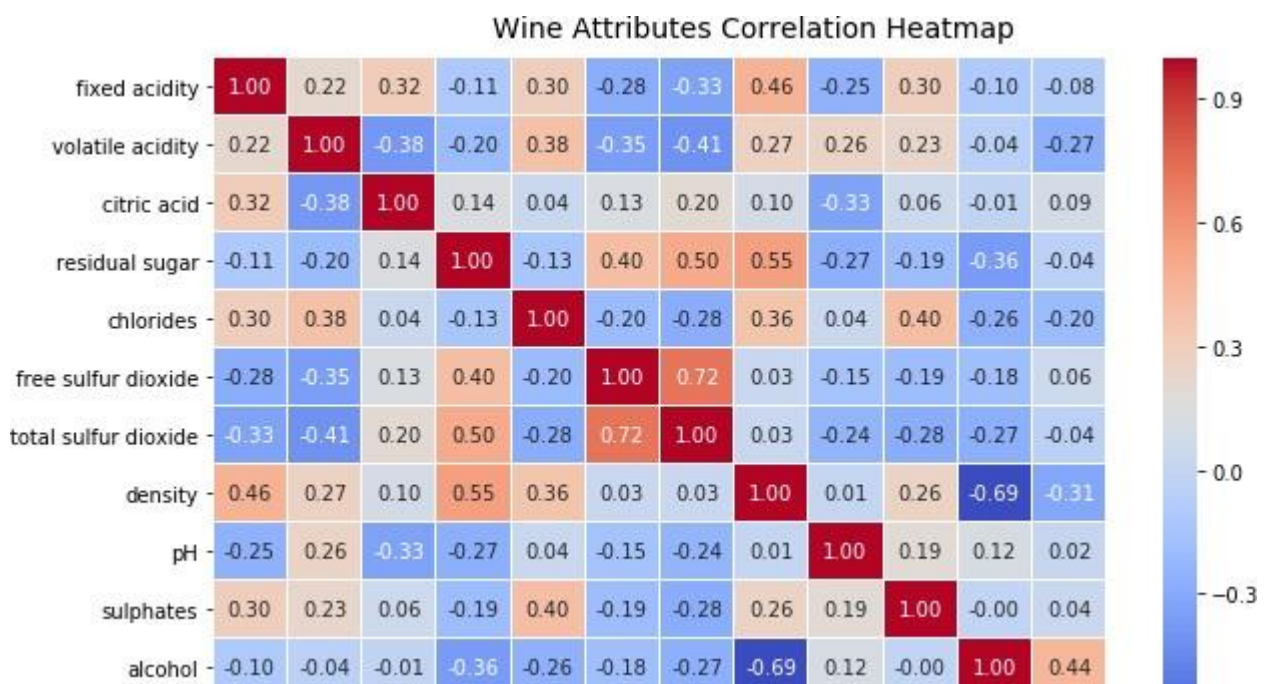
Visualizing two-dimensional data with a correlation heatmap

The gradients in the heatmap vary based on the strength of the correlation and you can clearly see it is very easy to spot potential attributes having strong correlations amongst themselves. Another way to visualize the same is to use *pair-wise scatter plots* amongst attributes of interest.

```python
# Pair-wise Scatter Plots
cols = ['density', 'residual sugar', 'total sulfur dioxide', 'fixed acidity']
pp = sns.pairplot(wines[cols], size=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

fig = pp.fig
fig.subplots_adjust(top=0.93, wspace=0.3)
t = fig.suptitle('Wine Attributes Pairwise Plots', fontsize=14)
```

effective_data_viz_7.py hosted with ♥ by GitHub   view raw
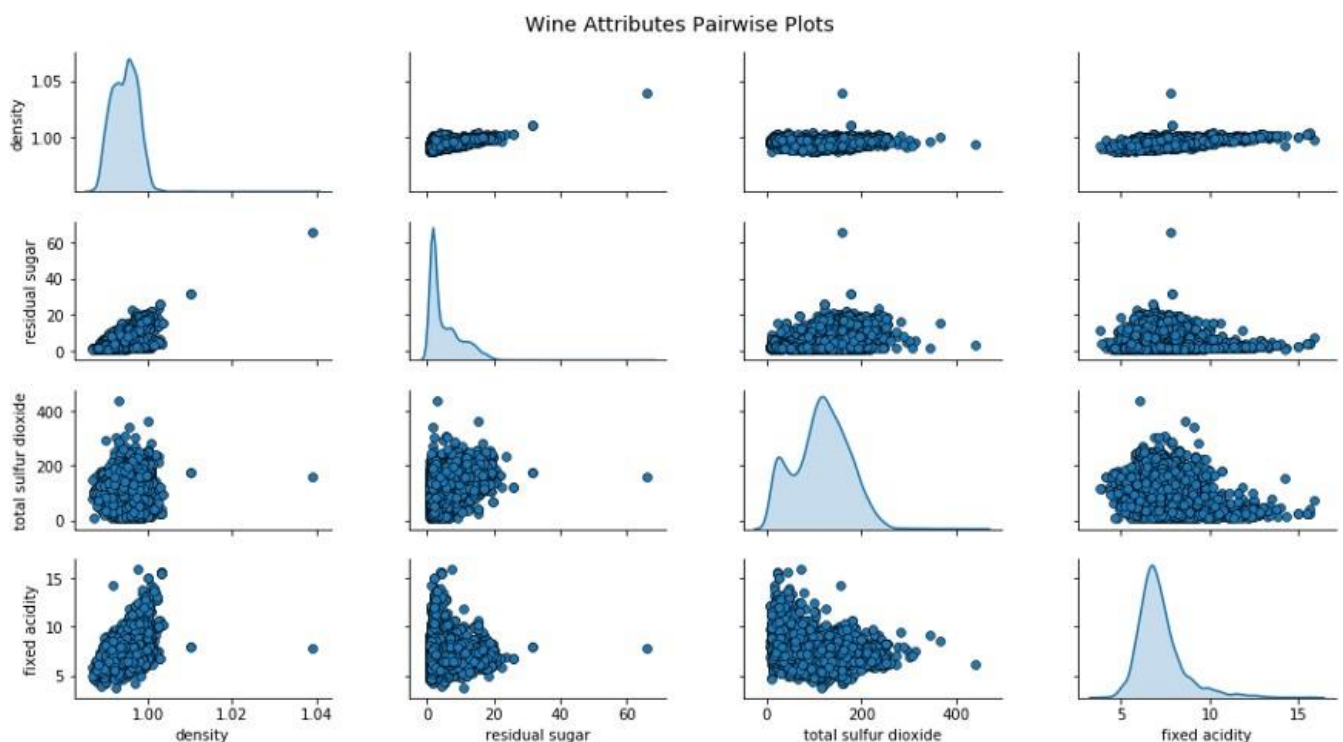
Visualizing two-dimensional data with pair-wise scatter plots

Based on the above plot, you can see that scatter plots are also a decent way of observing potential relationships or patterns in two-dimensions for data attributes.

> *An important point to note about pairwise scatter plots is that the plots are actually symmetric. The scatterplot for any pair of attributes (X, Y) looks different from the same attributes in (Y, X) only because the vertical and horizontal scales are different. It does not contain any new information.*

Another way of visualizing multivariate data for multiple attributes together is to use *parallel coordinates*.

```
1    # Scaling attribute values to avoid few outiers
2    cols = ['density', 'residual sugar', 'total sulfur dioxide', 'fixed acidity']
3    subset_df = wines[cols]
4
5    from sklearn.preprocessing import StandardScaler
6    ss = StandardScaler()
7
8    scaled_df  =  ss.fit_transform(subset_df)
9    scaled_df = pd.DataFrame(scaled_df, columns=cols)
10   final_df = pd.concat([scaled_df, wines['wine_type']], axis=1)
11   final_df.head()
12
13   # plot parallel coordinates
14   from pandas.plotting import parallel_coordinates
15   pc = parallel_coordinates(final_df, 'wine_type', color=('#FFE888', '#FF9999'))
```

effective_data_viz_8.py hosted with 💚 by GitHub   view raw
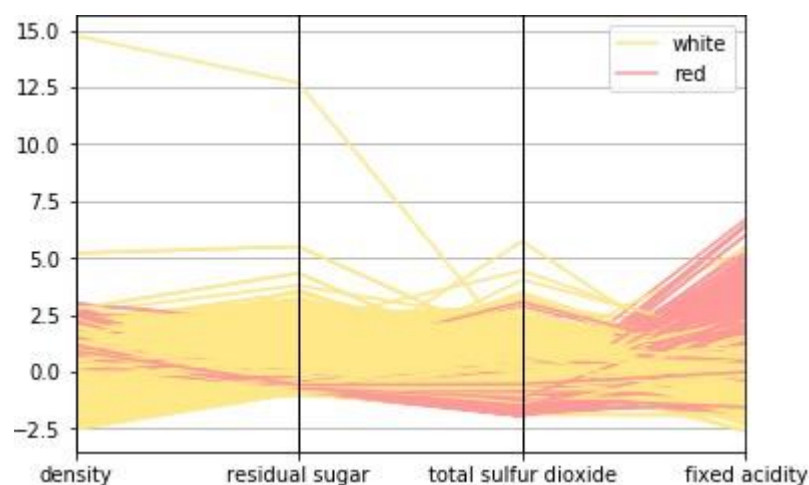
Parallel coordinates to visualize multi-dimensional data

Basically, in this visualization as depicted above, points are represented as connected line segments. Each vertical line represents one data attribute. One complete set of connected line segments across all the attributes represents one data point. Hence points that tend to cluster will appear closer together. Just by looking at it, we can clearly see that density is slightly more for *red wines* as compared to *white wines*. Also residual sugar and total sulfur dioxide is higher for *white wines* as compared to *red* and fixed acidity is higher for *red wines* as compared to white wines. Check out the statistics from the statistic table we derived earlier to validate this assumption!

Let's look at some ways in which we can *visualize two continuous, numeric attributes*. *Scatter plots* and *joint plots* in particular are good ways to not only check for patterns, relationships but also see the individual distributions for the attributes.
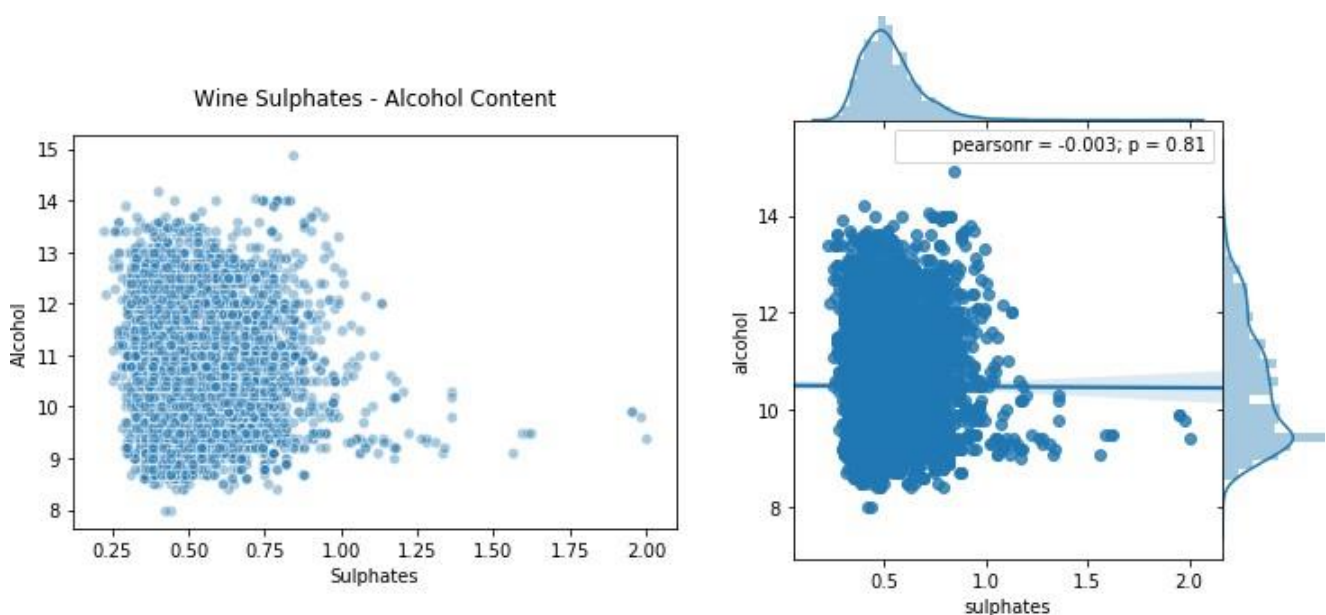
```
1   # Scatter Plot
2   plt.scatter(wines['sulphates'], wines['alcohol'],
3               alpha=0.4, edgecolors='w')
4
5   plt.xlabel('Sulphates')
6   plt.ylabel('Alcohol')
7   plt.title('Wine Sulphates - Alcohol Content',y=1.05)
8
9
10  # Joint Plot
11  jp = sns.jointplot(x='sulphates', y='alcohol', data=wines,
12                     kind='reg', space=0, size=5, ratio=4)
```

effective_data_viz_9.py hosted with 🖤 by **GitHub**   view raw

Visualizing two-dimensional continuous, numeric data using scatter plots and joint plots

The *scatter plot* is depicted on the left side and the *joint plot* on the right in the above figure. Like we mentioned, you can check out correlations, relationships as well as individual distributions in the joint plot.
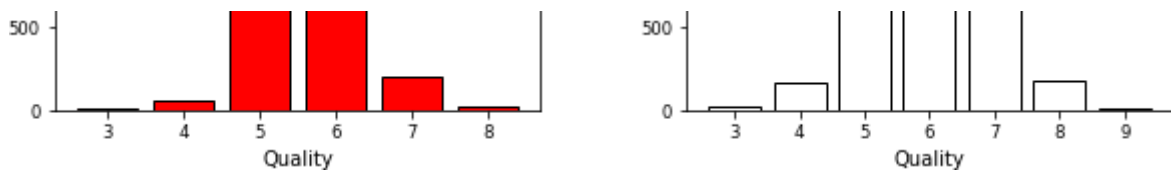
How about *visualizing two discrete, categorical attributes?* One way is to leverage separate plots (subplots) or *facets* for one of the categorical dimensions.

```
1    # Using subplots or facets along with Bar Plots
2    fig = plt.figure(figsize = (10, 4))
3    title = fig.suptitle("Wine Type - Quality", fontsize=14)
4    fig.subplots_adjust(top=0.85, wspace=0.3)
5    # red wine - wine quality
6    ax1 = fig.add_subplot(1,2, 1)
7    ax1.set_title("Red Wine")
8    ax1.set_xlabel("Quality")
9    ax1.set_ylabel("Frequency")
10   rw_q = red_wine['quality'].value_counts()
11   rw_q = (list(rw_q.index), list(rw_q.values))
12   ax1.set_ylim([0, 2500])
13   ax1.tick_params(axis='both', which='major', labelsize=8.5)
14   bar1 = ax1.bar(rw_q[0], rw_q[1], color='red',
15                  edgecolor='black', linewidth=1)
16
17   # white wine - wine quality
18   ax2 = fig.add_subplot(1,2, 2)
19   ax2.set_title("White Wine")
20   ax2.set_xlabel("Quality")
21   ax2.set_ylabel("Frequency")
22   ww_q = white_wine['quality'].value_counts()
23   ww_q = (list(ww_q.index), list(ww_q.values))
24   ax2.set_ylim([0, 2500])
25   ax2.tick_params(axis='both', which='major', labelsize=8.5)
26   bar2 = ax2.bar(ww_q[0], ww_q[1], color='white',
27                  edgecolor='black', linewidth=1)
```
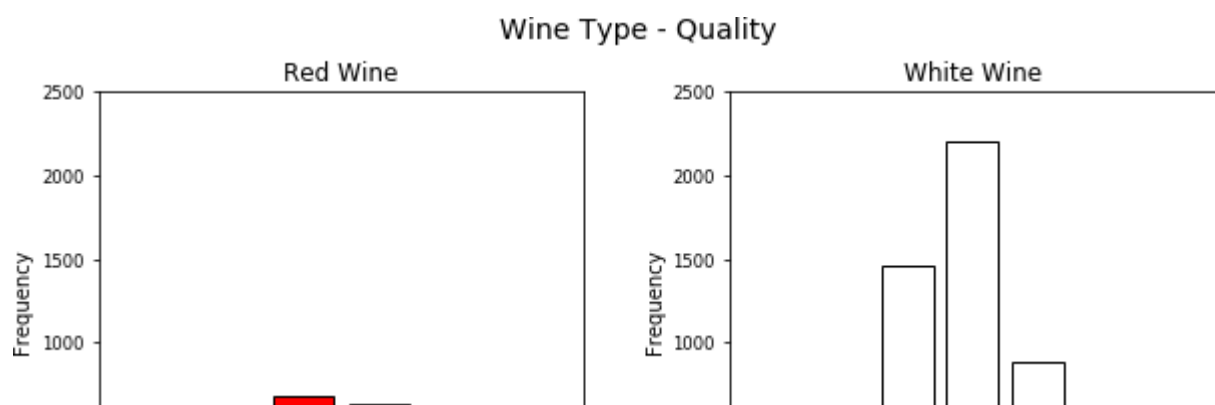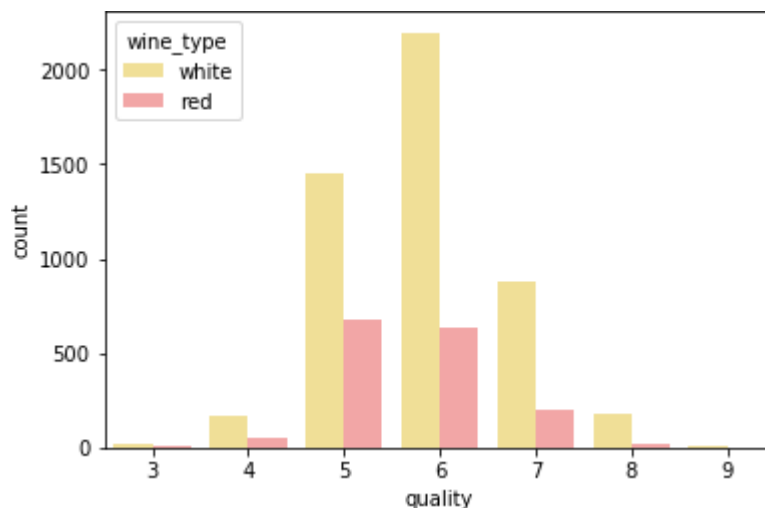
Visualizing two-dimensional discrete, categorical data using bar plots and subplots (facets)

While this is a good way to visualize categorical data, as you can see, leveraging matplotlib has resulted in writing a lot of code. Another good way is to use *stacked bars* or *multiple bars* for the different attributes in a single plot. We can leverage seaborn for the same easily.

```
1    # Multi-bar Plot
2    cp = sns.countplot(x="quality", hue="wine_type", data=wines,
3                       palette={"red": "#FF9999", "white": "#FFE888"})
```

effective_data_viz_11.py hosted with ♥ by **GitHub**   view raw

Visualizing two-dimensional discrete, categorical data in a single bar chart

This definitely looks cleaner and you can also effectively compare the different categories easily from this single plot.
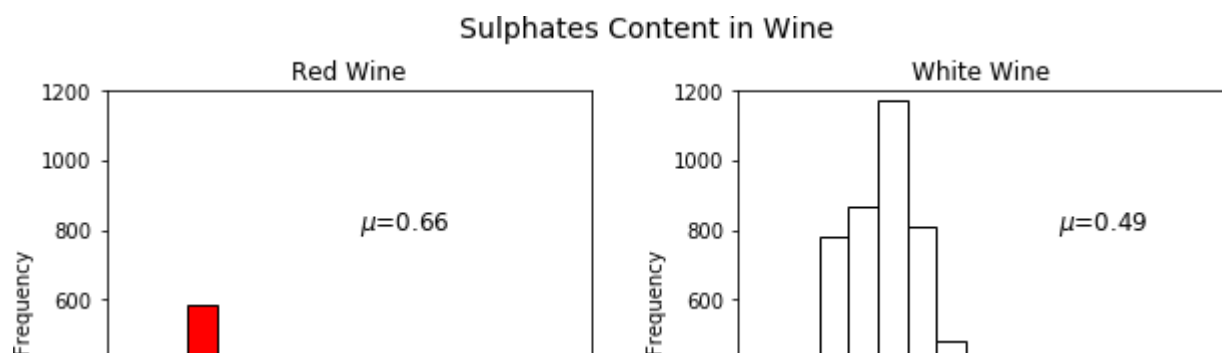
Let's look at *visualizing mixed attributes in two-dimensions* (essentially numeric and categorical together). One way is to use *faceting\subplots* along with generic *histograms* or *density plots*.
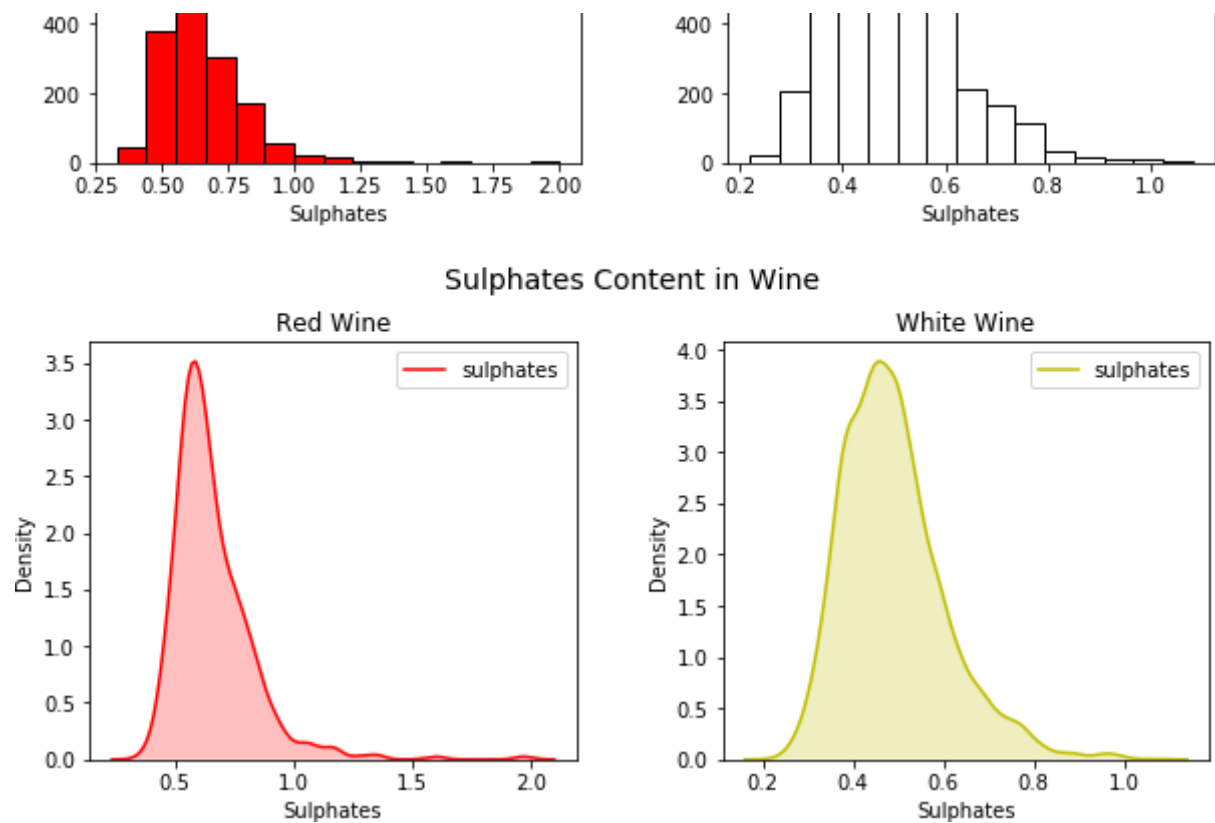
```
1    # facets with histograms
2    fig = plt.figure(figsize = (10,4))
3    title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
4    fig.subplots_adjust(top=0.85, wspace=0.3)
```

```
 5
 6    ax1 = fig.add_subplot(1,2, 1)
 7    ax1.set_title("Red Wine")
 8    ax1.set_xlabel("Sulphates")
 9    ax1.set_ylabel("Frequency")
10    ax1.set_ylim([0, 1200])
11    ax1.text(1.2, 800, r'$\mu$='+str(round(red_wine['sulphates'].mean(),2)),
12            fontsize=12)
13    r_freq, r_bins, r_patches = ax1.hist(red_wine['sulphates'], color='red', bins=15,
14                                    edgecolor='black', linewidth=1)
15
16    ax2 = fig.add_subplot(1,2, 2)
17    ax2.set_title("White Wine")
18    ax2.set_xlabel("Sulphates")
19    ax2.set_ylabel("Frequency")
20    ax2.set_ylim([0, 1200])
21    ax2.text(0.8, 800, r'$\mu$='+str(round(white_wine['sulphates'].mean(),2)),
22            fontsize=12)
23    w_freq, w_bins, w_patches = ax2.hist(white_wine['sulphates'], color='white', bins=15,
24                                    edgecolor='black', linewidth=1)
25
26
27    # facets with density plots
28    fig = plt.figure(figsize = (10, 4))
29    title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
30    fig.subplots_adjust(top=0.85, wspace=0.3)
31
32    ax1 = fig.add_subplot(1,2, 1)
33    ax1.set_title("Red Wine")
34    ax1.set_xlabel("Sulphates")
35    ax1.set_ylabel("Density")
36    sns.kdeplot(red_wine['sulphates'], ax=ax1, shade=True, color='r')
37
38    ax2 = fig.add_subplot(1,2, 2)
39    ax2.set_title("White Wine")
40    ax2.set_xlabel("Sulphates")
41    ax2.set_ylabel("Density")
42    sns.kdeplot(white_wine['sulphates'], ax=ax2, shade=True, color='y')
```


Sulphates Content in Wine

Visualizing mixed attributes in two-dimensions leveraging facets and histograms\density plots

While this is good, once again we have a lot of boilerplate code which we can avoid by leveraging seaborn and even depict the plots in one single chart.

```python
1   # Using multiple Histograms
2   fig = plt.figure(figsize = (6, 4))
3   title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
4   fig.subplots_adjust(top=0.85, wspace=0.3)
5   ax = fig.add_subplot(1,1, 1)
6   ax.set_xlabel("Sulphates")
7   ax.set_ylabel("Frequency")
8
9   g = sns.FacetGrid(wines, hue='wine_type', palette={"red": "r", "white": "y"})
10  g.map(sns.distplot, 'sulphates', kde=False, bins=15, ax=ax)
11  ax.legend(title='Wine Type')
12  plt.close(2)
```

effective_data_viz_13.py hosted with 🖤 by **GitHub**   view raw
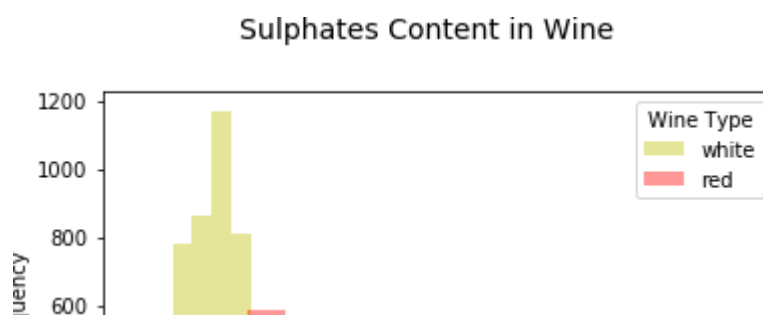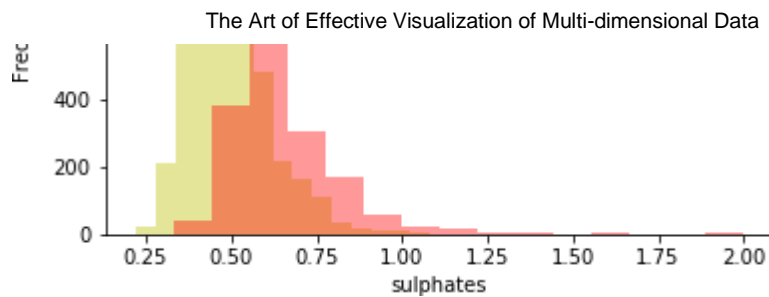
Leveraging multiple histograms for mixed attributes in two-dimensions

You can see the plot generated above is clear and concise and we can easily compare across the distributions easily. Besides this, *box plots* are another way of effectively depicting groups of numeric data based on the different values in the categorical attribute. *Box plots* are a good way to know the quartile values in the data and also potential outliers.
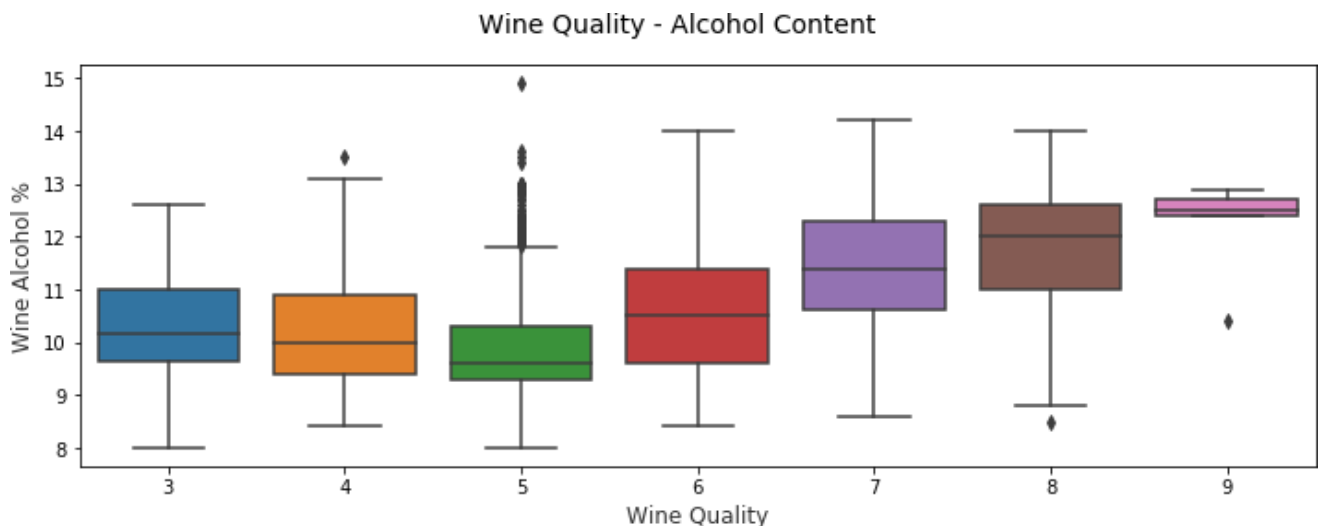
```python
# Box Plots
f, (ax) = plt.subplots(1, 1, figsize=(12, 4))
f.suptitle('Wine Quality - Alcohol Content', fontsize=14)

sns.boxplot(x="quality", y="alcohol", data=wines,  ax=ax)
ax.set_xlabel("Wine Quality",size = 12,alpha=0.8)
ax.set_ylabel("Wine Alcohol %",size = 12,alpha=0.8)
```

effective_data_viz_14.py hosted with 🖤 by **GitHub**   view raw

Box Plots as an effective representation of two-dimensional mixed attributes

Another similar visualization is *violin plots*, which are another effective way to visualize grouped numeric data using kernel density plots (depicts probability density of the data at different values).

```python
# Violin Plots
```
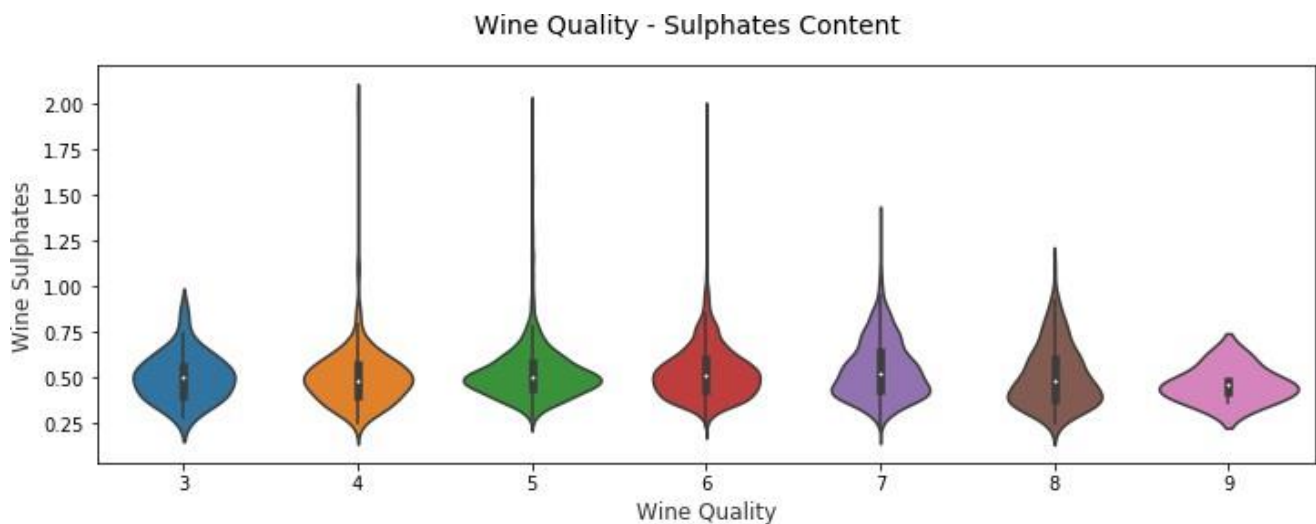
```
2    f, (ax) = plt.subplots(1, 1, figsize=(12, 4))
3    f.suptitle('Wine Quality - Sulphates Content', fontsize=14)
4
5    sns.violinplot(x="quality", y="sulphates", data=wines,  ax=ax)
6    ax.set_xlabel("Wine Quality",size = 12,alpha=0.8)
7    ax.set_ylabel("Wine Sulphates",size = 12,alpha=0.8)
```

effective_data_viz_15.py hosted with 💚 by **GitHub**   view raw

Violin Plots as an effective representation of two-dimensional mixed attributes

You can clearly see the density plots above for the different wine `quality` categories for wine `sulphate` .

> *Visualizing data till two-dimensions is pretty straightforward but starts becoming complex as the number of dimensions (attributes) start increasing. The reason is because we are bound by the two-dimensions of our display mediums and our environment.*
>
> *For three-dimensional data, we can introduce a fake notion of depth by taking a z-axis in our chart or leveraging subplots and facets.*
>
> *However for data higher than three-dimensions, it becomes even more difficult to visualize the same. The best way to go higher than three dimensions is to use plot facets, color, shapes, sizes, depth and so on. You can also use time as a dimension by making an animated plot for other attributes over time (considering time is a dimension in the data). Check out Hans Roslin's excellent talk to get an idea of the same!*

Visualizing data in Three Dimensions (3-D)

Considering three attributes or dimensions in the data, we can visualize them by considering a *pair-wise scatter plot* and introducing the notion of *color* or *hue* to

separate out values in a categorical dimension.

```
1    # Scatter Plot with Hue for visualizing data in 3-D
2    cols = ['density', 'residual sugar', 'total sulfur dioxide', 'fixed acidity', 'wine_type']
3    pp = sns.pairplot(wines[cols], hue='wine_type', size=1.8, aspect=1.8,
4    palette={"red": "#FF9999", "white": "#FFE888"},
5                      plot_kws=dict(edgecolor="black", linewidth=0.5))
6    fig = pp.fig
7    fig.subplots_adjust(top=0.93, wspace=0.3)
8    t = fig.suptitle('Wine Attributes Pairwise Plots', fontsize=14)
```

effective_data_viz_16.py hosted with 🖤 by GitHub   view raw
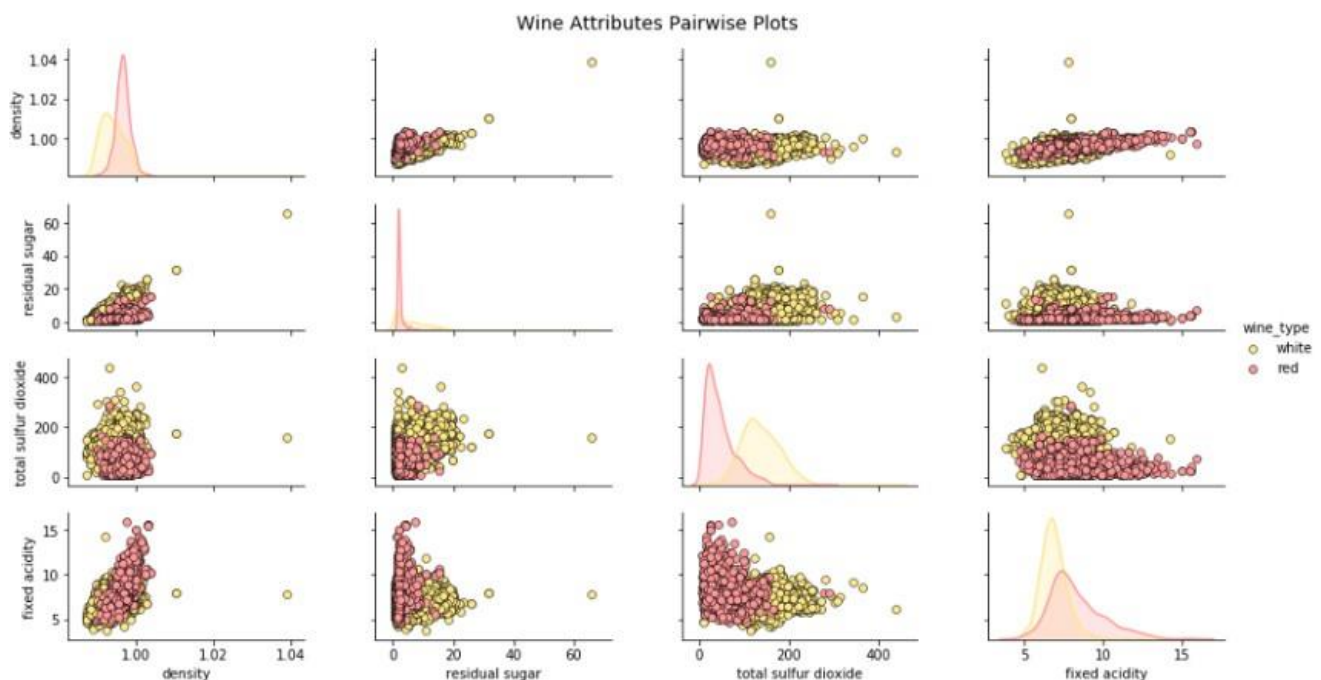
Visualizing three-dimensional data with scatter plots and hue (color)

The above plot enables you to check out correlations and patterns and also compare around wine groups. Like we can clearly see total sulfur dioxide and residual sugar is higher for *white wine* as compared to *red*.

Let's look at strategies for *visualizing three continuous, numeric attributes*. One way would be to have two dimensions represented as the regular *length* (*x*-axis)and *breadth* (*y*-axis) and also take the notion of *depth* (*z*-axis) for the third dimension.

```
1    # Visualizing 3-D numeric data with Scatter Plots
2    # length, breadth and depth
3    fig = plt.figure(figsize=(8, 6))
4    ax = fig.add_subplot(111, projection='3d')
5
```
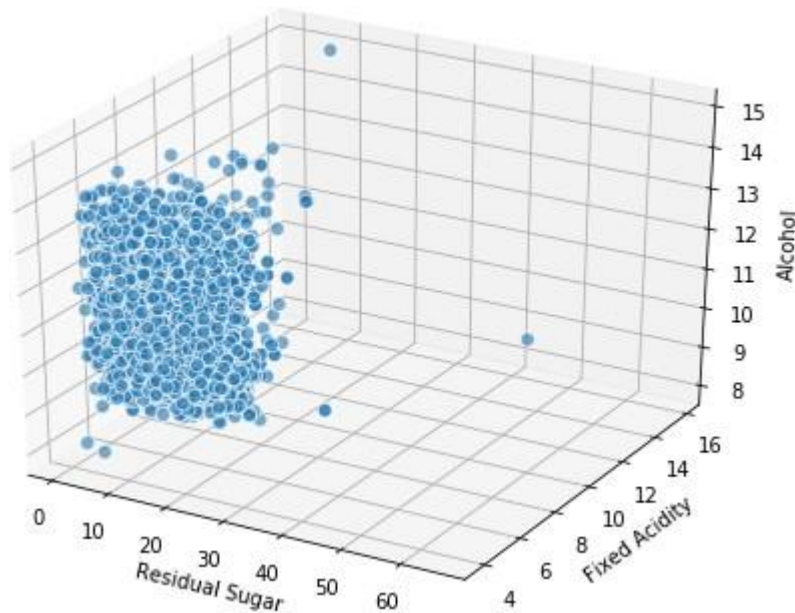
```
6   xs = wines['residual sugar']
7   ys = wines['fixed acidity']
8   zs = wines['alcohol']
9   ax.scatter(xs, ys, zs, s=50, alpha=0.6, edgecolors='w')
10
11  ax.set_xlabel('Residual Sugar')
12  ax.set_ylabel('Fixed Acidity')
13  ax.set_zlabel('Alcohol')
```

effective_data_viz_17.py hosted with 🦑 by **GitHub**   view raw

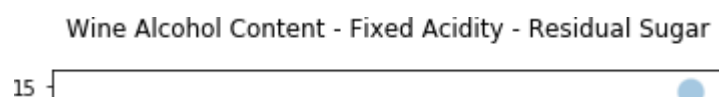Visualizing three-dimensional numeric data by introducing the notion of depth

We can also still leverage the regular 2-D axes and introduce the notion of *size* as the third dimension (essentially a *bubble chart*) where the size of the dots indicate the quantity of the third dimension.

```
1   # Visualizing 3-D numeric data with a bubble chart
2   # length, breadth and size
3   plt.scatter(wines['fixed acidity'], wines['alcohol'], s=wines['residual sugar']*25,
4               alpha=0.4, edgecolors='w')
5
6   plt.xlabel('Fixed Acidity')
7   plt.ylabel('Alcohol')
8   plt.title('Wine Alcohol Content - Fixed Acidity - Residual Sugar',y=1.05)
```

effective_data_viz_18.py hosted with 🦑 by **GitHub**   view raw

Wine Alcohol Content - Fixed Acidity - Residual Sugar
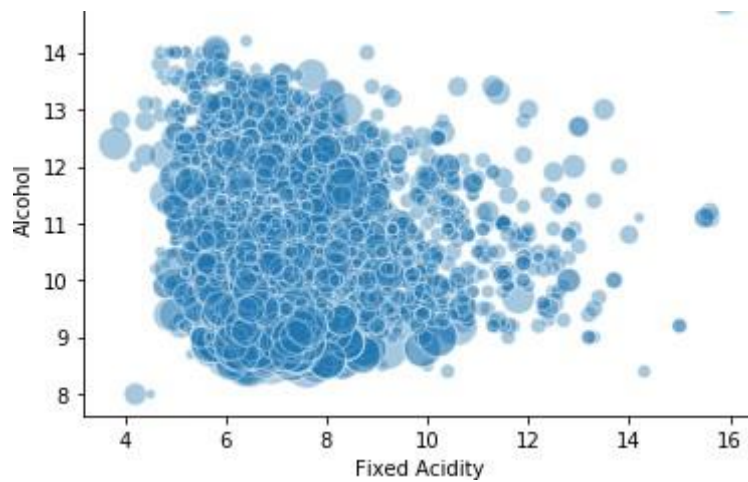
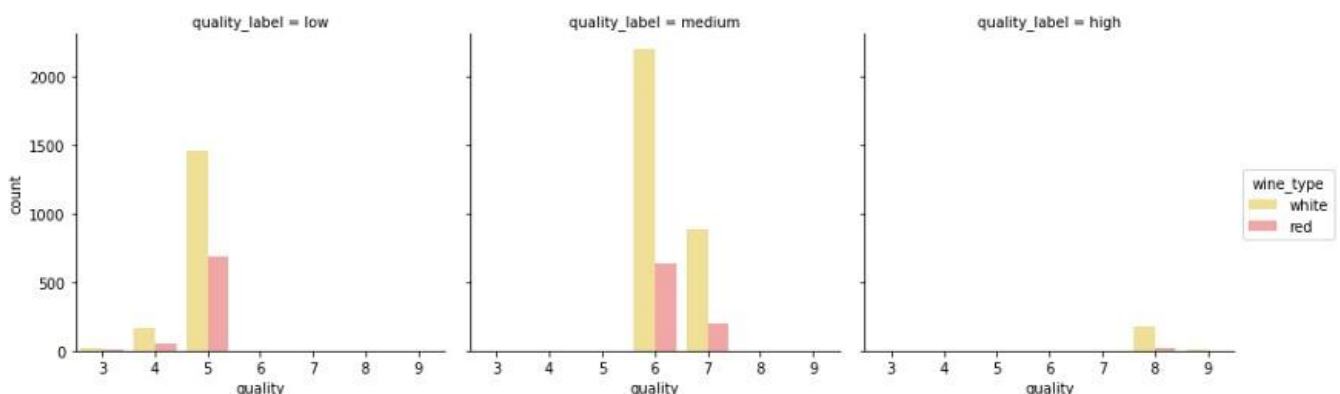Visualizing three-dimensional numeric data by introducing the notion of size

Thus you can see how the chart above is not a conventional scatter plot but more of a bubble chart with varying point sizes (bubbles) based on the quantity of residual sugar . Of course its not always that you will find definite patterns in the data like in this case, we see varying sizes across the other two dimensions.

For *visualizing three discrete, categorical attributes,* while we can use the conventional *bar plots*, we can leverage the notion of *hue* as well as *facets* or *subplots* to support the additional third dimension. The seaborn framework helps us keep the code to a minimum and plot this effectively.

```
1    # Visualizing 3-D categorical data using bar plots
2    # leveraging the concepts of hue and facets
3    fc = sns.factorplot(x="quality", hue="wine_type", col="quality_label",
4                        data=wines, kind="count",
5                        palette={"red": "#FF9999", "white": "#FFE888"})
```

effective_data_viz_19.py hosted with ♥ by GitHub   view raw

Visualizing three-dimensional categorical data by introducing the notion of hue and facets

The chart above clearly shows the frequency pertaining to each of the dimensions and you can see how easy and effective this can be in understanding relevant insights.

Considering visualization for *three mixed attributes,* we can use the notion of *hue* for separating our groups in one of the categorical attributes while using conventional visualizations like *scatter plots* for visualizing two dimensions for numeric attributes.
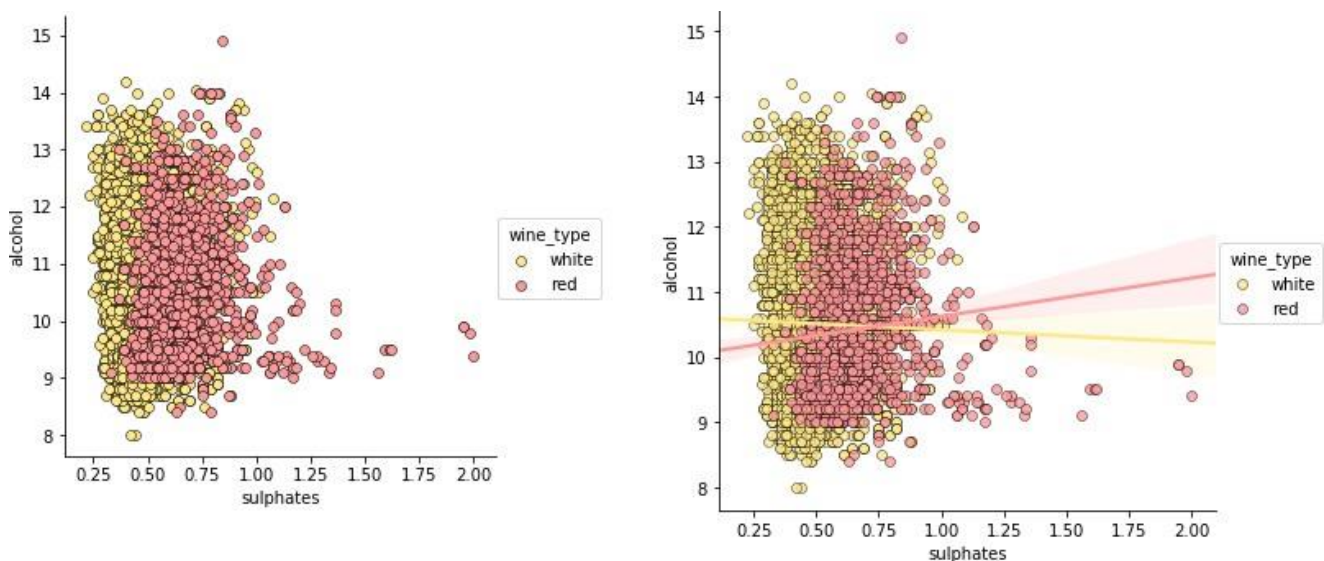
```
1    # Visualizing 3-D mix data using scatter plots
2    # leveraging the concepts of hue for categorical dimension
3    jp = sns.pairplot(wines, x_vars=["sulphates"], y_vars=["alcohol"], size=4.5,
4                      hue="wine_type", palette={"red": "#FF9999", "white": "#FFE888"},
5                      plot_kws=dict(edgecolor="k", linewidth=0.5))
6
7    # we can also view relationships\correlations as needed
8    lp = sns.lmplot(x='sulphates', y='alcohol', hue='wine_type',
9                    palette={"red": "#FF9999", "white": "#FFE888"},
10                   data=wines, fit_reg=True, legend=True,
11                   scatter_kws=dict(edgecolor="k", linewidth=0.5))
```

effective_data_viz_20.py hosted with ♥ by **GitHub**   view raw

Visualizing mixed attributes in three-dimensions leveraging scatter plots and the concept of hue

Thus hue acts as a good separator for the categories or groups and while there is no or very weak correlation as observed above, we can still understand from these plots that sulphates are slightly higher for *red wines* as compared to *white*. Instead of a scatter plot, you can also use a *kernel density plot* to understand the data in three dimensions.

```
1    # Visualizing 3-D mix data using kernel density plots
2    # leveraging the concepts of hue for categorical dimension
```
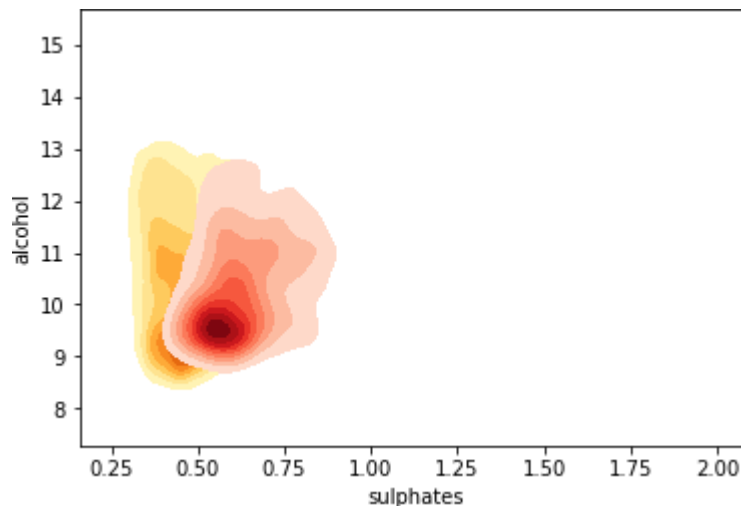
```
3   ax = sns.kdeplot(white_wine[ sulphates ], white_wine[ alcohol ],
4                    cmap="YlOrBr", shade=True, shade_lowest=False)
5   ax = sns.kdeplot(red_wine['sulphates'], red_wine['alcohol'],
6                    cmap="Reds", shade=True, shade_lowest=False)
```

effective_data_viz_21.py hosted with 🖤 by **GitHub**   view raw

Visualizing mixed attributes in three-dimensions leveraging kernel density plots and the concept of hue

It is quite evident and expected that *red wine* samples have higher sulphate levels as compared to *white wines.* You can also see the density concentrations based on the hue intensity.

In case we are dealing with *more than one categorical attribute in the three dimensions,* we can use *hue* and *one of the regular axes* for visualizing data and use visualizations like *box plots* or *violin plots* to visualize the different groups of data.

```
1   # Visualizing 3-D mix data using violin plots
2   # leveraging the concepts of hue and axes for > 1 categorical dimensions
3   f, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))
4   f.suptitle('Wine Type - Quality - Acidity', fontsize=14)
5
6   sns.violinplot(x="quality", y="volatile acidity",
7                  data=wines, inner="quart", linewidth=1.3,ax=ax1)
8   ax1.set_xlabel("Wine Quality",size = 12,alpha=0.8)
9   ax1.set_ylabel("Wine Volatile Acidity",size = 12,alpha=0.8)
10
11  sns.violinplot(x="quality", y="volatile acidity", hue="wine_type",
12                 data=wines, split=True, inner="quart", linewidth=1.3,
13                 palette={"red": "#FF9999", "white": "white"}, ax=ax2)
14  ax2.set_xlabel("Wine Quality",size = 12,alpha=0.8)
15  ax2.set_ylabel("Wine Volatile Acidity",size = 12,alpha=0.8)
16  l = plt.legend(loc='upper right', title='Wine Type')
```
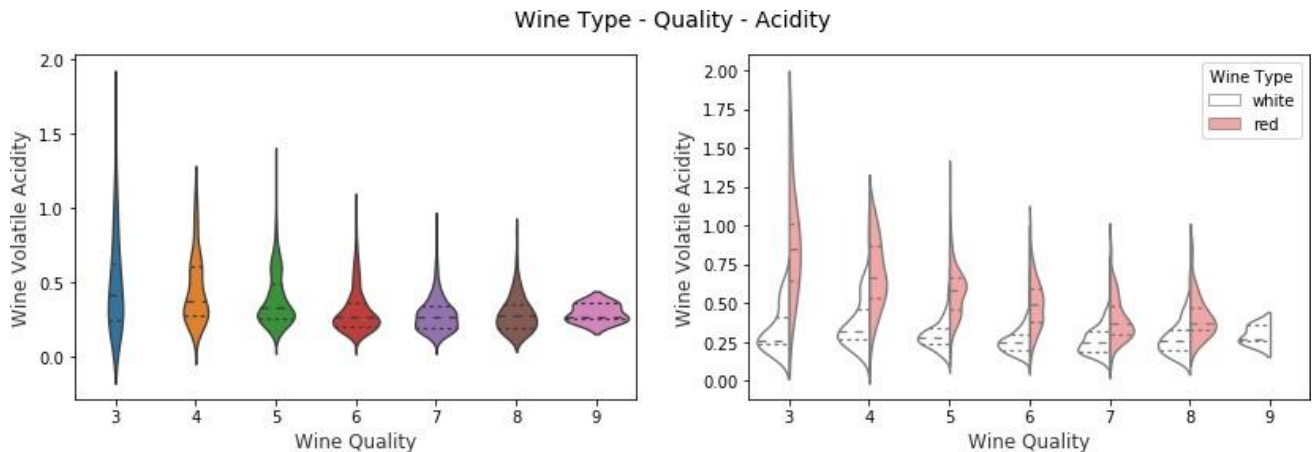
Visualizing mixed attributes in three-dimensions leveraging split violin plots and the concept of hue

In the figure above, we can see that in the 3-D visualization on the right hand plot, we have represented wine quality on the x-axis and wine_type as the *hue*. We can clearly see some interesting insights like volatile acidity is higher for *red wines* as compared to *white wines*.

You can also consider using *box plots* for representing mixed attributes with more than one categorical variable in a similar way.

```
1    # Visualizing 3-D mix data using box plots
2    # leveraging the concepts of hue and axes for > 1 categorical dimensions
3    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))
4    f.suptitle('Wine Type - Quality - Alcohol Content', fontsize=14)
5
6    sns.boxplot(x="quality", y="alcohol", hue="wine_type",
7                data=wines, palette={"red": "#FF9999", "white": "white"}, ax=ax1)
8    ax1.set_xlabel("Wine Quality",size = 12,alpha=0.8)
9    ax1.set_ylabel("Wine Alcohol %",size = 12,alpha=0.8)
10
11   sns.boxplot(x="quality_label", y="alcohol", hue="wine_type",
12                data=wines, palette={"red": "#FF9999", "white": "white"}, ax=ax2)
13   ax2.set_xlabel("Wine Quality Class",size = 12,alpha=0.8)
14   ax2.set_ylabel("Wine Alcohol %",size = 12,alpha=0.8)
15   l = plt.legend(loc='best', title='Wine Type')
```

Wine Type - Quality - Alcohol Content
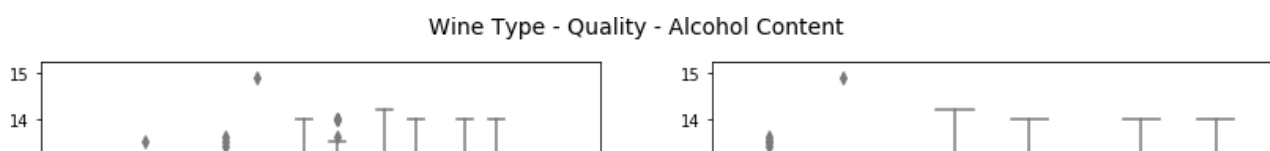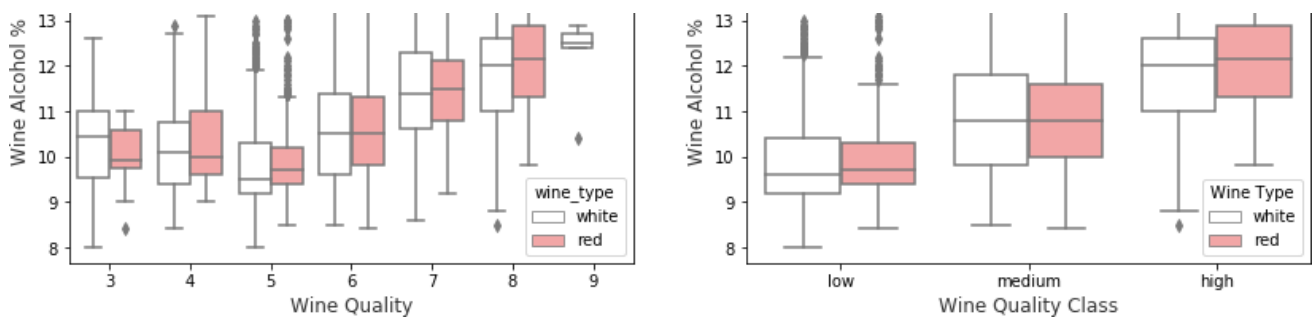
Visualizing mixed attributes in three-dimensions leveraging box plots and the concept of hue

We can see that both for quality and quality_label attributes, the wine alcohol content increases with better quality. Also *red wines* tend to have a sightly higher median alcohol content as compared to *white wines* based on the *quality class*. However if we check the *quality ratings*, we can see that for *lower rated wines* (3 & 4), the *white wine* median alcohol content is greater than *red wine* samples. Otherwise *red wines* seem to have a slightly higher median alcohol content in general as compared to *white wines*.

Visualizing data in Four Dimensions (4-D)

Based on our discussion earlier, we leverage various components of the charts visualize multiple dimensions. One way to visualize data in four dimensions is to use *depth* and *hue* as specific data dimensions in a conventional plot like a *scatter plot*.

```python
1   # Visualizing 4-D mix data using scatter plots
2   # leveraging the concepts of hue and depth
3   fig = plt.figure(figsize=(8, 6))
4   t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Type', fontsize=14)
5   ax = fig.add_subplot(111, projection='3d')
6
7   xs = list(wines['residual sugar'])
8   ys = list(wines['alcohol'])
9   zs = list(wines['fixed acidity'])
10  data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]
11  colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]
12
13  for data, color in zip(data_points, colors):
14      x, y, z = data
15      ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=30)
16
17  ax.set_xlabel('Residual Sugar')
18  ax.set_ylabel('Alcohol')
19  ax.set_zlabel('Fixed Acidity')
```

effective data viz 24.py hosted with 🖤 by **GitHub**   view raw

Wine Residual Sugar - Alcohol Content - Acidity - Type
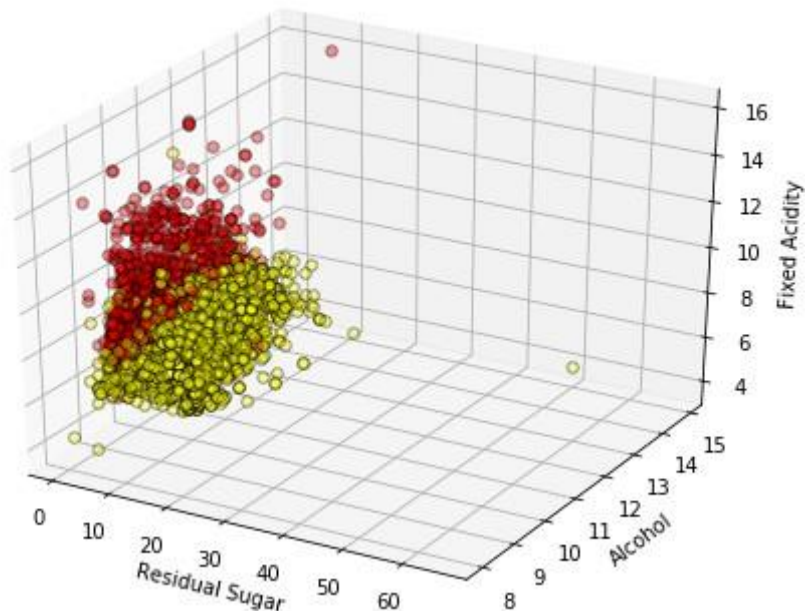
Visualizing data in four-dimensions leveraging scatter plots and the concept of hue and depth

The wine_type attribute is denoted by the hue which is quite evident from the above plot. Also, while interpreting these visualizations start getting difficult due to the complex nature of the plots, you can still gather insights like fixed acidity is higher for *red wines* and residual sugar is higher for *white wines*. Of course if there were some association between alcohol and fixed acidity we might have seen a gradually increasing or decreasing plane of data points showing some trend.

Another strategy is to keep a 2-D plot but use *hue* and data point *size* as data dimensions. Typically this would be a *bubble chart* similar to what we visualized earlier.

```
1   # Visualizing 4-D mix data using bubble plots
2   # leveraging the concepts of hue and size
3   size = wines['residual sugar']*25
4   fill_colors = ['#FF9999' if wt=='red' else '#FFE888' for wt in list(wines['wine_type'])]
5   edge_colors = ['red' if wt=='red' else 'orange' for wt in list(wines['wine_type'])]
6
7   plt.scatter(wines['fixed acidity'], wines['alcohol'], s=size,
8               alpha=0.4, color=fill_colors, edgecolors=edge_colors)
9
10  plt.xlabel('Fixed Acidity')
11  plt.ylabel('Alcohol')
12  plt.title('Wine Alcohol Content - Fixed Acidity - Residual Sugar - Type',y=1.05)
```

effective_data_viz_25.py hosted with ♥ by **GitHub**   view raw
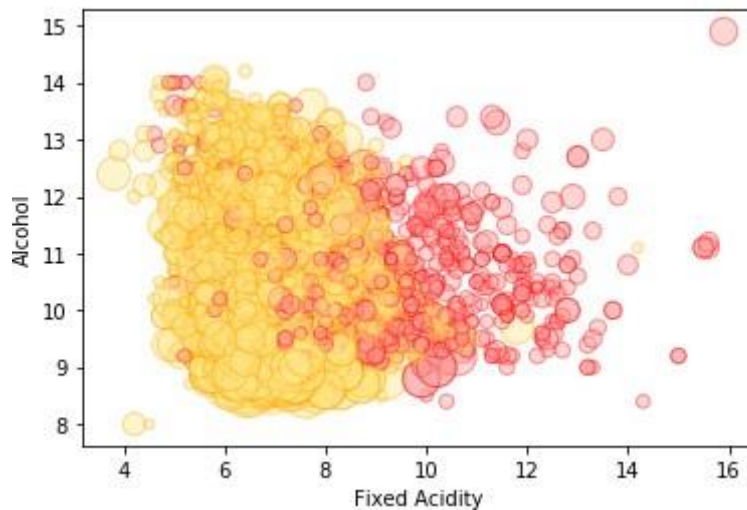
Visualizing data in four-dimensions leveraging bubble charts and the concept of hue and size

We use *hue* to represent wine_type and the data point *size* to represent residual sugar . We do see similar patterns from what we observed in the previous chart and bubble sizes are larger for *white wine* in general indicate residual sugar values are higher for *white wine* as compared to *red*.

If we have more that two categorical attributes to represent, we can reuse our concept of leveraging *hue* and *facets* to depict these attributes and regular plots like *scatter plots* to represent the numeric attributes. Let's look at a couple of examples.
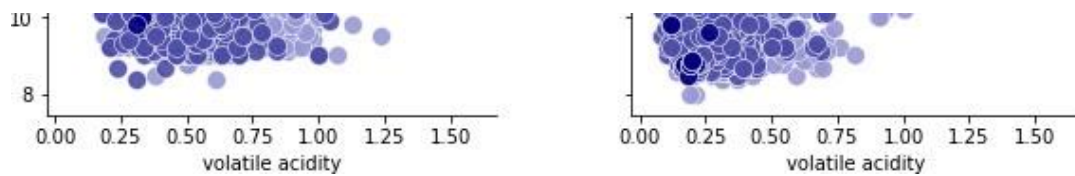
```python
1   # Visualizing 4-D mix data using scatter plots
2   # leveraging the concepts of hue and facets for > 1 categorical attributes
3   g = sns.FacetGrid(wines, col="wine_type", hue='quality_label',
4                     col_order=['red', 'white'], hue_order=['low', 'medium', 'high'],
5                     aspect=1.2, size=3.5, palette=sns.light_palette('navy', 4)[1:])
6   g.map(plt.scatter, "volatile acidity", "alcohol", alpha=0.9,
7         edgecolor='white', linewidth=0.5, s=100)
8   fig = g.fig
9   fig.subplots_adjust(top=0.8, wspace=0.3)
10  fig.suptitle('Wine Type - Alcohol - Quality - Acidity', fontsize=14)
11  l = g.add_legend(title='Wine Quality Class')
```
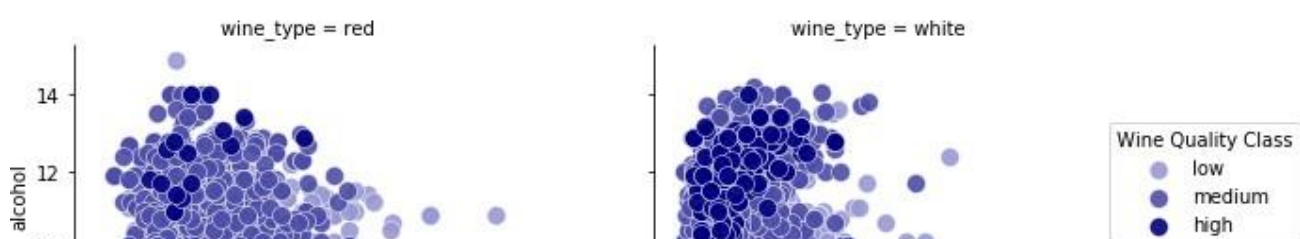
effective_data_viz_26.py hosted with 🖤 by GitHub   view raw

Visualizing data in four-dimensions leveraging scatter plots and the concept of hue and facets

The effectiveness of this visualization is verified by the fact we can easily spot multiple patterns. The $_{volatile\ acidity}$ levels for *white wines* are lower and also *high quality wines* have lower acidity levels. Also based on *white wine* samples, *high quality wines* have higher levels of $_{alcohol}$ and *low quality wines* have the lowest levels of $_{alcohol}$!

Let's take up a similar example with some other attributes and build a visualization in four dimensions.

```
1   # Visualizing 4-D mix data using scatter plots
2   # leveraging the concepts of hue and facets for > 1 categorical attributes
3   g = sns.FacetGrid(wines, col="wine_type", hue='quality_label',
4                     col_order=['red', 'white'], hue_order=['low', 'medium', 'high'],
5                     aspect=1.2, size=3.5, palette=sns.light_palette('green', 4)[1:])
6   g.map(plt.scatter, "volatile acidity", "total sulfur dioxide", alpha=0.9,
7         edgecolor='white', linewidth=0.5, s=100)
8   fig = g.fig
9   fig.subplots_adjust(top=0.8, wspace=0.3)
10  fig.suptitle('Wine Type - Sulfur Dioxide - Acidity - Quality', fontsize=14)
11  l = g.add_legend(title='Wine Quality Class')
```

effective_data_viz_27.py hosted with 🖤 by **GitHub**   view raw
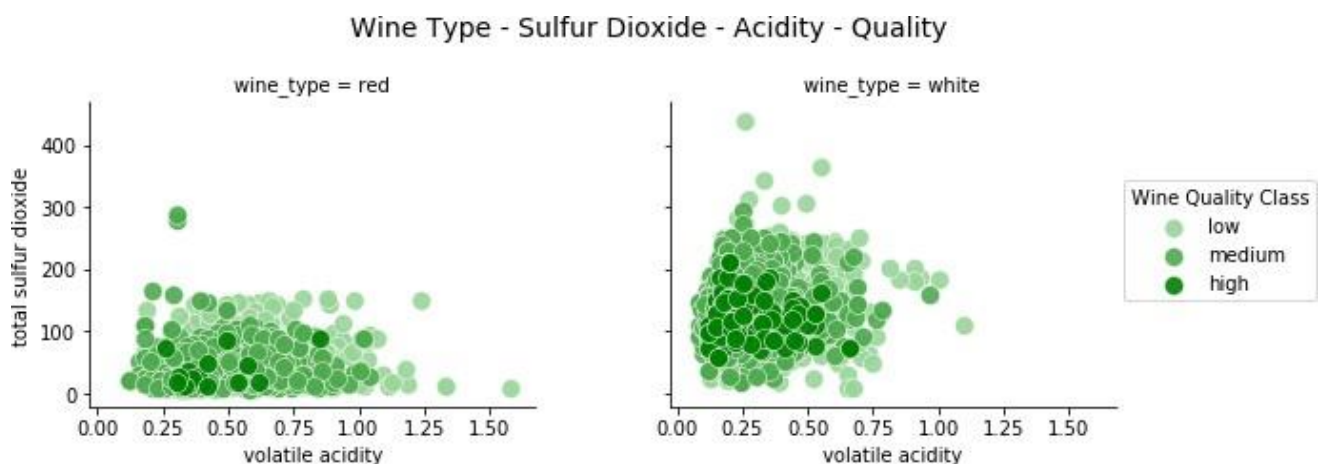
Visualizing data in four-dimensions leveraging scatter plots and the concept of hue and facets

We clearly see that *high quality wines* have lower content of $_{total\ sulfur\ dioxide}$ which is quite relevant if you also have the necessary domain knowledge about wine

composition. We also see that total sulfur dioxide levels for *red wine* are lower than *white wine*. The volatile acidity levels are however higher for *red wines* in several data points.

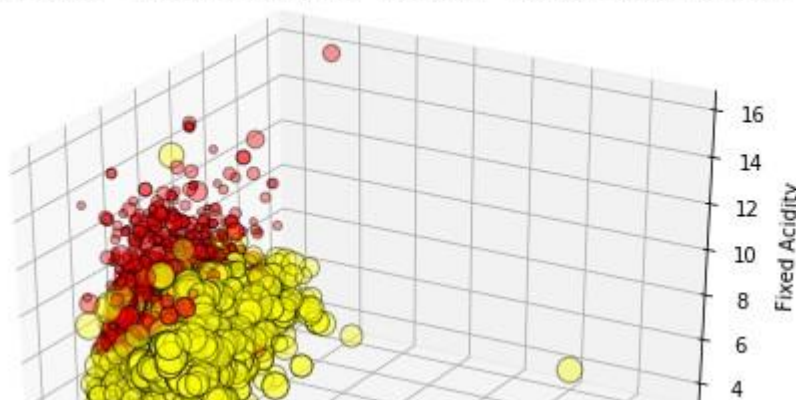## Visualizing data in Five Dimensions (5-D)

Once again following a similar strategy as we followed in the previous section, to visualize data in five dimensions, we leverage various plotting components. Let's use *depth*, *hue* and *size* to represent three of the data dimensions besides *regular axes* representing the other two dimensions. Since we use the notion of size, we will be basically plotting a three dimensional *bubble chart*.

```
1    # Visualizing 5-D mix data using bubble charts
2    # leveraging the concepts of hue, size and depth
3    fig = plt.figure(figsize=(8, 6))
4    ax = fig.add_subplot(111, projection='3d')
5    t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Typ
6
7    xs = list(wines['residual sugar'])
8    ys = list(wines['alcohol'])
9    zs = list(wines['fixed acidity'])
10   data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]
11
12   ss = list(wines['total sulfur dioxide'])
13   colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]
14
15   for data, color, size in zip(data_points, colors, ss):
16       x, y, z = data
17       ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=size)
18
19   ax.set_xlabel('Residual Sugar')
20   ax.set_ylabel('Alcohol')
21   ax.set_zlabel('Fixed Acidity')
```


Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Type
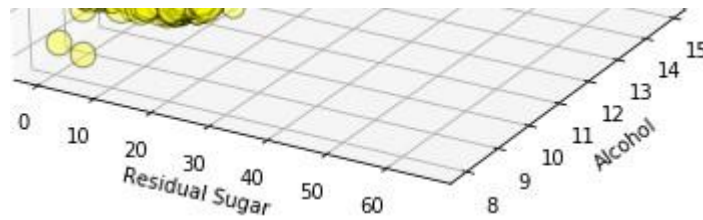
Visualizing data in five-dimensions leveraging bubble charts and the concept of hue, depth and size

This chart depicts the same patterns and insights that we talked about in the previous section. However, we can also see that based on the point sizes which are represented by total sulfur dioxide , *white wines* have higher total sulfur dioxide levels as compared to *red wines*.

Instead of *depth*, we can also use *facets* along with *hue* to represent more than one categorical attribute in these five data dimensions. One of the attributes representing *size* can be *numerical (continuous)* or even *categorical* (but we might need to represent it with numbers for data point sizes). While we don't depict that here due to the lack of categorical attributes, feel free to try it out on your own datasets.

```
1   # Visualizing 5-D mix data using bubble charts
2   # leveraging the concepts of hue, size and facets
3   g = sns.FacetGrid(wines, col="wine_type", hue='quality_label',
4                     col_order=['red', 'white'], hue_order=['low', 'medium', 'high'],
5                     aspect=1.2, size=3.5, palette=sns.light_palette('black', 4)[1:])
6   g.map(plt.scatter, "residual sugar", "alcohol", alpha=0.8,
7       edgecolor='white', linewidth=0.5, s=wines['total sulfur dioxide']*2)
8   fig = g.fig
9   fig.subplots_adjust(top=0.8, wspace=0.3)
10  fig.suptitle('Wine Type - Sulfur Dioxide - Residual Sugar - Alcohol - Quality', fontsize=14)
11  l = g.add_legend(title='Wine Quality Class')
```

effective_data_viz_29.py hosted with ♥ by **GitHub**   view raw
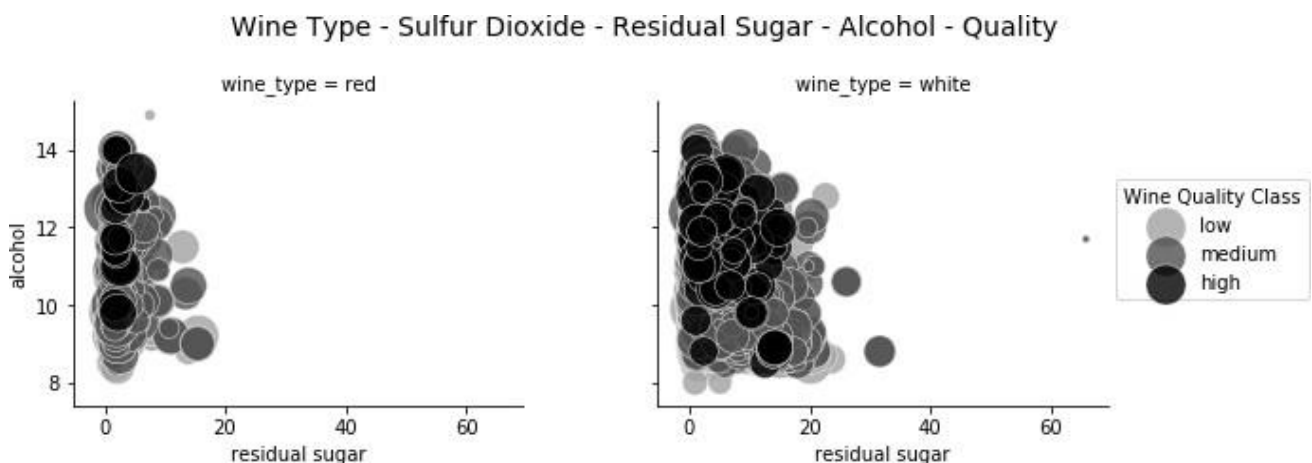
Visualizing data in five-dimensions leveraging bubble charts and the concept of hue, facets and size

This is basically an alternative approach to visualizing the same plot which we plotted previously for five dimensions. While the additional dimension of *depth* might confuse many when looking at the plot we plotted previously, this plot due to the advantage of *facets*, still remains effectively on the 2-D plane and hence is often more effective and easy to interpret.

We can already see that it's becoming complex handling so many data dimensions! If some of you are thinking, why not add more dimensions? Let's go ahead and give it a shot!

Visualizing data in Six Dimensions (6-D)

Now that we are having fun (I hope!), let's add another data dimension in our visualizations. We will leverage *depth*, *hue*, *size* and *shape* besides our *regular two axes* to depict all the six data dimensions.

```
 1   # Visualizing 6-D mix data using scatter charts
 2   # leveraging the concepts of hue, size, depth and shape
 3   fig = plt.figure(figsize=(8, 6))
 4   t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Typ
 5   ax = fig.add_subplot(111, projection='3d')
 6
 7   xs = list(wines['residual sugar'])
 8   ys = list(wines['alcohol'])
 9   zs = list(wines['fixed acidity'])
10   data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]
11
12   ss = list(wines['total sulfur dioxide'])
13   colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]
14   markers = [',' if q == 'high' else 'x' if q == 'medium' else 'o' for q in list(wines['quality_
```
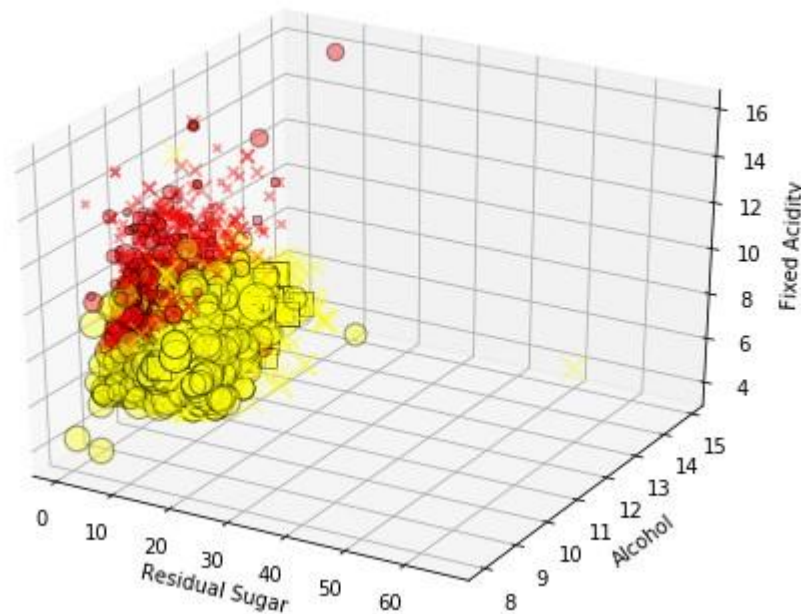
```
15
16    for data, color, size, mark in zip(data_points, colors, ss, markers):
17        x, y, z = data
18        ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=size, marker=mark)
19
20    ax.set_xlabel('Residual Sugar')
21    ax.set_ylabel('Alcohol')
22    ax.set_zlabel('Fixed Acidity')
```


Visualizing data in six-dimensions leveraging scatter charts and the concept of hue, depth, shape and size

Wow that is six dimensions in one plot! We have wine quality_label depicted by *shapes*, *high* (the squared pixel), *medium* (the X marks) and *low* (the circles) quality wines. The wine_type is represented by *hue,* fixed acidity by the *depth* and data point *size* represents total sulfur dioxide content.

Interpreting this might seem a bit taxing but consider a couple of components at a time when trying to understand what's going on.

1. Considering *shape & y-axis,* we have *high* and *medium* quality wines having higher alcohol levels as compared to *low* quality wines.

2. Considering *hue* and *size,* we have higher content of total sulfur dioxide for *white wines* as compared to *red wines.*

3. Considering *depth* and *hue,* we have *white wines* having lower fixed acidity levels as compared to *red wines*.

4. Considering *hue* and *x-axis,* we have *red wines* having lower levels of residual sugar as compared to *white wines*.

5. Considering *hue* and *shape, white wines* seem to have more *high* quality wines as compared to *red wines* (possibly due to larger sample size of *white wines*).

We can also build a 6-D visualization by removing the *depth* component and use *facets* instead for a categorical attribute.

```
1    # Visualizing 6-D mix data using scatter charts
2    # leveraging the concepts of hue, facets and size
3    g = sns.FacetGrid(wines, row='wine_type', col="quality", hue='quality_label', size=4)
4    g.map(plt.scatter,  "residual sugar", "alcohol", alpha=0.5,
5          edgecolor='k', linewidth=0.5, s=wines['total sulfur dioxide']*2)
6    fig = g.fig
7    fig.set_size_inches(18, 8)
8    fig.subplots_adjust(top=0.85, wspace=0.3)
9    fig.suptitle('Wine Type - Sulfur Dioxide - Residual Sugar - Alcohol - Quality Class - Quality
10   l = g.add_legend(title='Wine Quality Class')
```

effective_data_viz_31.py hosted with 🖤 by **GitHub**   view raw
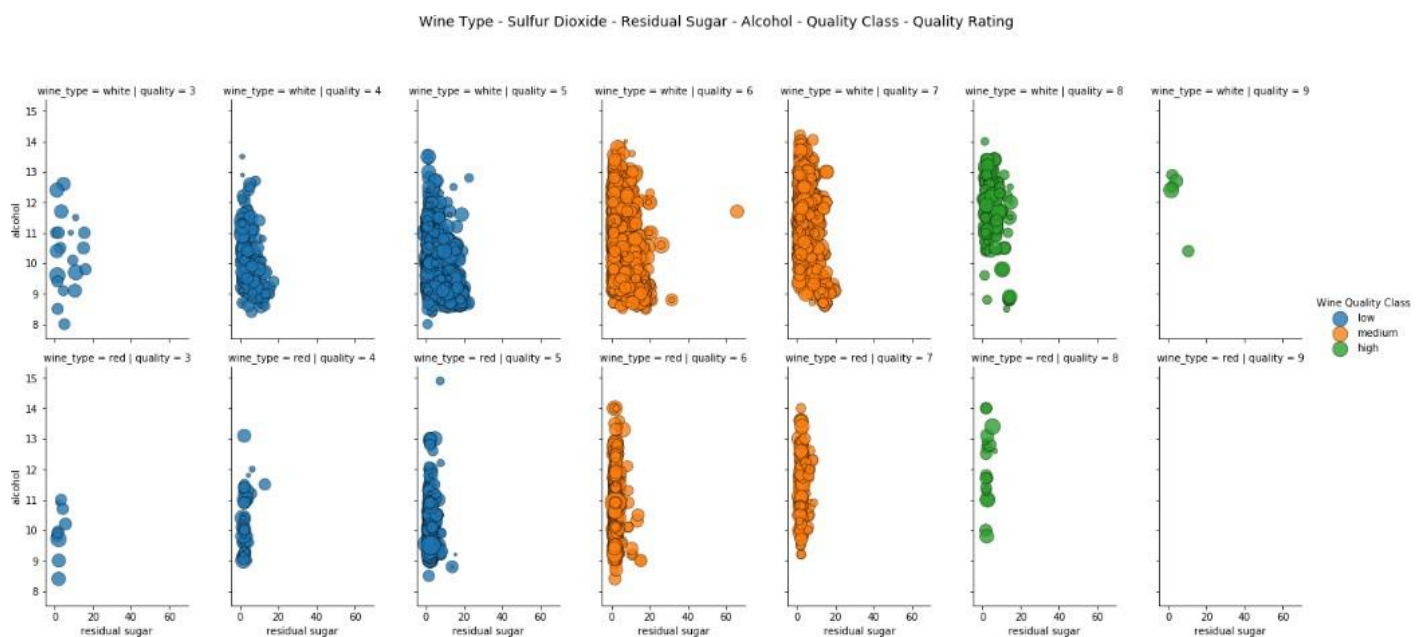
Visualizing data in six-dimensions leveraging scatter charts and the concept of hue, facets and size

Thus in this scenario, we leverage *facets* and *hue* to represent three categorical attributes and the *two regular axes* and *size* to represent three numerical attributes for our 6-D data visualization.