

# The Art of Effective Visualization of Multi-dimensional Data

A hands-on approach

Dipanjan Sarkar

Data Scientist, Author &  
Editor

# **GET THE SLIDES AND CODE HERE**

[http://bit.ly/pycon18\\_ds](http://bit.ly/pycon18_ds)

# Session Agenda



## Introduction

- What is data visualization?
- Why Data visualization?



## Effective Multi-dimensional Data Visualization

Whirlwind tour of the grammar of graphics



## Visualization Tools and Frameworks

- Visualization frameworks in Python
- Visualization frameworks in R
- Popular tools & frameworks



## Visualizing Structured Data

- Univariate analysis & visualizations
- Multivariate analysis & visualizations
- Visualizing from 1-D up to 6-D

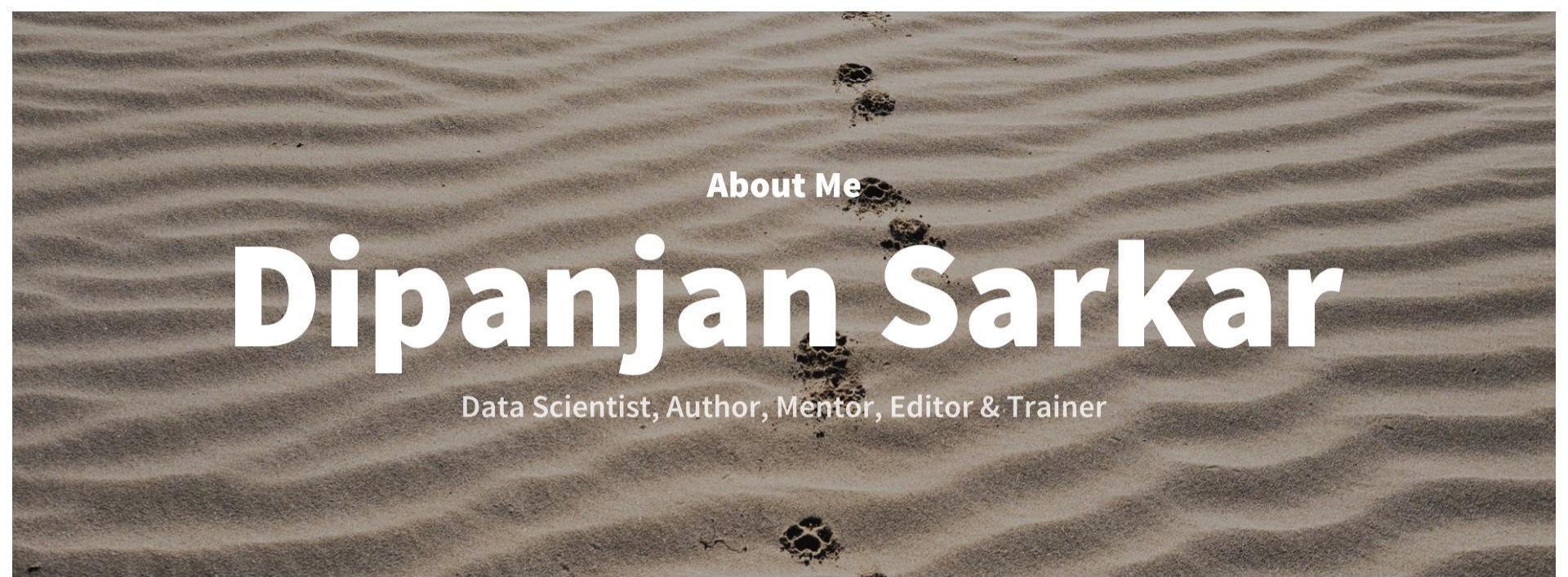


## Visualizing Unstructured Data

- Visualizing Text Data
- Visualizing Image Data
- Visualizing Audio Data



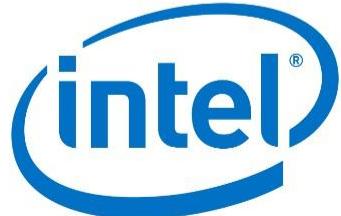
## Final Words



About Me

# Dipanjan Sarkar

Data Scientist, Author, Mentor, Editor & Trainer

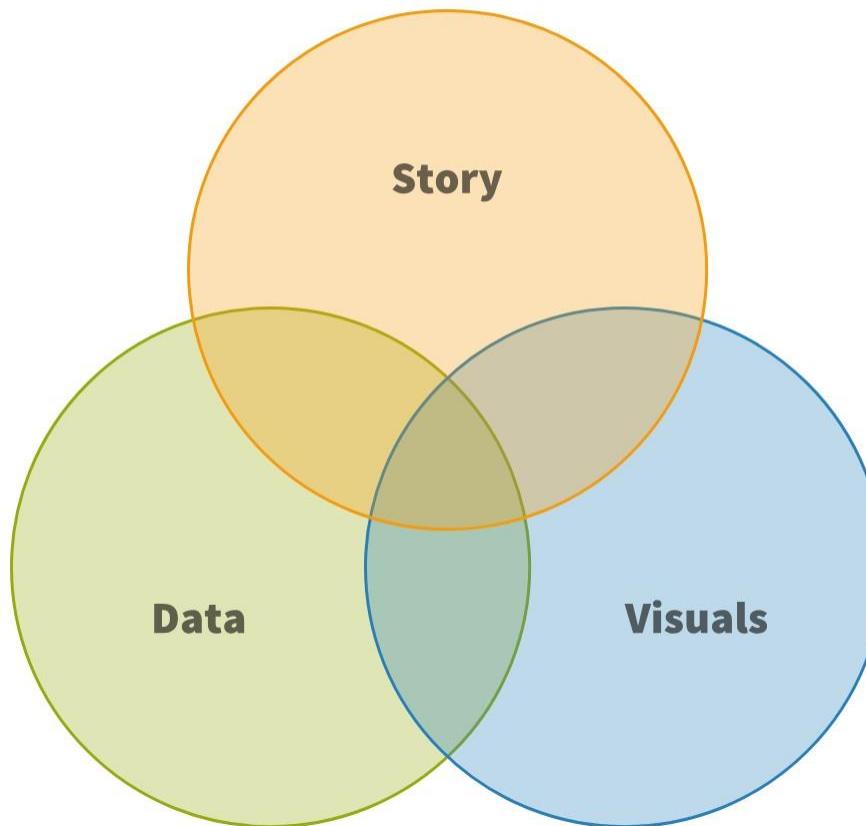


The background of the image is an aerial photograph of a rural landscape. It features several agricultural fields with distinct green and brown diagonal stripes, likely from crop rotation or different soil types. A network of white roads cuts through the fields, with one prominent road running diagonally across the center. A small dark vehicle is visible on one of the roads. The overall scene is a mix of natural and human-made patterns.

# Introduction

Understanding the what and why of Data Visualization

# Data Visualization & Storytelling





# What is Data Visualization?

- Data aggregation, summarization & visualization are some of the core pillars in any data science or analysis pipeline
- It is a specific technique or methodology for creating pictorial representations in the form of images, diagrams or animations to convey effective information from data
- Since the days of traditional Business Intelligence to even in this age of Artificial Intelligence, it has been a powerful tool, widely adopted by organizations

# Why Data Visualization?



90% of information transmitted to the human brain is visual



The human brain processes visuals 60K times faster than text



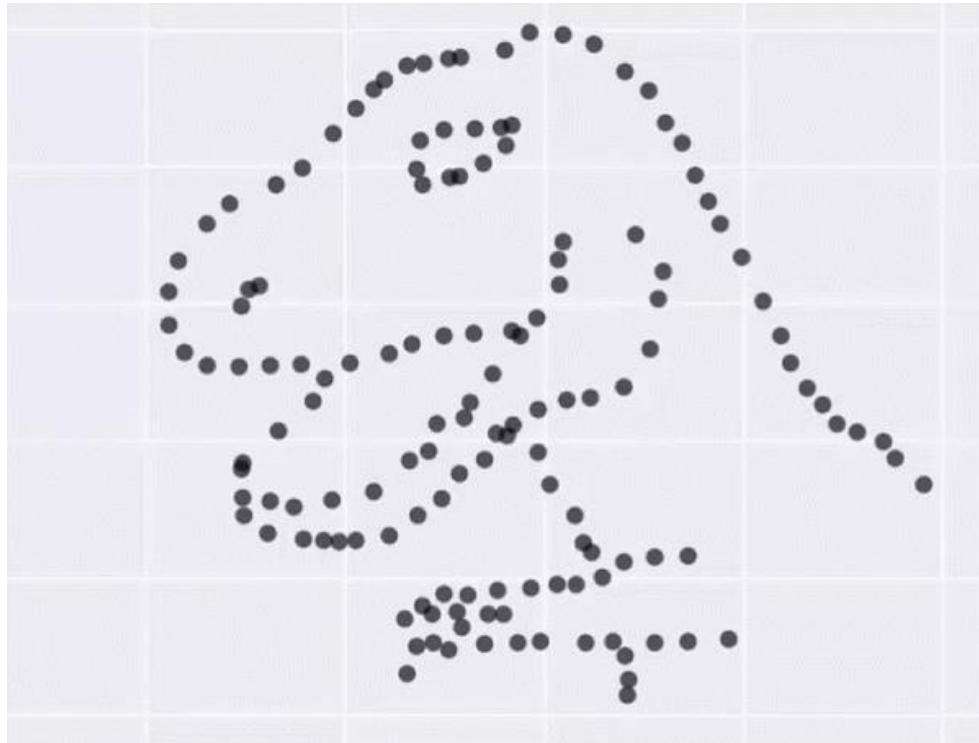
65% of humans are visual learners



The human brain can process an observed visual in 13 - 80 ms

# Why Data Visualization

Introducing The Datasaurus Dozen!



**What is common in  
all these images?**

Ans: Summary Stats are the same!

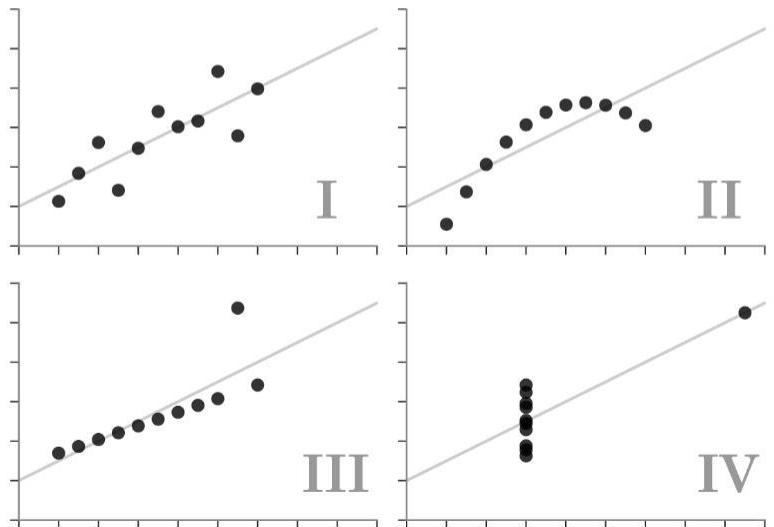
X Mean:	54.26
Y Mean:	47.83
X SD :	16.76
Y SD :	26.93
Corr. :	-0.06

# Why Data Visualization

Introducing Anscombe's Quartet

## Anscombe's Quartet

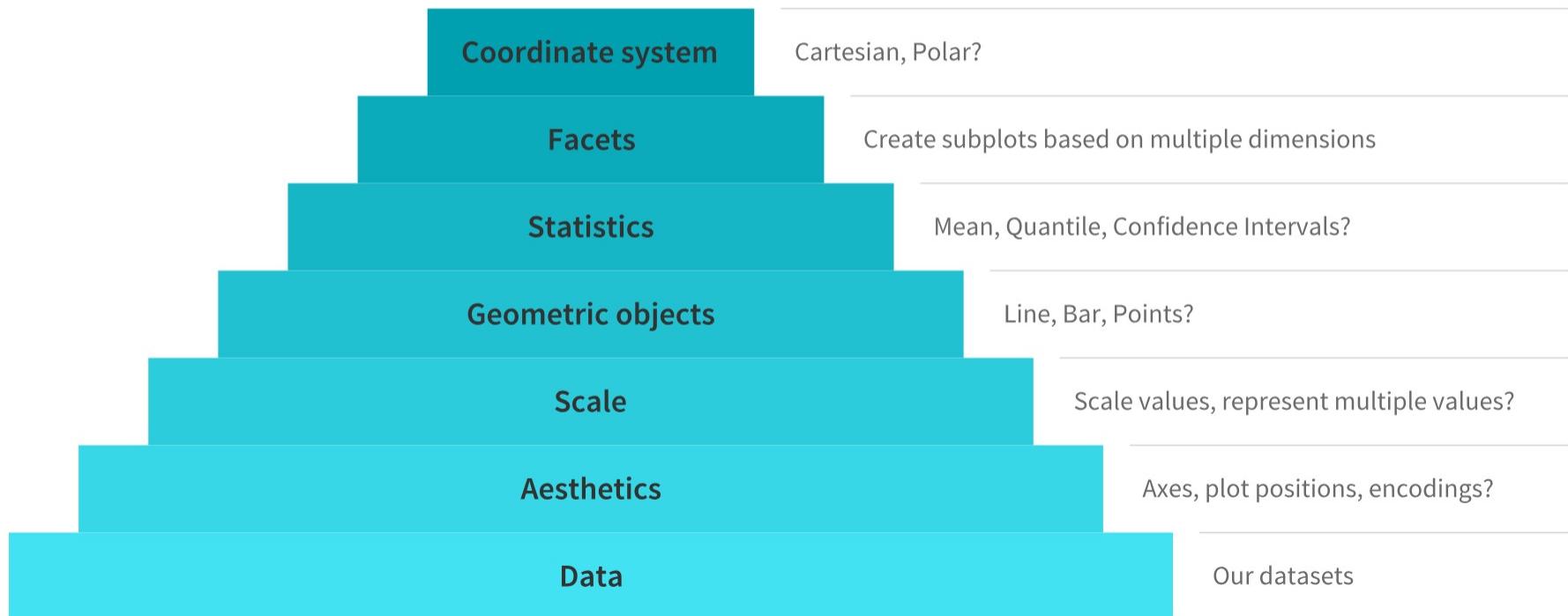
Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.



# Effective Multi-dimensional Data Visualization

Whirlwind tour of the grammar of graphics

# Major Components of the Grammar of Graphics



# Always start with the data

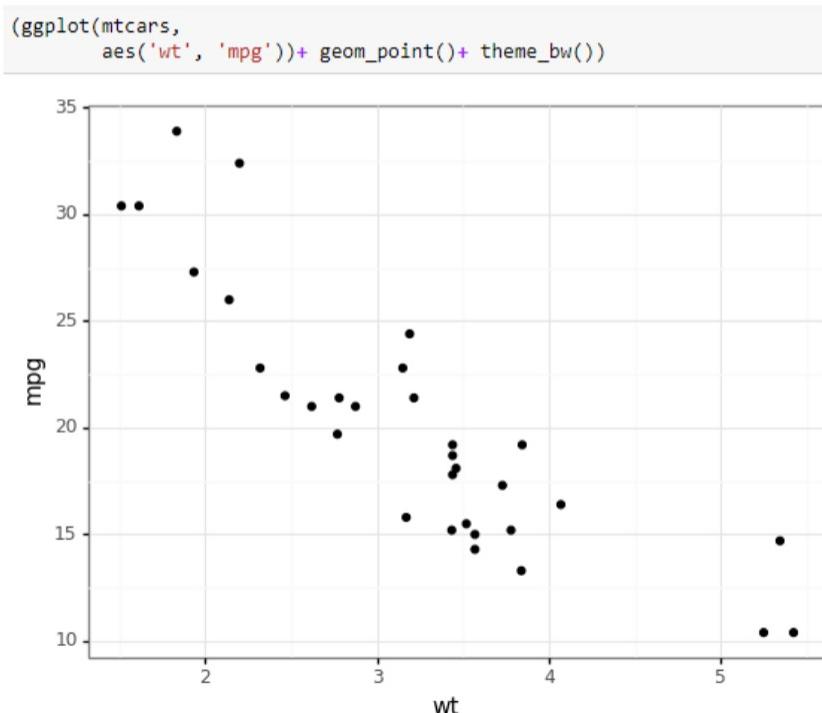
```
from plotnine import *
from plotnine.data import mtcars

mtcars.head()
```

		name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0		Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1		Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2		Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3		Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4		Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

# Start building with data, aesthetics, scale & geoms

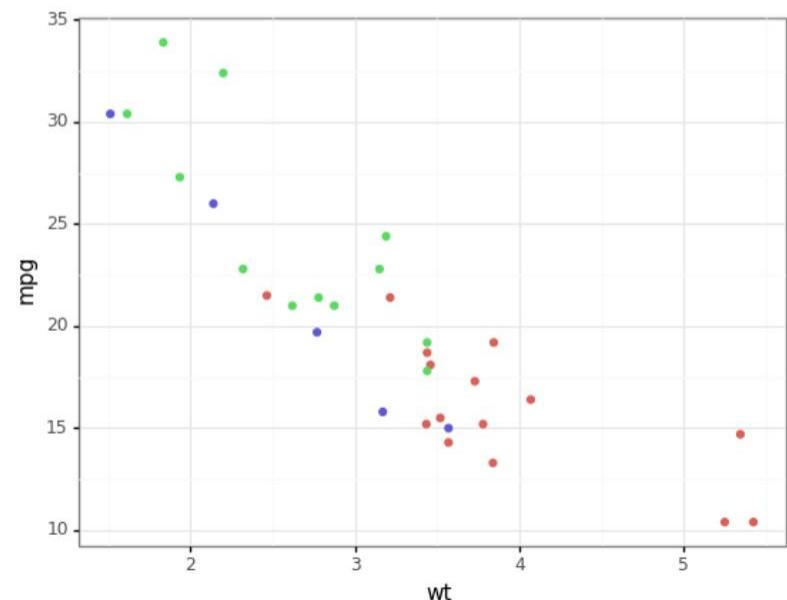
Visualize up to two dimensions in your data



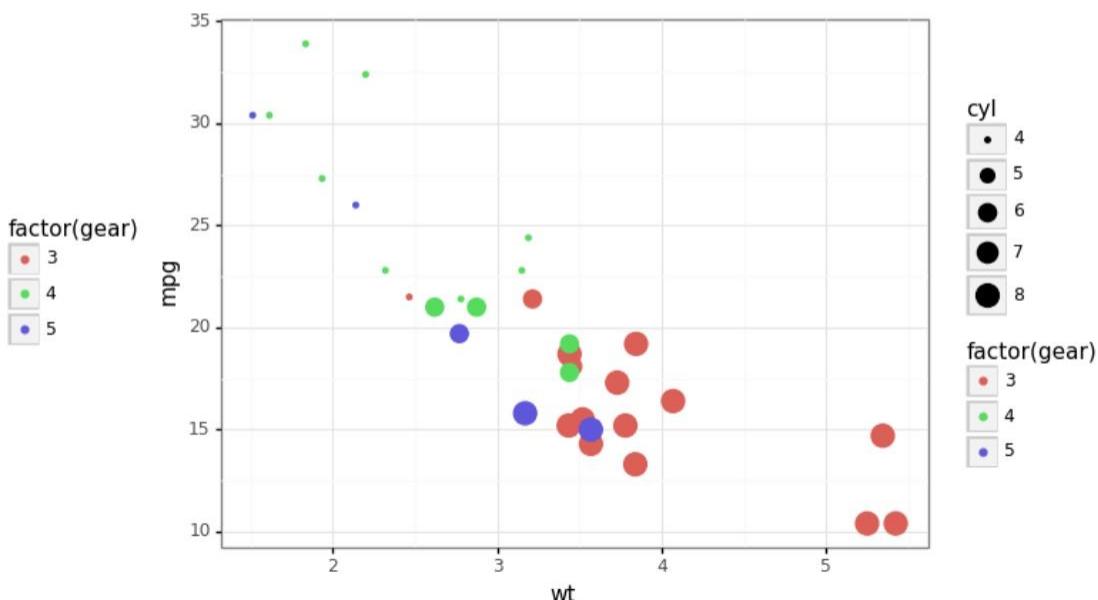
# Visualize from 2-D to 4-D with aesthetics & scale

Leveraging color and size here for visualizing up to four dimensions

```
(ggplot(mtcars,  
       aes('wt', 'mpg', color='factor(gear)'))  
+ geom_point() + theme_bw())
```



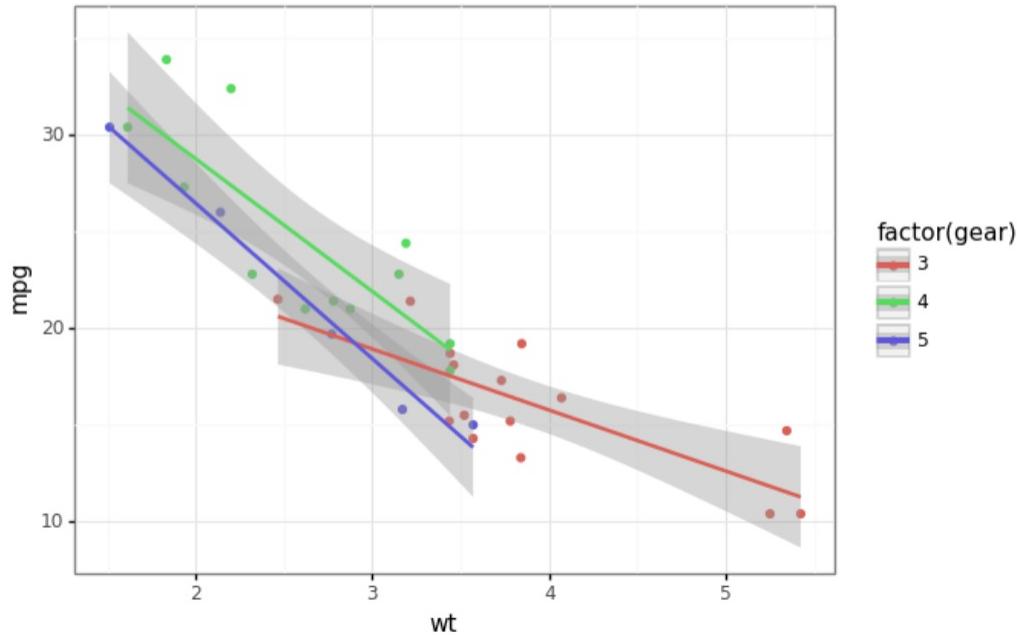
```
(ggplot(mtcars,  
       aes('wt', 'mpg', color='factor(gear)', size='cyl'))  
+ geom_point() + theme_bw())
```



# Visualize data, aesthetics with statistics

Visualizing data with rich statistical measures

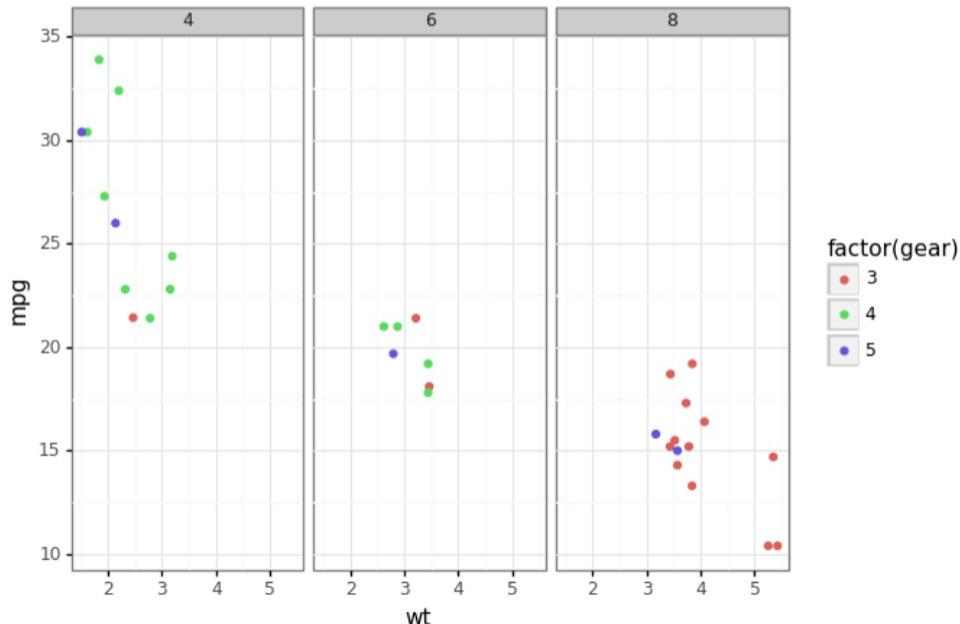
```
(ggplot(mtcars,  
       aes('wt', 'mpg', color='factor(gear)'))  
+ geom_point() + stat_smooth(method='lm')  
+ theme_bw())
```



# Visualize up to 4-D with aesthetics, scale & facets

Leveraging color & facets here for visualizing up to four dimensions

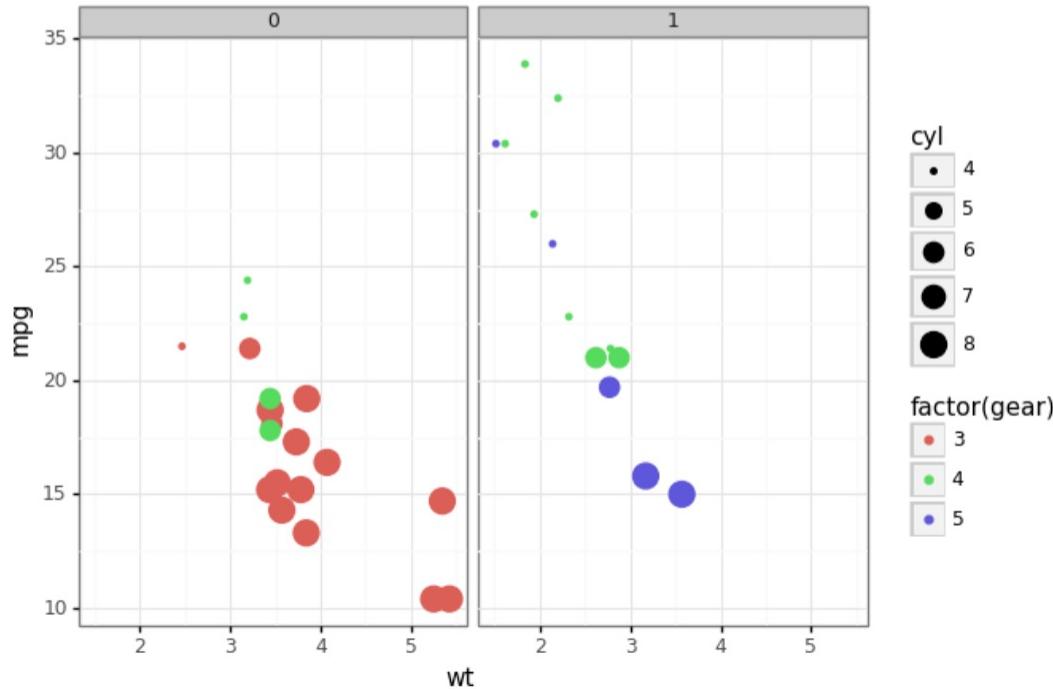
```
(ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)'))  
+ geom_point() + facet_wrap(~cyl) + theme_bw())
```



# Visualize up to 5-D with aesthetics, scale & facets

Leveraging color, size & facets here for visualizing up to five dimensions

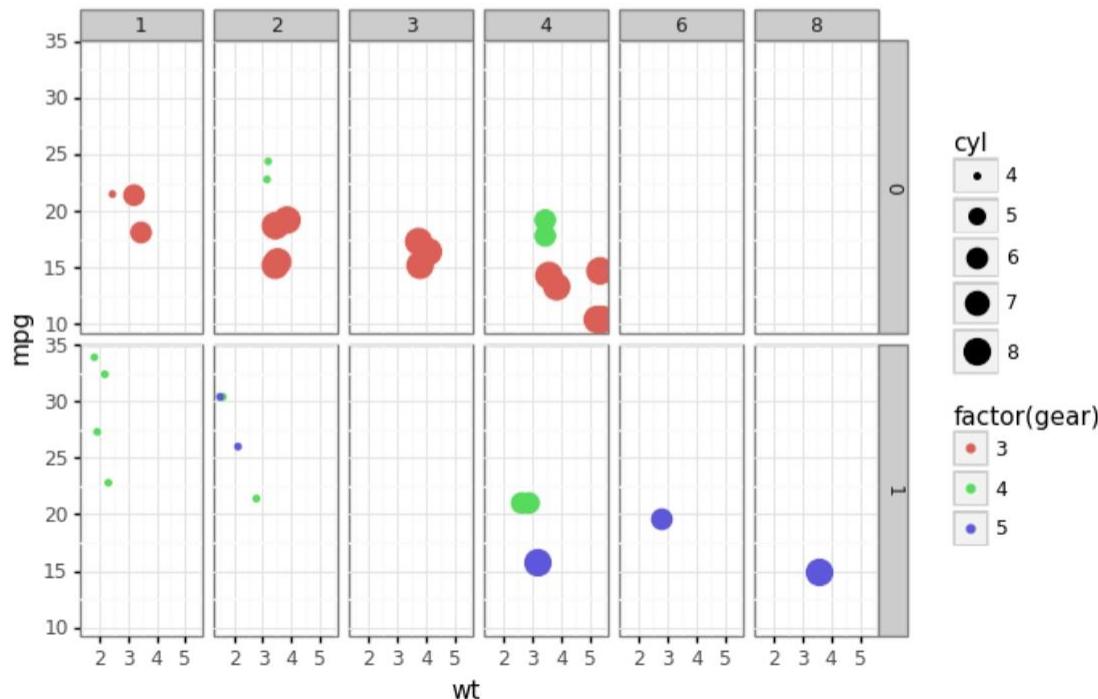
```
(ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)', size='cyl'))  
+ geom_point() + facet_wrap('~am') + theme_bw())
```



# Visualize up to 6-D with aesthetics, scale & facets

Leveraging color, size & multiple facets here for visualizing up to six dimensions

```
(ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)', size='cyl'))  
+ geom_point() + facet_grid('am ~ carb') + theme_bw())
```



# Data Visualization Tools & Frameworks

Quick Glance at popular tools & frameworks - General Purpose, Python & R

# Popular Data Visualization Tools & Frameworks



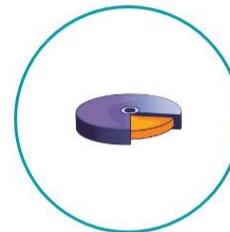
D3.js



Tableau



MS Excel



FusionCharts



Highcharts



Leaflet



Datawrapper



Plotly



Kibana

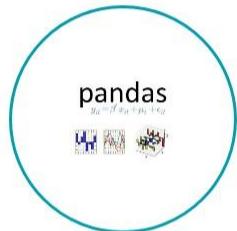
# Python Data Visualization Frameworks



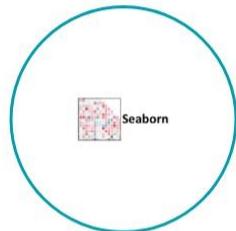
Matplotlib



Plotnine (ggplot)



Pandas



Seaborn



Bokeh



Pygal



Plotly

# R Data Visualization Frameworks



ggplot2



lattice



gigrap



taucharts



Plotly

# Visualizing Structured Data

Visualizing data from 1-D up to 6-D

# Start with the Data - as always!

Dataset is based on various properties of red and white wine

```
wines.head()
```

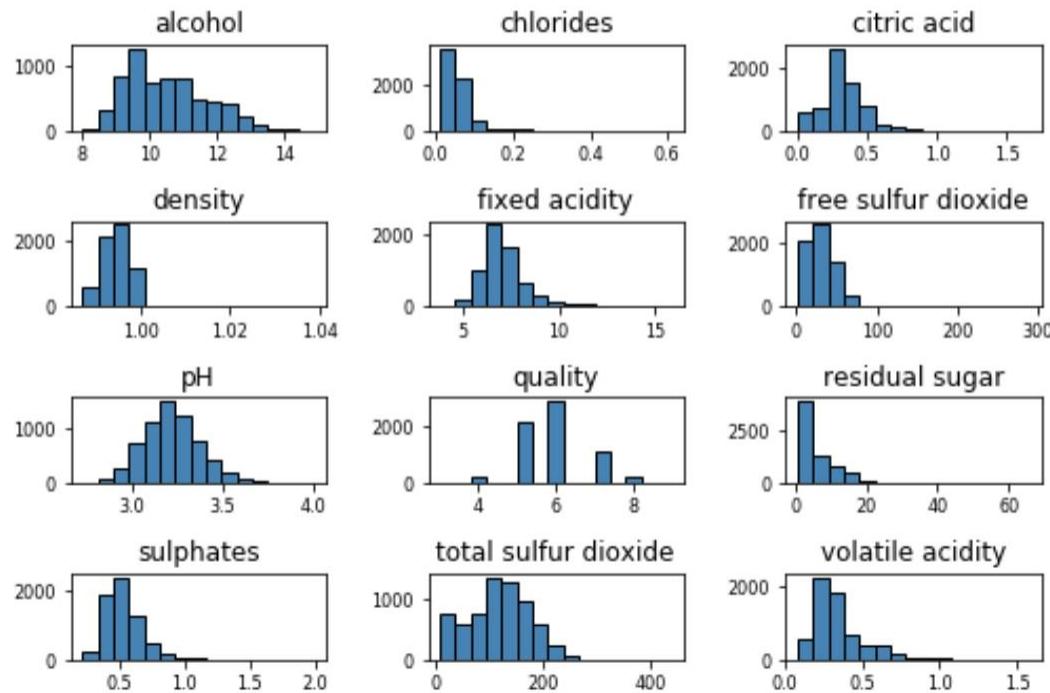
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	wine_type	quality_label
0	7.0	0.17	0.74	12.8	0.045	24.0	126.0	0.99420	3.26	0.38	12.2	8	white	high
1	7.7	0.64	0.21	2.2	0.077	32.0	133.0	0.99560	3.27	0.45	9.9	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

# Basic Data Understanding

- **fixed acidity:** Acids are one of the fundamental properties of wine and contribute greatly to the taste of the wine. Reducing acids significantly might lead to wines tasting flat. Fixed acids include tartaric, malic, citric, and succinic acids which are found in grapes (except succinic). This variable is usually expressed in  $\frac{g(tartaric\ acid)}{dm^3}$  in the dataset.
- **volatile acidity:** These acids are to be distilled out from the wine before completing the production process. It is primarily constituted of acetic acid though other acids like lactic, formic and butyric acids might also be present. Excess of volatile acids are undesirable and lead to unpleasant flavor. In the US, the legal limits of volatile acidity are 1.2 g/L for red table wine and 1.1 g/L for white table wine. The volatile acidity is expressed in  $\frac{g(acetic\ acid)}{dm^3}$  in the dataset.
- **citric acid:** This is one of the fixed acids which gives a wine its freshness. Usually most of it is consumed during the fermentation process and sometimes it is added separately to give the wine more freshness. It's usually expressed in  $\frac{g}{dm^3}$  in the dataset.
- **residual sugar:** This typically refers to the natural sugar from grapes which remains after the fermentation process stops, or is stopped. It's usually expressed in  $\frac{g}{dm^3}$  in the dataset.
- **chlorides:** This is usually a major contributor to saltiness in wine. It's usually expressed in  $\frac{g(sodium\ chloride)}{dm^3}$  in the dataset.
- **free sulfur dioxide:** This is the part of the sulphur dioxide that when added to a wine is said to be free after the remaining part binds. Winemakers will always try to get the highest proportion of free sulphur to bind. They are also known as sulfites and too much of it is undesirable and gives a pungent odour. This variable is expressed in  $\frac{mg}{dm^3}$  in the dataset.
- **total sulfur dioxide:** This is the sum total of the bound and the free sulfur dioxide ( $SO_2$ ). Here, it's expressed in  $\frac{mg}{dm^3}$ . This is mainly added to kill harmful bacteria and preserve quality and freshness. There are usually legal limits for sulfur levels in wines and excess of it can even kill good yeast and give out undesirable odour.

# Univariate Analysis - Visualizing 1-D

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

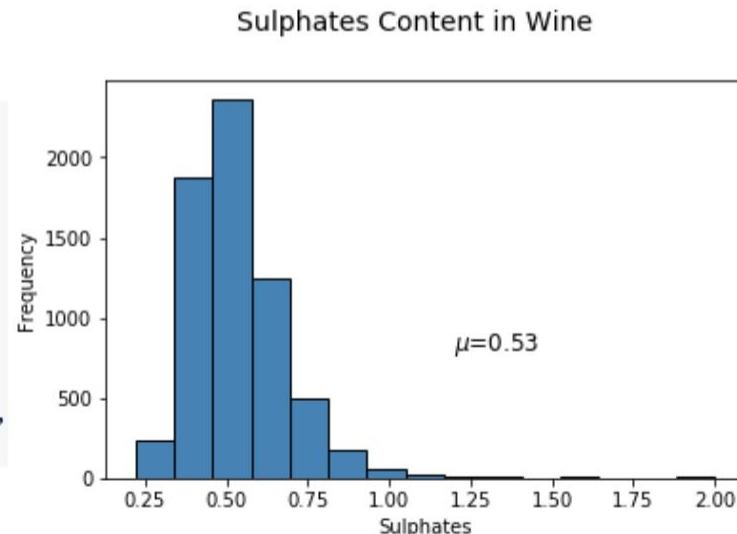


# Univariate Analysis - Visualizing 1-D

Visualizing a continuous numeric attribute

```
fig = plt.figure(figsize = (6,4))
title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
fig.subplots_adjust(top=0.85, wspace=0.3)

ax = fig.add_subplot(1,1, 1)
ax.set_xlabel("Sulphates")
ax.set_ylabel("Frequency")
ax.text(1.2, 800, r'$\mu$='+str(round(wines['sulphates'].mean(),2)),
       fontsize=12)
freq, bins, patches = ax.hist(wines['sulphates'], color='steelblue', bins=15,
                               edgecolor='black', linewidth=1)
```

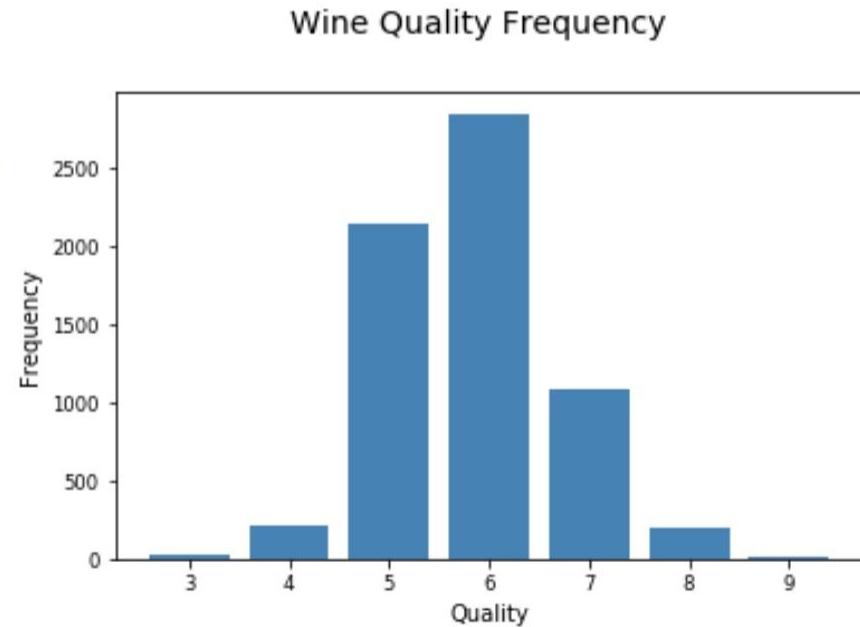


# Univariate Analysis - Visualizing 1-D

Visualizing a discrete categorical attribute

```
fig = plt.figure(figsize = (6, 4))
title = fig.suptitle("Wine Quality Frequency", fontsize=14)
fig.subplots_adjust(top=0.85, wspace=0.3)

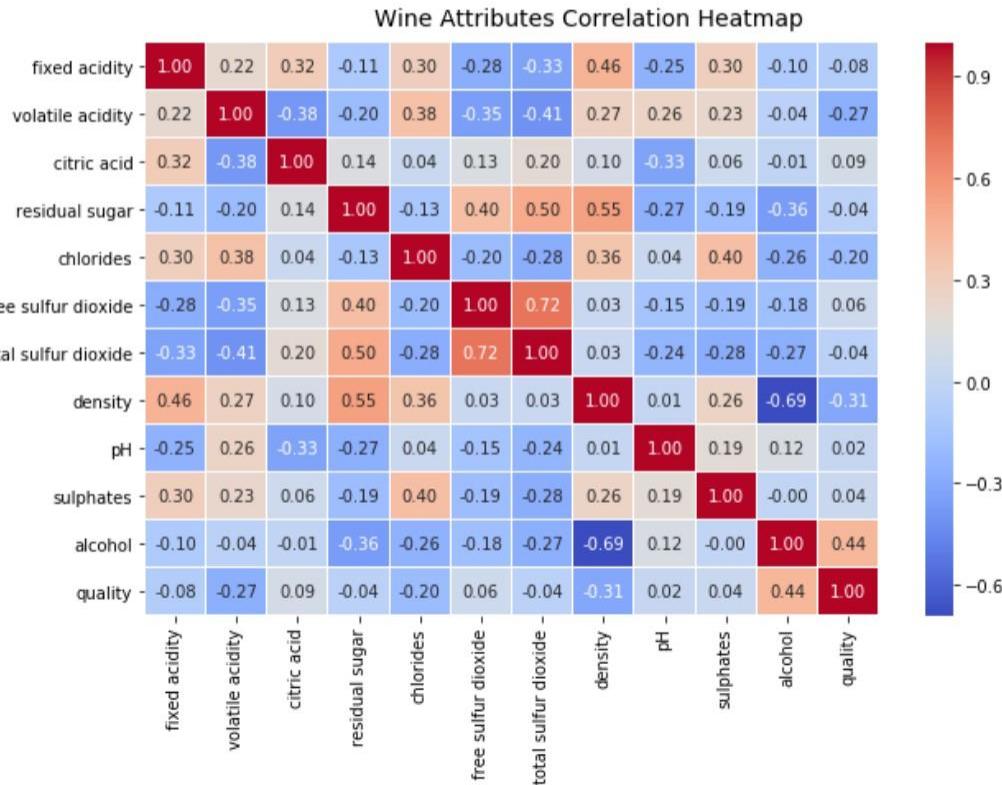
ax = fig.add_subplot(1,1, 1)
ax.set_xlabel("Quality")
ax.set_ylabel("Frequency")
w_q = wines['quality'].value_counts()
w_q = (list(w_q.index), list(w_q.values))
ax.tick_params(axis='both', which='major', labelsize=8.5)
bar = ax.bar(w_q[0], w_q[1], color='steelblue')
```



# Multivariate Analysis - Visualizing 2-D

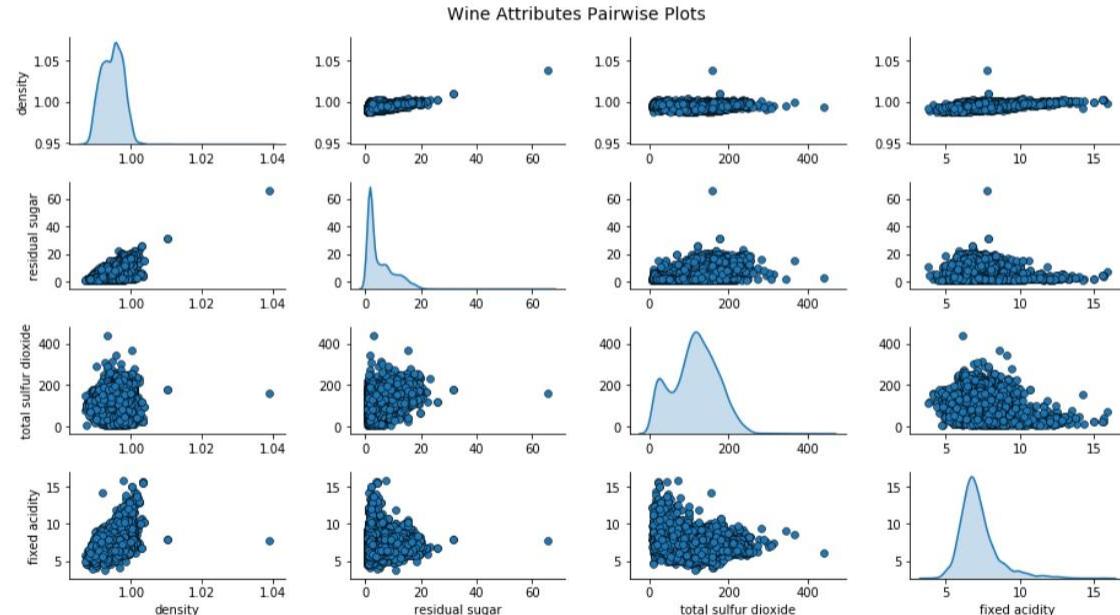
Visualizing feature correlations

```
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)
```



# Multivariate Analysis - Visualizing 2-D

Visualizing numeric features - use a pairplot



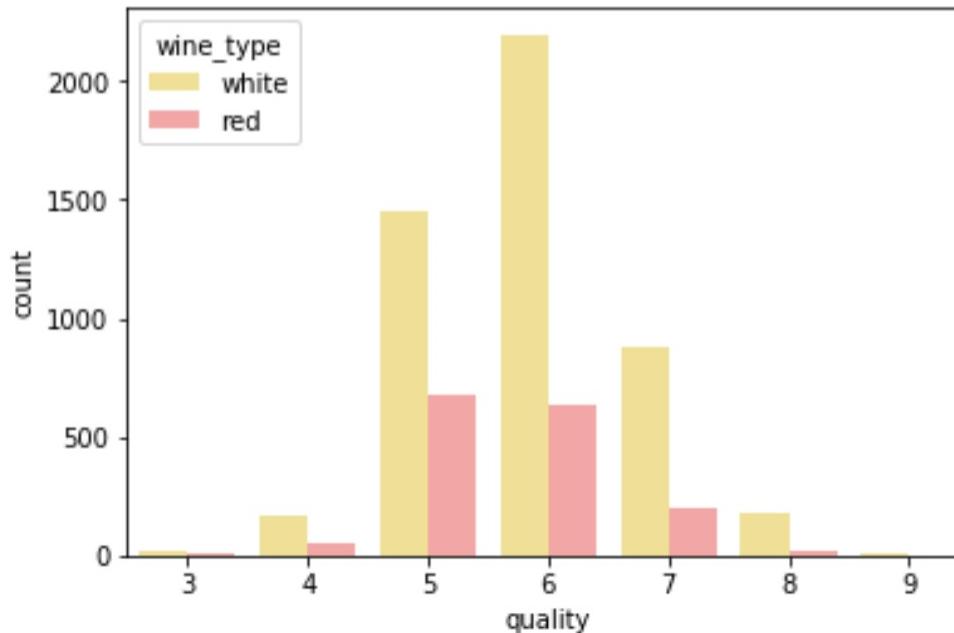
```
cols = ['density', 'residual sugar', 'total sulfur dioxide', 'fixed acidity']
pp = sns.pairplot(wines[cols], size=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

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

# Multivariate Analysis - Visualizing 2-D

Visualizing two categorical attributes - easier and cleaner with seaborn

```
cp = sns.countplot(x="quality", hue="wine_type", data=wines,  
                    palette={"red": "#FF9999", "white": "#FFE888"})
```

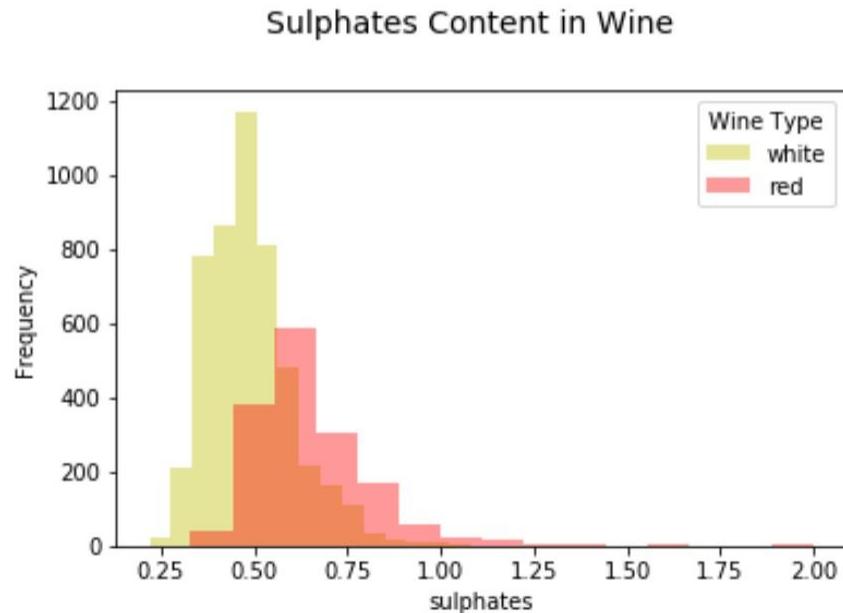


# Multivariate Analysis - Visualizing 2-D

Visualizing a mix of categorical & numeric attributes

```
fig = plt.figure(figsize = (6, 4))
title = fig.suptitle("Sulphates Content in Wine", fontsize=14)
fig.subplots_adjust(top=0.85, wspace=0.3)
ax = fig.add_subplot(1,1, 1)
ax.set_xlabel("Sulphates")
ax.set_ylabel("Frequency")

g = sns.FacetGrid(wines, hue='wine_type',
                  palette={"red": "r", "white": "y"})
g.map(sns.distplot, 'sulphates', kde=False, bins=15, ax=ax)
ax.legend(title='Wine Type')
plt.close(2)
```

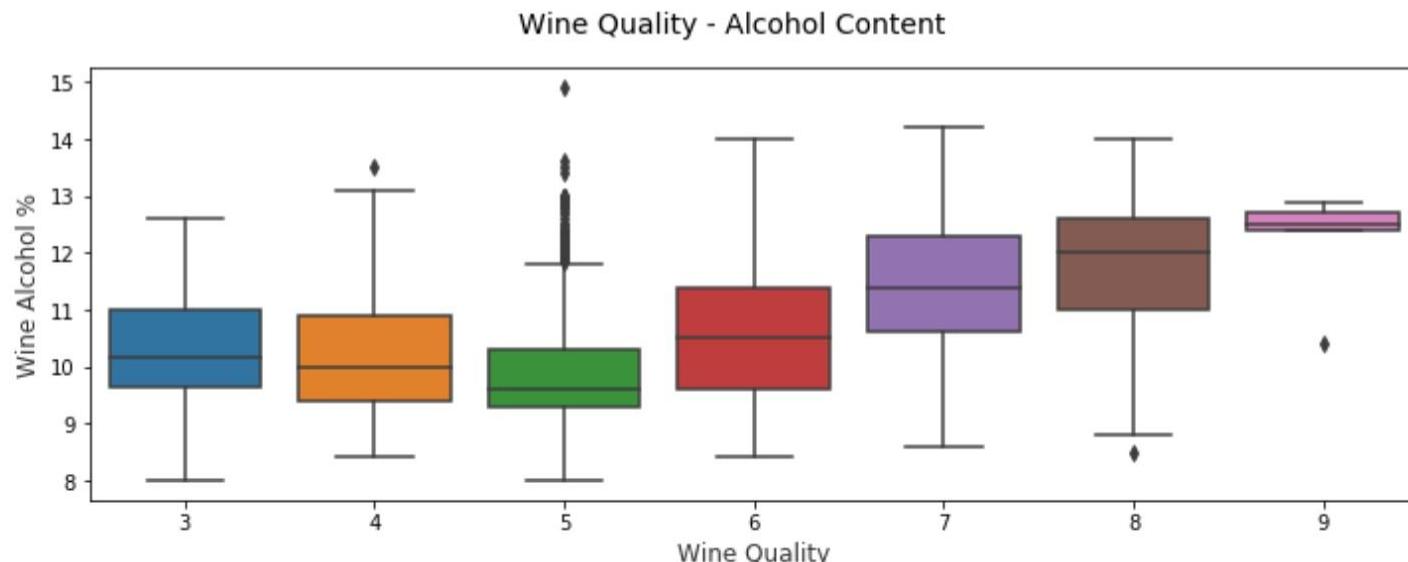


# Multivariate Analysis - Visualizing 2-D

Visualizing a mix of categorical & numeric attributes - more categories? use box-plots or violin plots

```
f, (ax) = plt.subplots(1, 1, figsize=(12, 4))
t = f.suptitle('Wine Quality - Alcohol Content', fontsize=14)

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



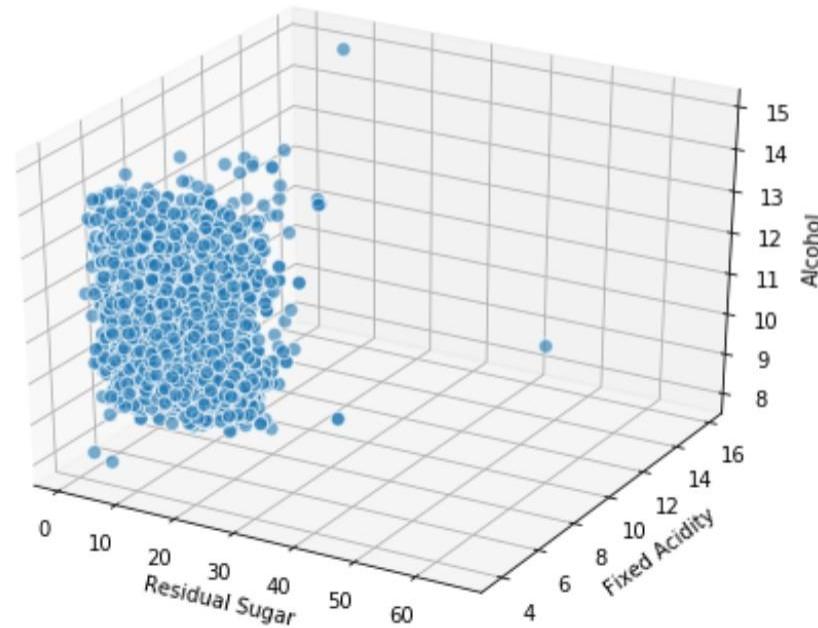
# Multivariate Analysis - Visualizing 3-D

Visualizing numeric attributes - is this effective?

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

xs = wines['residual sugar']
ys = wines['fixed acidity']
zs = wines['alcohol']
ax.scatter(xs, ys, zs, s=50, alpha=0.6, edgecolors='w')

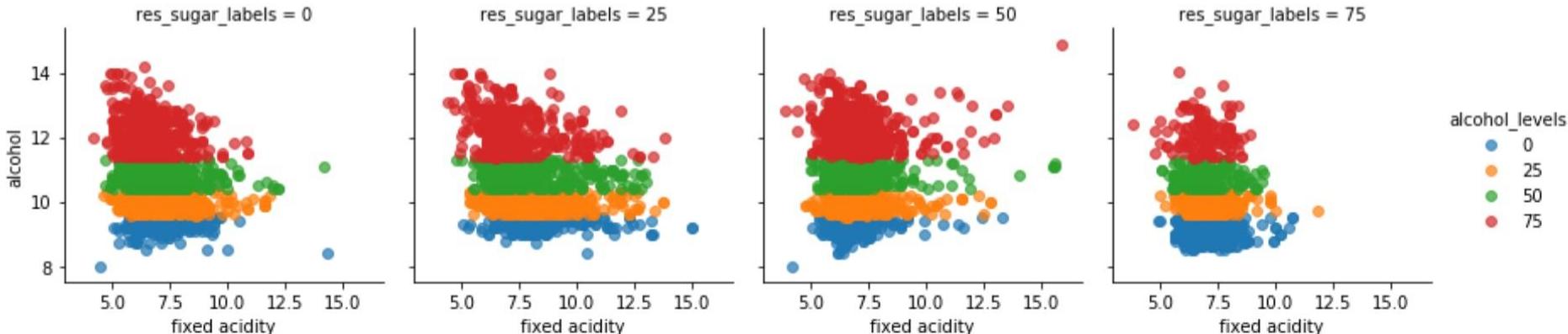
x1 = ax.set_xlabel('Residual Sugar')
y1 = ax.set_ylabel('Fixed Acidity')
z1 = ax.set_zlabel('Alcohol')
```



# Multivariate Analysis - Visualizing 3-D

Visualizing numeric attributes - bin one or more attributes and use facets, color and\or size

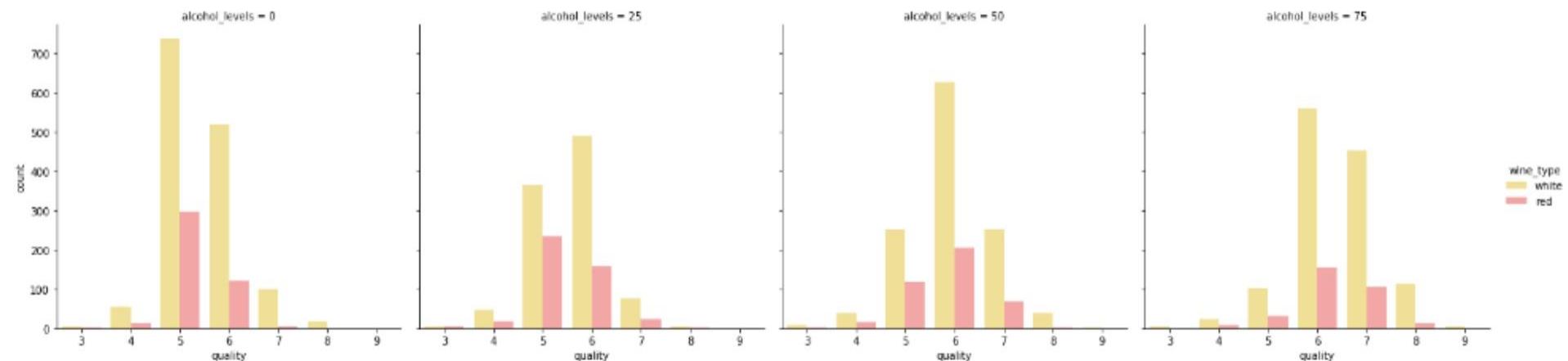
```
quantile_list = [0, .25, .5, .75, 1.]  
quantile_labels = ['0', '25', '50', '75']  
wines['res_sugar_labels'] = pd.qcut(wines['residual sugar'],  
                                     q=quantile_list, labels=quantile_labels)  
wines['alcohol_levels'] = pd.qcut(wines['alcohol'],  
                                     q=quantile_list, labels=quantile_labels)  
g = sns.FacetGrid(wines, col="res_sugar_labels",  
                  hue='alcohol_levels')  
g.map(plt.scatter, "fixed acidity", "alcohol", alpha=.7)  
g.add_legend();
```



# Multivariate Analysis - Visualizing 3-D

Visualizing categorical attributes using facets and color

```
fc = sns.factorplot(x="quality", hue="wine_type", col="alcohol_levels",
                     data=wines, kind="count",
                     palette={"red": "#FF9999", "white": "#FFE888"})
```



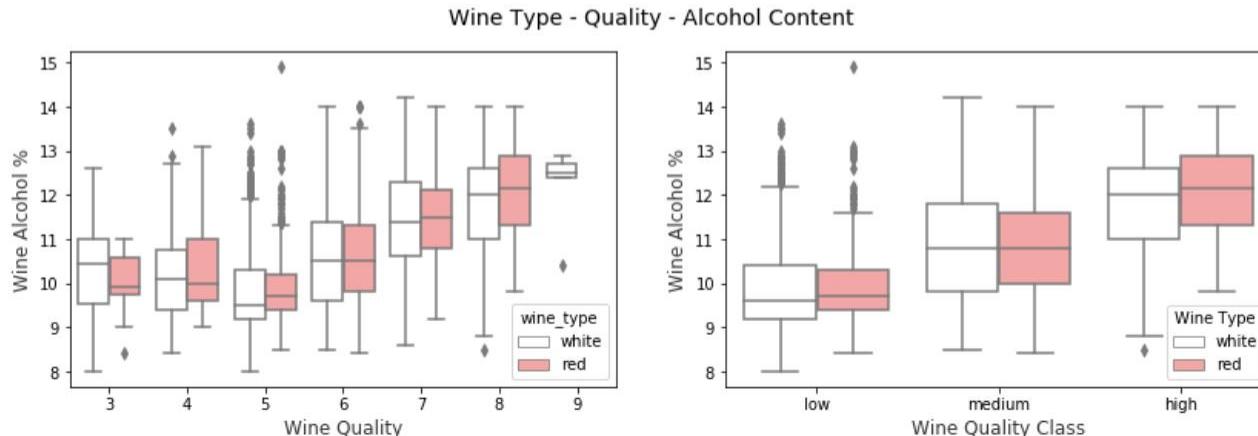
# Multivariate Analysis - Visualizing 3-D

Visualizing mixed attributes

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))
f.suptitle('Wine Type - Quality - Alcohol Content', fontsize=14)

sns.boxplot(x="quality", y="alcohol", hue="wine_type",
            data=wines, palette={"red": "#FF9999", "white": "white"}, ax=ax1)
ax1.set_xlabel("Wine Quality", size = 12, alpha=0.8)
ax1.set_ylabel("Wine Alcohol %", size = 12, alpha=0.8)

sns.boxplot(x="quality_label", y="alcohol", hue="wine_type",
            data=wines, palette={"red": "#FF9999", "white": "white"}, ax=ax2)
ax2.set_xlabel("Wine Quality Class", size = 12, alpha=0.8)
ax2.set_ylabel("Wine Alcohol %", size = 12, alpha=0.8)
l = plt.legend(loc='best', title='Wine Type')
```



# Multivariate Analysis - Visualizing 4-D

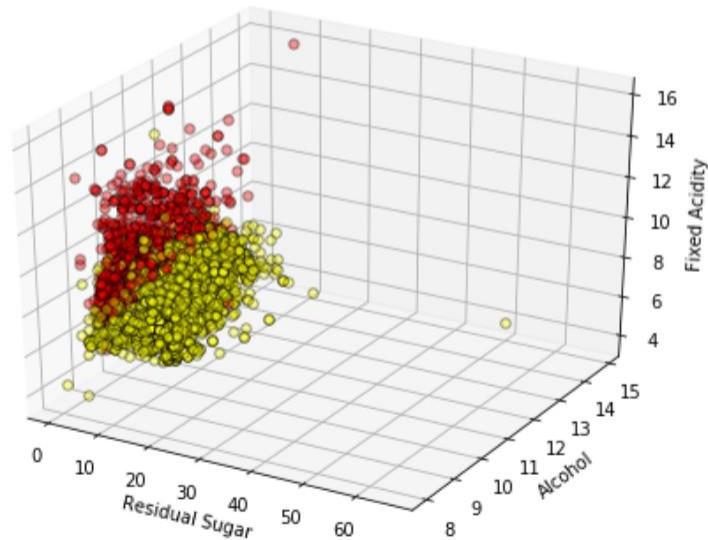
Visualizing mixed attributes (more numeric variables)

```
fig = plt.figure(figsize=(8, 6))
t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Type', fontsize=14)
ax = fig.add_subplot(111, projection='3d')

xs = list(wines['residual sugar'])
ys = list(wines['alcohol'])
zs = list(wines['fixed acidity'])
data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]
colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]

for data, color in zip(data_points, colors):
    x, y, z = data
    ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=30)

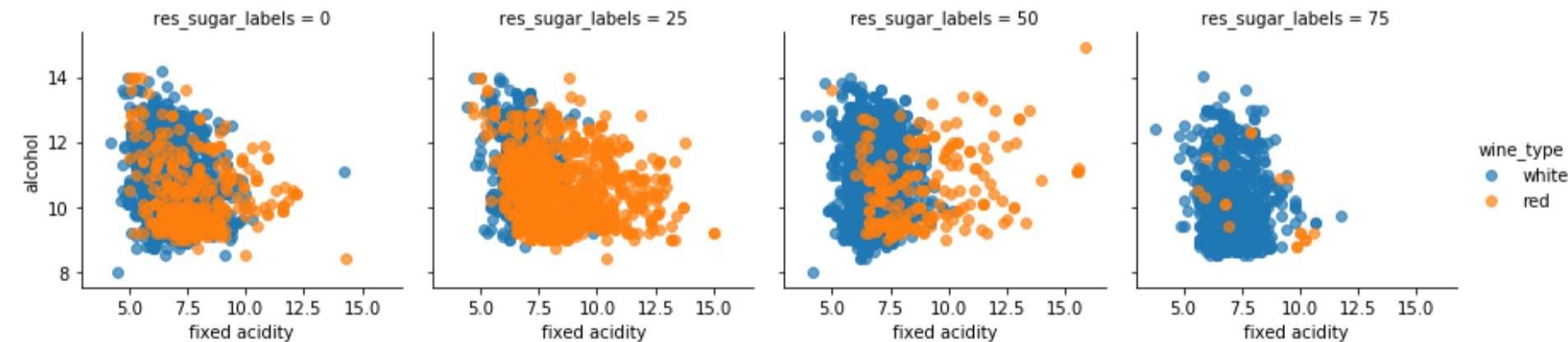
xl = ax.set_xlabel('Residual Sugar')
yl = ax.set_ylabel('Alcohol')
zl = ax.set_zlabel('Fixed Acidity')
```



# Multivariate Analysis - Visualizing 4-D

Visualizing mixed attributes (more numeric variables) - need to bin one numeric variable

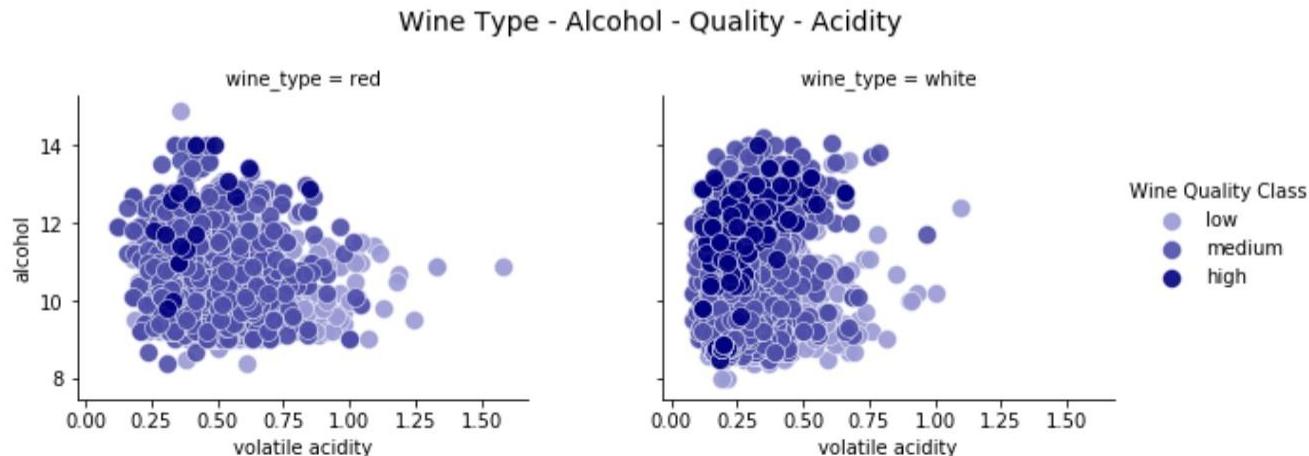
```
g = sns.FacetGrid(wines, col="res_sugar_labels",
                   hue='wine_type')
g.map(plt.scatter, "fixed acidity", "alcohol", alpha=.7)
g.add_legend();
```



# Multivariate Analysis - Visualizing 4-D

Visualizing mixed attributes

```
g = sns.FacetGrid(wines, col="wine_type", hue='quality_label',
                   col_order=['red', 'white'], hue_order=['low', 'medium', 'high'],
                   aspect=1.2, size=3.5, palette=sns.light_palette('navy', 4)[1:])
g.map(plt.scatter, "volatile acidity", "alcohol", alpha=0.9,
      edgecolor='white', linewidth=0.5, s=100)
fig = g.fig
fig.subplots_adjust(top=0.8, wspace=0.3)
fig.suptitle('Wine Type - Alcohol - Quality - Acidity', fontsize=14)
l = g.add_legend(title='Wine Quality Class')
```



# Multivariate Analysis - Visualizing 5-D

Visualizing mixed attributes - not always effective

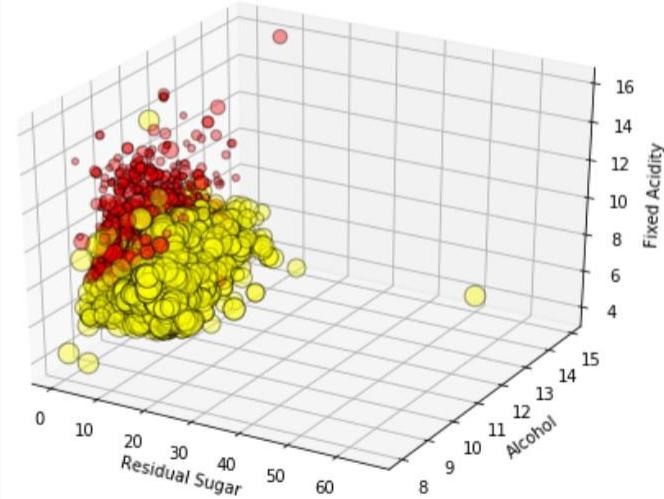
```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Type',
                 fontsize=14)

xs = list(wines['residual sugar'])
ys = list(wines['alcohol'])
zs = list(wines['fixed acidity'])
data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]

ss = list(wines['total sulfur dioxide'])
colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]

for data, color, size in zip(data_points, colors, ss):
    x, y, z = data
    ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=size)

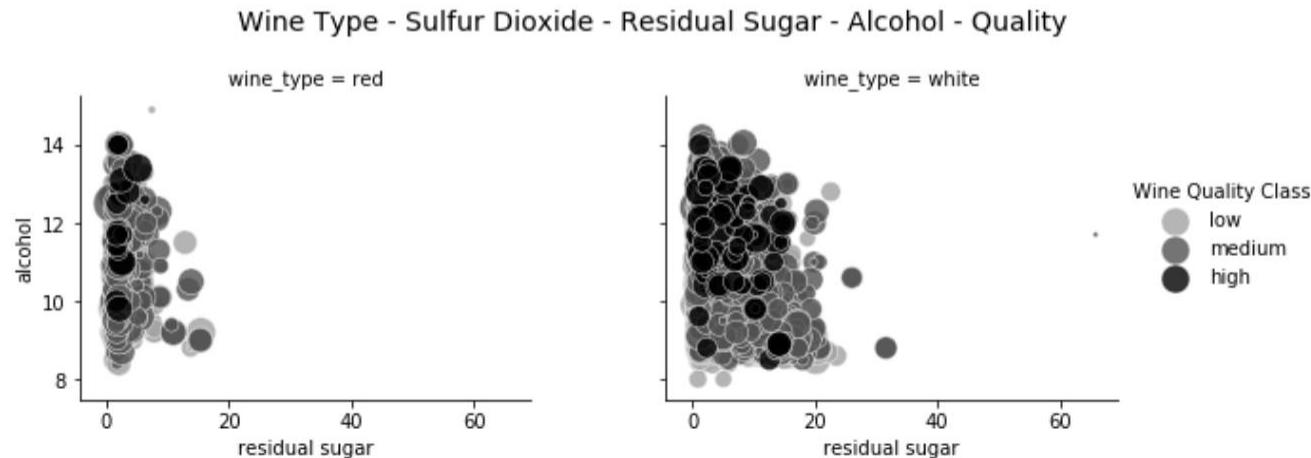
ax.set_xlabel('Residual Sugar')
ax.set_ylabel('Alcohol')
ax.set_zlabel('Fixed Acidity')
```



# Multivariate Analysis - Visualizing 5-D

Visualizing mixed attributes - start with faceting and sizes

```
g = sns.FacetGrid(wines, col="wine_type", hue='quality_label',
                   col_order=['red', 'white'], hue_order=['low', 'medium', 'high'],
                   aspect=1.2, size=3.5, palette=sns.light_palette('black', 4)[1:])
g.map(plt.scatter, "residual sugar", "alcohol", alpha=0.8,
      edgecolor='white', linewidth=0.5, s=wines['total sulfur dioxide'])
fig = g.fig
fig.subplots_adjust(top=0.8, wspace=0.3)
fig.suptitle('Wine Type - Sulfur Dioxide - Residual Sugar - Alcohol - Quality', fontsize=14)
l = g.add_legend(title='Wine Quality Class')
```



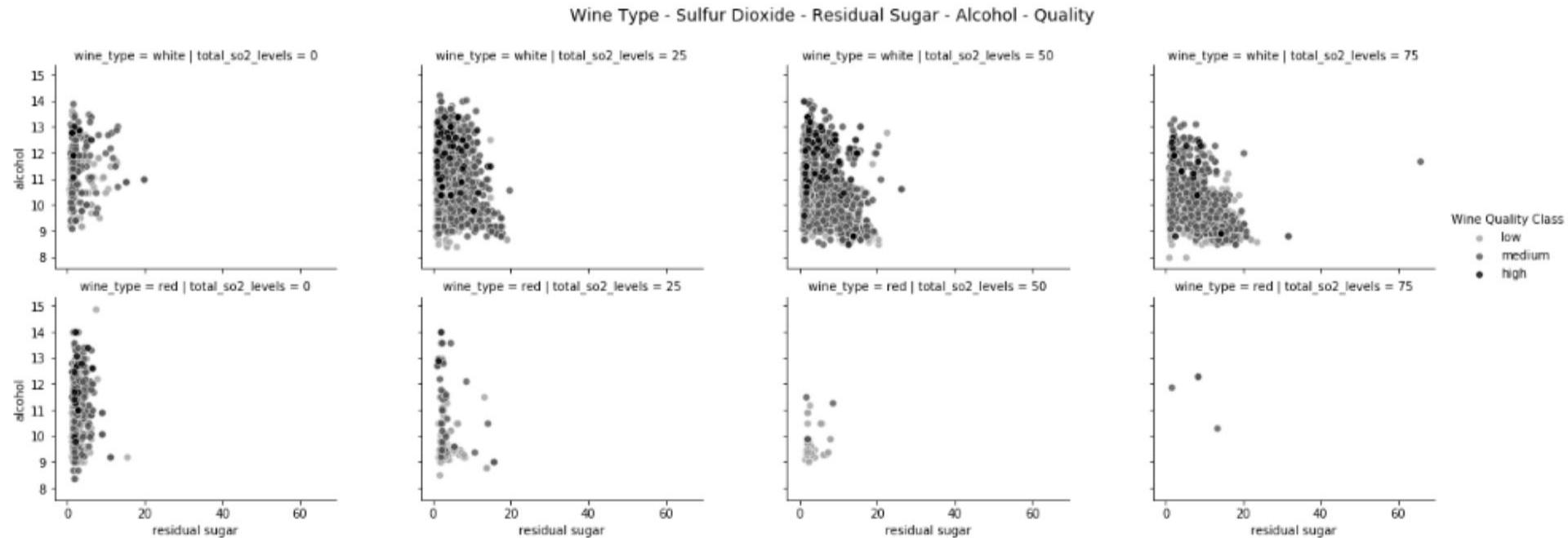
# Multivariate Analysis - Visualizing 5-D

Visualizing mixed attributes - multi-facets often better than size (often needs binning of numerics)

```
quantile_list = [0, .25, .5, .75, 1.]
quantile_labels = ['0', '25', '50', '75']
wines['total_so2_levels'] = pd.qcut(wines['total sulfur dioxide'],
                                     q=quantile_list, labels=quantile_labels)
g = sns.FacetGrid(wines, row="wine_type", col='total_so2_levels', hue='quality_label',
                   hue_order=['low', 'medium', 'high'], palette=sns.light_palette('black', 4)[1:],
                   aspect=1.2, size=3.5)
g.map(plt.scatter, "residual sugar", "alcohol", alpha=0.8,
      edgecolor='white', linewidth=0.5)
fig = g.fig
fig.subplots_adjust(top=0.8, wspace=0.3)
fig.suptitle('Wine Type - Sulfur Dioxide - Residual Sugar - Alcohol - Quality', fontsize=14)
l = g.add_legend(title='Wine Quality Class')
```

# Multivariate Analysis - Visualizing 5-D

Visualizing mixed attributes - multi-facets often better than size (often needs binning of numerics)



# Multivariate Analysis - Visualizing 6-D

Visualizing mixed attributes - color, size, shape & depth

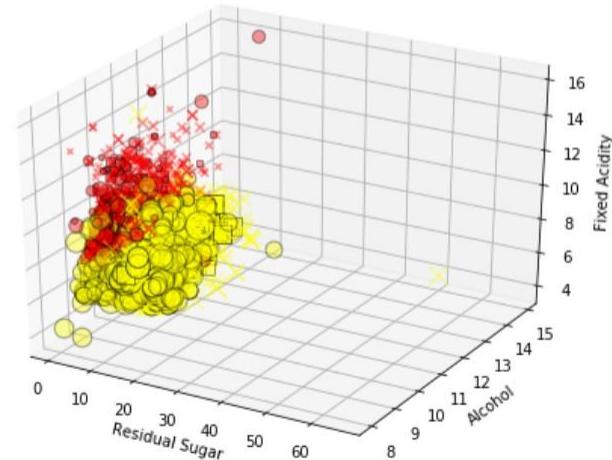
```
fig = plt.figure(figsize=(8, 6))
t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Type - Quality',
                 fontsize=14)
ax = fig.add_subplot(111, projection='3d')

xs = list(wines['residual sugar'])
ys = list(wines['alcohol'])
zs = list(wines['fixed acidity'])
data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]

ss = list(wines['total sulfur dioxide'])
colors = ['red' if wt == 'red' else 'yellow' for wt in list(wines['wine_type'])]
markers = [',' if q == 'high' else 'x' if q == 'medium' else 'o' for q in list(wines['quality_label'])]

for data, color, size, mark in zip(data_points, colors, ss, markers):
    x, y, z = data
    ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=size, marker=mark)

ax.set_xlabel('Residual Sugar')
ax.set_ylabel('Alcohol')
ax.set_zlabel('Fixed Acidity')
```



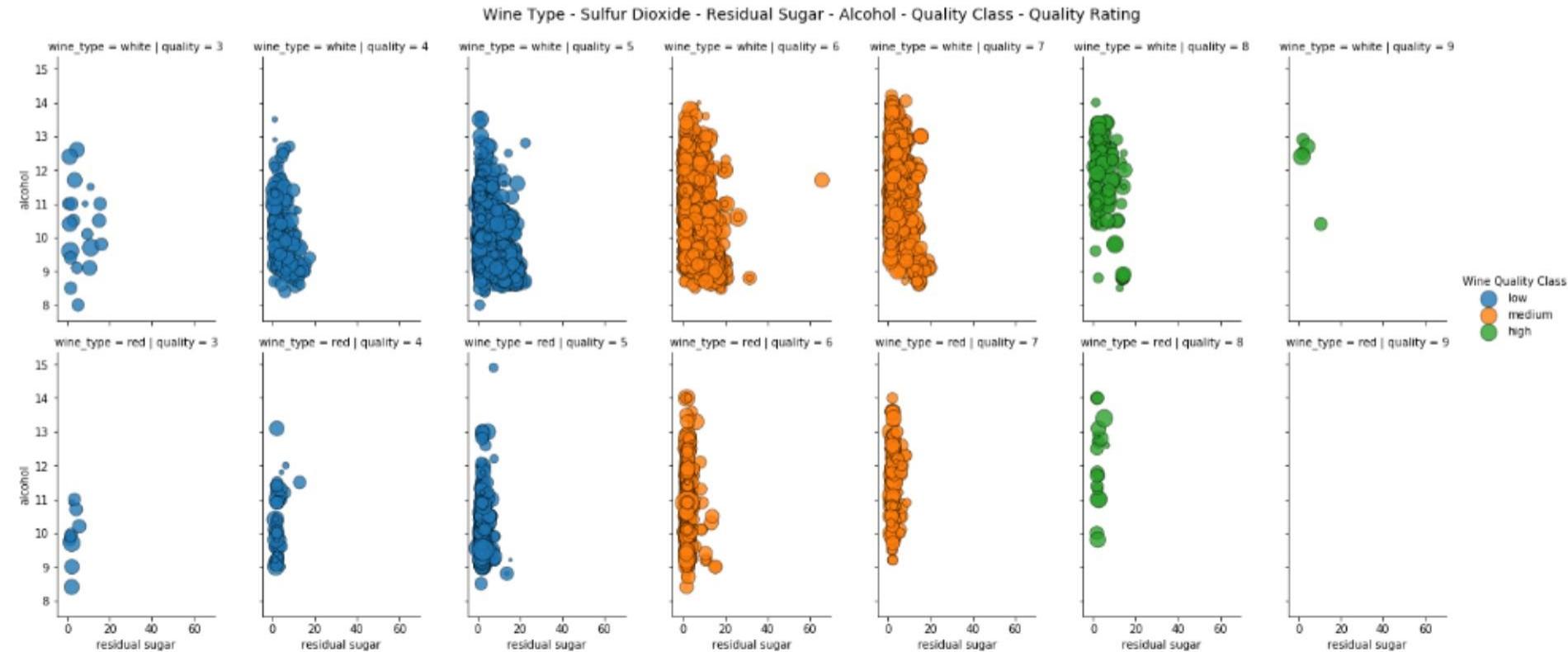
# Multivariate Analysis - Visualizing 6-D

Visualizing mixed attributes - color, size & multi-facets

```
g = sns.FacetGrid(wines, row='wine_type', col="quality", hue='quality_label', size=4)
g.map(plt.scatter, "residual sugar", "alcohol", alpha=0.8,
      edgecolor='k', linewidth=0.5, s=wines['total sulfur dioxide'])
fig = g.fig
fig.set_size_inches(18, 8)
fig.subplots_adjust(top=0.85, wspace=0.3)
fig.suptitle('Wine Type - Sulfur Dioxide - Residual Sugar - Alcohol - Quality Class - Quality Rating',
             fontsize=14)
l = g.add_legend(title='Wine Quality Class')
```

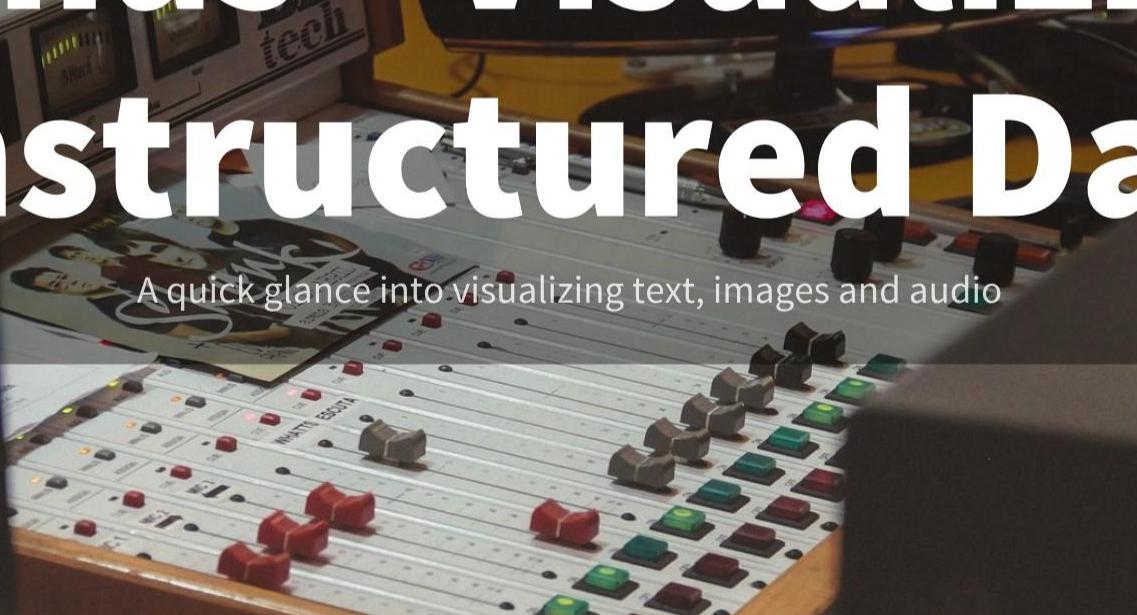
# Multivariate Analysis - Visualizing 6-D

Visualizing mixed attributes - color, size & multi-facets



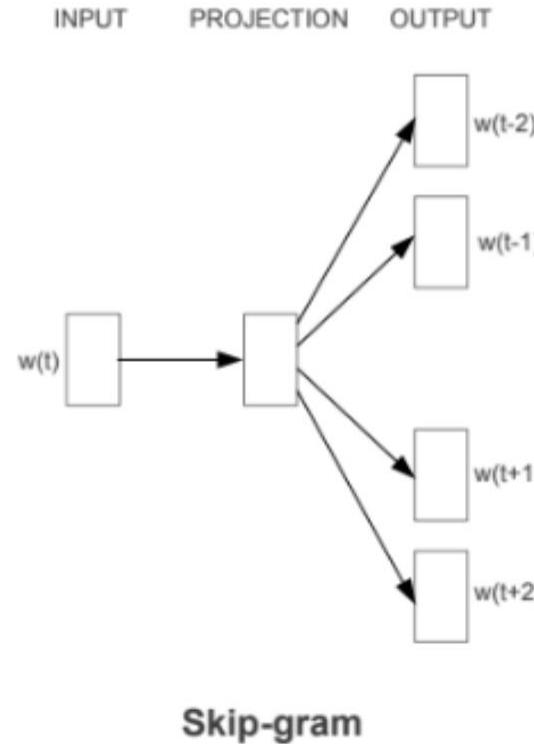
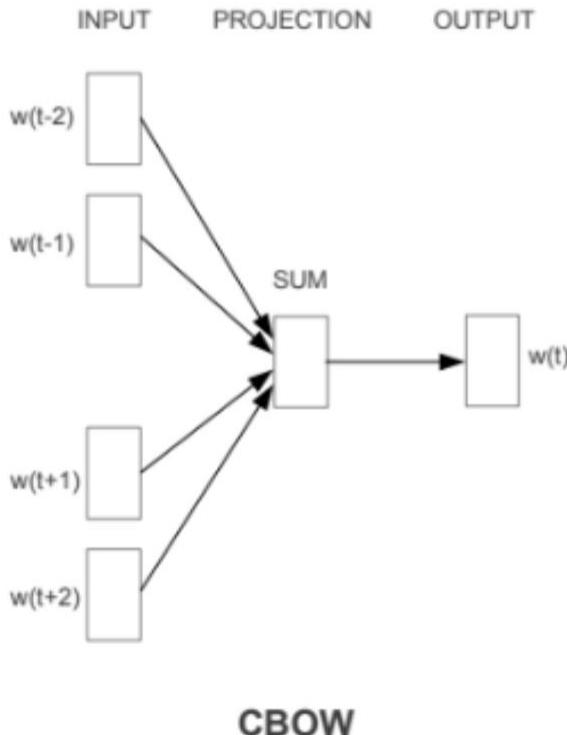
# Bonus- Visualizing Unstructured Data

A quick glance into visualizing text, images and audio



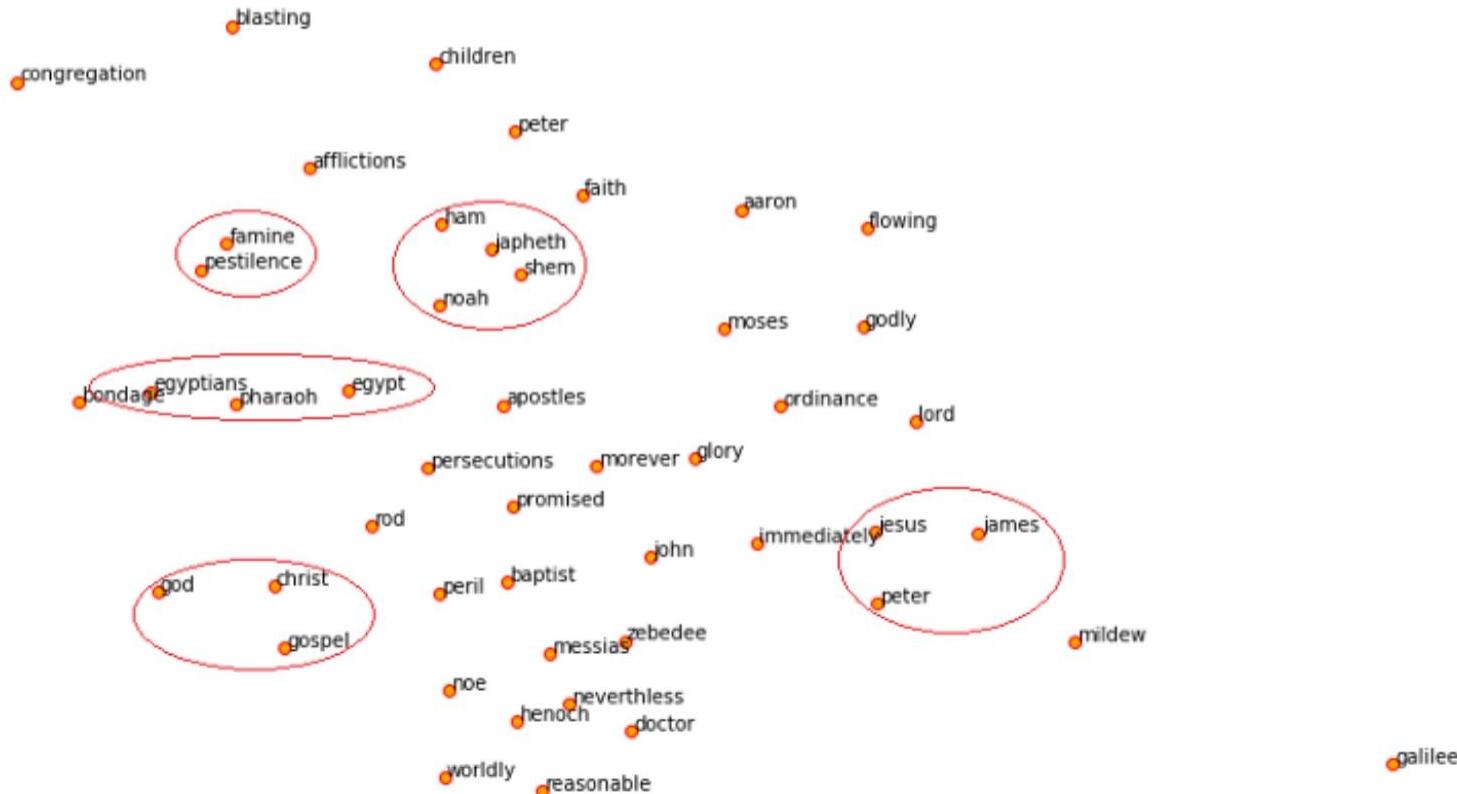
# Visualizing Text Data - Language Semantics

## Word Embedding Methods



# Visualizing Text Data - Language Semantics

Visualizing embeddings from the Bible



# Visualizing Text Data - Language Semantics

Can visualize documents based on similarity once you have document vectors

	Document	Category
0	The sky is blue and beautiful.	weather
1	Love this blue and beautiful sky!	weather
2	The quick brown fox jumps over the lazy dog.	animals
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food
4	I love green eggs, ham, sausages and bacon!	food
5	The brown fox is quick and the blue dog is lazy!	animals
6	The sky is very blue and the sky is very beautiful today	weather
7	The dog is lazy but the brown fox is quick!	animals

# Visualizing Text Data - Language Semantics

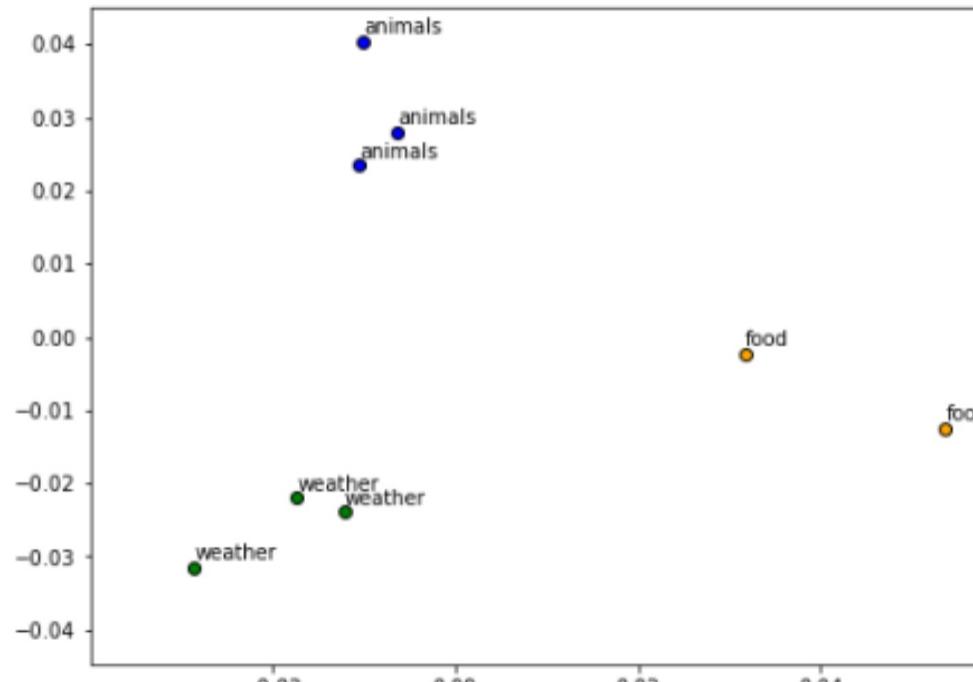
Can visualize documents based on similarity once you have document vectors

	0	1	2	3	4	5	6	7	8	9
0	0.004690	0.009370	-0.009667	0.026014	0.034989	0.010402	-0.033441	-0.011956	-0.000243	0.010552
1	0.005751	0.003210	-0.001964	0.016550	0.030962	0.004340	-0.019463	-0.009149	0.008256	0.019600
2	0.016712	0.004806	-0.001924	-0.027226	0.029162	-0.017201	-0.023197	-0.008610	-0.011976	0.020602
3	-0.009216	0.003900	-0.009232	-0.005232	0.042718	-0.032432	-0.006243	0.013524	0.008095	0.021227
4	-0.016321	-0.008715	-0.001633	-0.000501	0.027367	-0.037861	0.008515	0.021066	0.020373	0.016512
5	0.018538	0.007522	-0.009302	-0.025440	0.037199	-0.009890	-0.021419	-0.011769	-0.002221	0.018277
6	0.008532	0.008041	-0.016573	0.018653	0.036140	0.004038	-0.022891	0.000484	-0.005900	0.015766
7	0.024419	0.012915	-0.010596	-0.039350	0.037018	-0.013378	-0.020677	-0.004417	-0.011864	0.013540

Document level embeddings

# Visualizing Text Data - Language Semantics

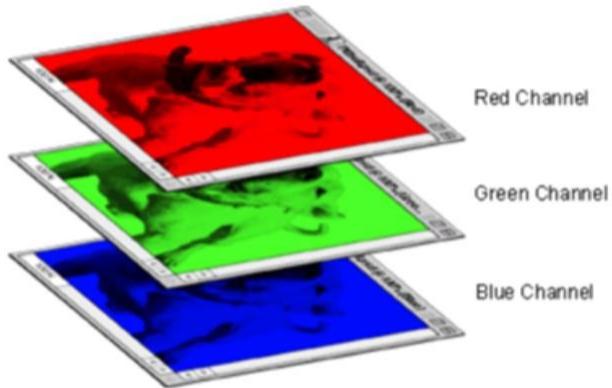
Can visualize documents based on similarity once you have document vectors - use PCA



Visualizing our document clusters

# Visualizing Image Data

Images are just n-dimensional tensors where each channel is a 2-D matrix



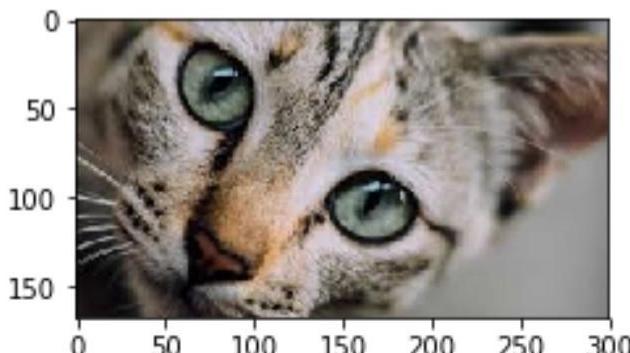
- Consider a face image of size  $200 \times 200$  pixels
  - A color (RGB) image has three channels
  - Total pixels in the image are  $200 \times 200 \times 3$

# Visualizing Image Data

Images are just n-dimensional tensors where each channel is a 2-D matrix

```
fig = plt.figure(figsize = (8,4))
ax1 = fig.add_subplot(1,2, 1)
ax1.imshow(cat)
ax2 = fig.add_subplot(1,2, 2)
ax2.imshow(dog)
```

```
<matplotlib.image.AxesImage at 0x2b3339f0908>
```



# Visualizing Image Data

Images are just n-dimensional tensors where each channel is a 2-D matrix

```
dog_r = dog.copy() # Red Channel  
dog_r[:, :, 1] = dog_r[:, :, 2] = 0 # set G,B pixels = 0  
dog_g = dog.copy() # Green Channel  
dog_g[:, :, 0] = dog_r[:, :, 2] = 0 # set R,B pixels = 0  
dog_b = dog.copy() # Blue Channel  
dog_b[:, :, 0] = dog_b[:, :, 1] = 0 # set R,G pixels = 0  
  
plot_image = np.concatenate((dog_r, dog_g, dog_b), axis=1)  
plt.figure(figsize = (10,4))  
plt.imshow(plot_image)
```

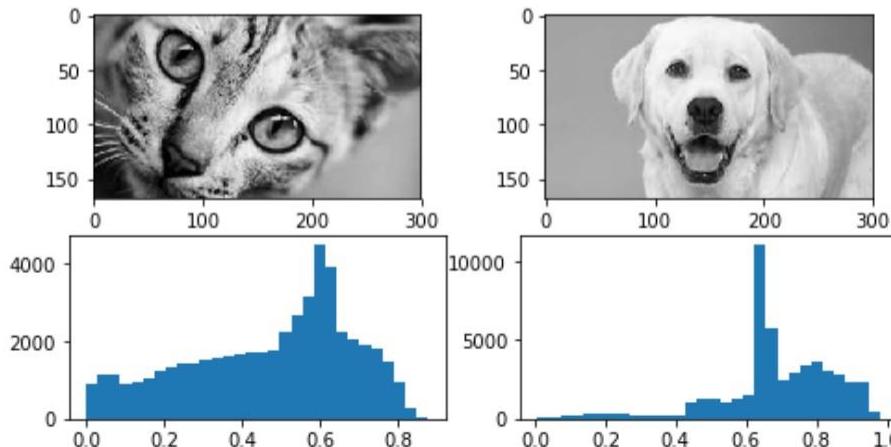
```
<matplotlib.image.AxesImage at 0x2b333b86240>
```



# Visualizing Image Data

Checking image intensity distributions

```
fig = plt.figure(figsize = (8,4))
ax1 = fig.add_subplot(2,2, 1)
ax1.imshow(cgs, cmap="gray")
ax2 = fig.add_subplot(2,2, 2)
ax2.imshow(dgs, cmap='gray')
ax3 = fig.add_subplot(2,2, 3)
c_freq, c_bins, c_patches = ax3.hist(cgs.flatten(), bins=30)
ax4 = fig.add_subplot(2,2, 4)
d_freq, d_bins, d_patches = ax4.hist(dgs.flatten(), bins=30)
```



# Visualizing Image Data

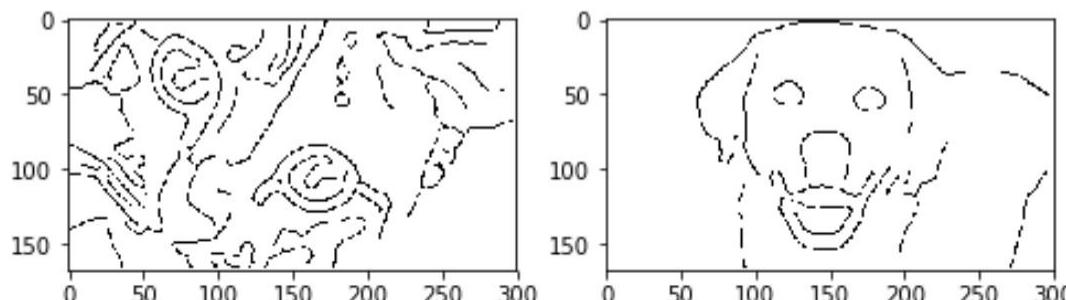
## Basic Edge Detection

```
from skimage.feature import canny

cat_edges = canny(cgs, sigma=3)
dog_edges = canny(dgs, sigma=3)

fig = plt.figure(figsize = (8,4))
ax1 = fig.add_subplot(1,2, 1)
ax1.imshow(cat_edges, cmap='binary')
ax2 = fig.add_subplot(1,2, 2)
ax2.imshow(dog_edges, cmap='binary')

<matplotlib.image.AxesImage at 0x2618e15c278>
```



# Visualizing Image Data

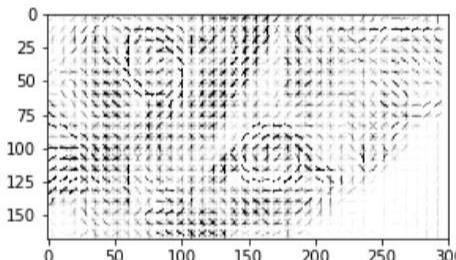
Histogram of Oriented Gradients - counts occurrences of gradient orientation in localized portions

```
from skimage.feature import hog
from skimage import exposure

fd_cat, cat_hog = hog(cgs, orientations=8, pixels_per_cell=(8, 8),
                      cells_per_block=(3, 3), visualise=True)
fd_dog, dog_hog = hog(dgs, orientations=8, pixels_per_cell=(8, 8),
                      cells_per_block=(3, 3), visualise=True)

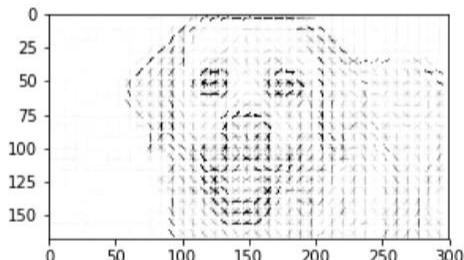
# rescaling intensity to get better plots
cat_hogs = exposure.rescale_intensity(cat_hog, in_range=(0, 0.04))
dog_hogs = exposure.rescale_intensity(dog_hog, in_range=(0, 0.04))

fig = plt.figure(figsize = (10,4))
ax1 = fig.add_subplot(1,2, 1)
ax1.imshow(cat_hogs, cmap='binary')
ax2 = fig.add_subplot(1,2, 2)
ax2.imshow(dog_hogs, cmap='binary')
```



```
print(fd_cat, fd_cat.shape)
```

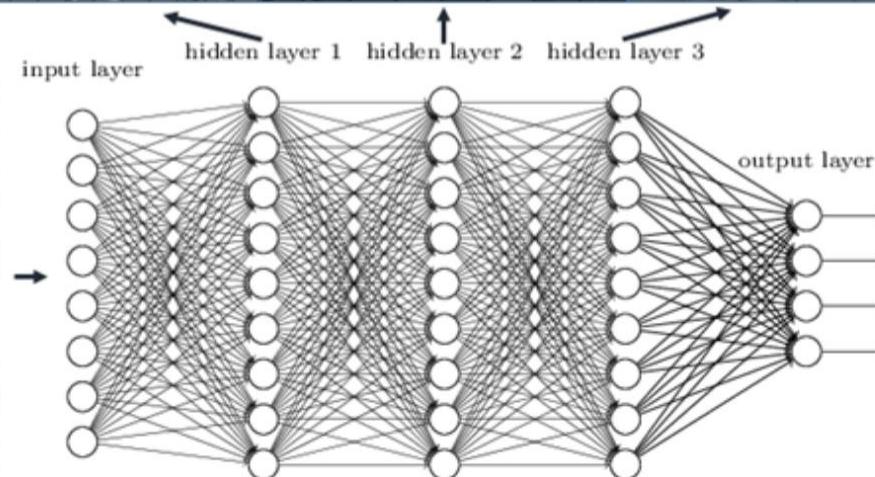
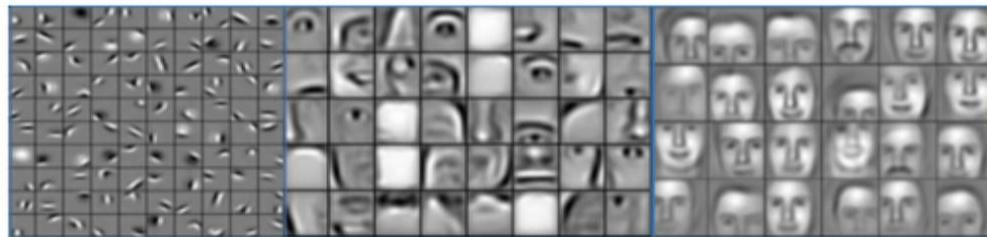
```
[ 0.00288784  0.00301086  0.0255757 ... , 0.          0.          0.          ] (47880,)
```



# Visualizing Image Data - CNNs changed the world!

Automated image representation with deep learning

Deep neural  
networks learn  
hierarchical feature  
representations



# Visualizing Audio Data - Can you see what you hear?

Audio data can be represented based on the synthesized signals

The screenshot shows the homepage of the Urban Sound Datasets website. The header includes a navigation bar with links for "HOME", "URBANSOUND", "URBANSOUND8K", "TAXONOMY", and "PUBLICATIONS". The main title "URBAN SOUND DATASETS" is prominently displayed in large white letters across the center of the page. Below the title, a subtitle reads "TWO DATASETS AND A TAXONOMY FOR URBAN SOUND RESEARCH". A "READ MORE" button is located in the lower center. At the bottom of the page, the URL "https://urbansounddataset.weebly.com/" is displayed.

URBAN SOUND DATASETS

HOME URBANSOUND URBANSOUND8K TAXONOMY PUBLICATIONS

# URBAN SOUND DATASETS

TWO DATASETS AND A TAXONOMY FOR URBAN SOUND RESEARCH

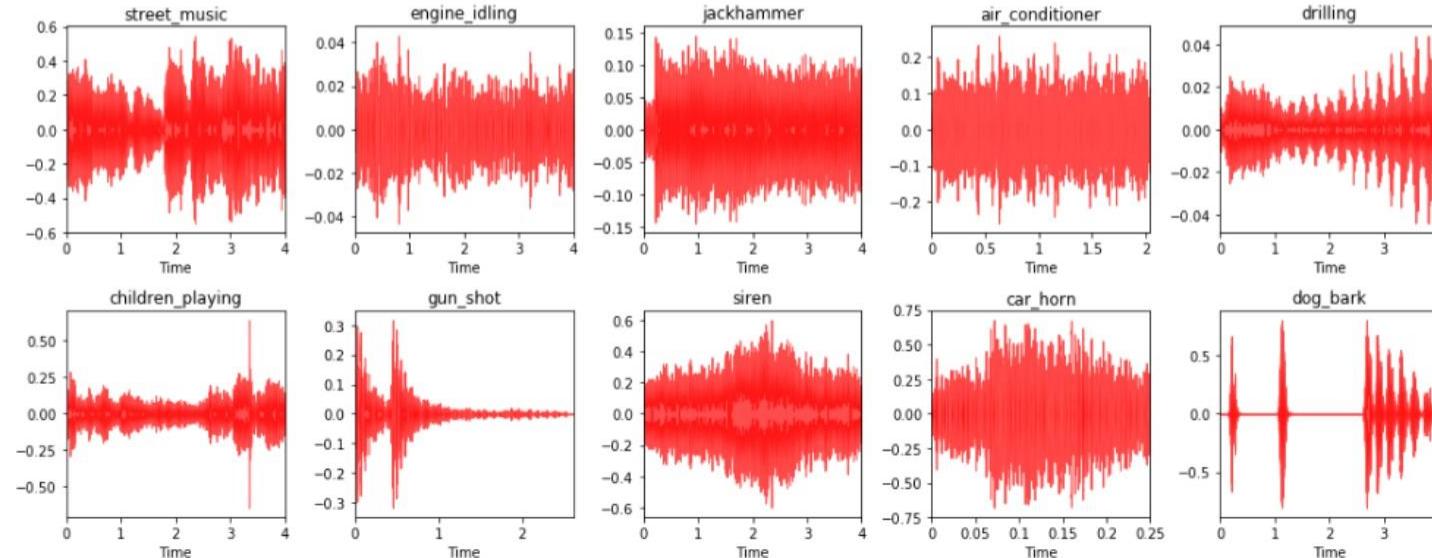
READ MORE

<https://urbansounddataset.weebly.com/>

# Visualizing Audio Data - Can you see what you hear?

Visualizing audio with wave\amp plots

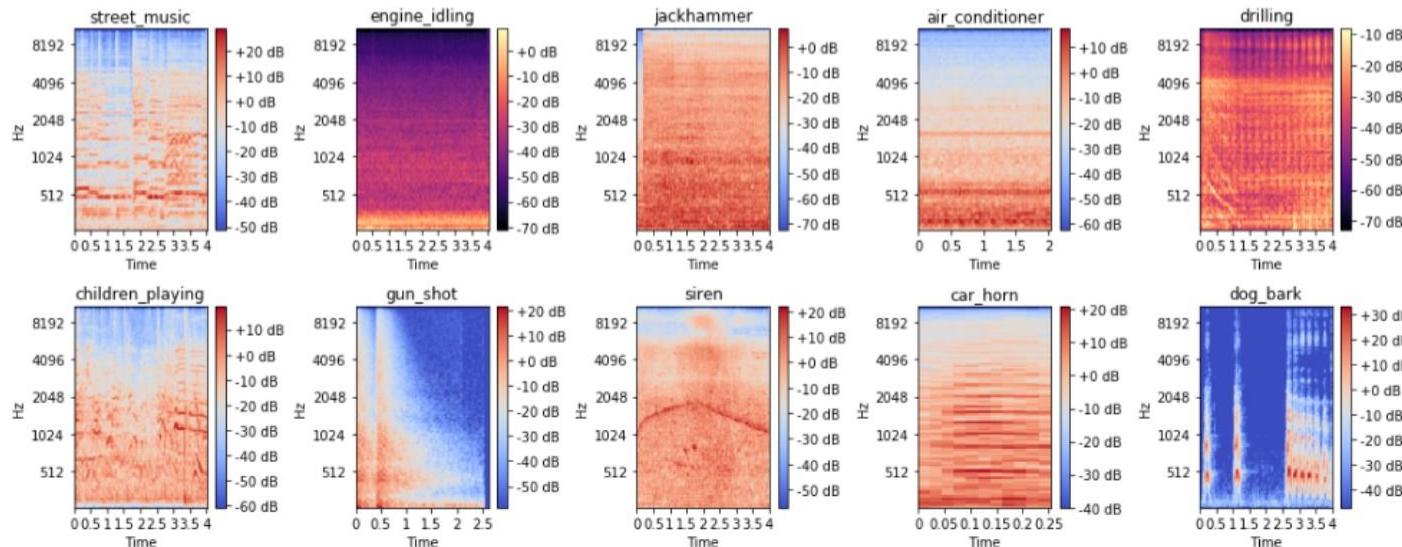
```
i = 1
fig = plt.figure(figsize=(15, 6))
for item in sample_data:
    plt.subplot(2, 5, i)
    librosa.display.waveplot(item[1][0], sr=item[1][1], color='r', alpha=0.7)
    plt.title(item[0])
    i += 1
plt.tight_layout()
```



# Visualizing Audio Data - Can you see what you hear?

Visualizing audio with mel-spectrograms

```
for item in sample_data:  
    plt.subplot(2, 5, i)  
    S = librosa.feature.melspectrogram(item[1][0], sr=item[1][1], n_mels=128)  
    log_S = librosa.logamplitude(S)  
    librosa.display.specshow(log_S, sr=item[1][1], x_axis='time', y_axis='mel')  
    plt.title(item[0])  
    plt.colorbar(format='%+02.0f dB')  
    i += 1  
plt.tight_layout()
```



# Visualizing Audio Data - Can you see what you hear?

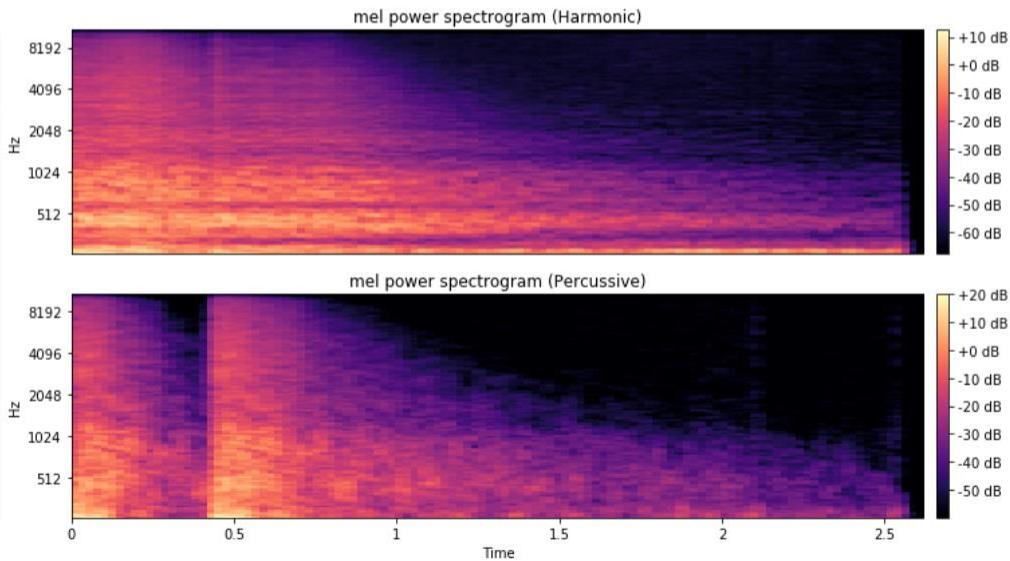
Visualizing audio with spectrograms from harmonic & percussive components

```
y_harmonic, y_percussive = librosa.effects.hpss(y)
S_harmonic    = librosa.feature.melspectrogram(y_harmonic, sr=22050,
                                              n_mels=128)
S_percussive = librosa.feature.melspectrogram(y_percussive, sr=22050)
log_Sh = librosa.power_to_db(S_harmonic)
log_Sp = librosa.power_to_db(S_percussive)

# Make a new figure
plt.figure(figsize=(12,6))

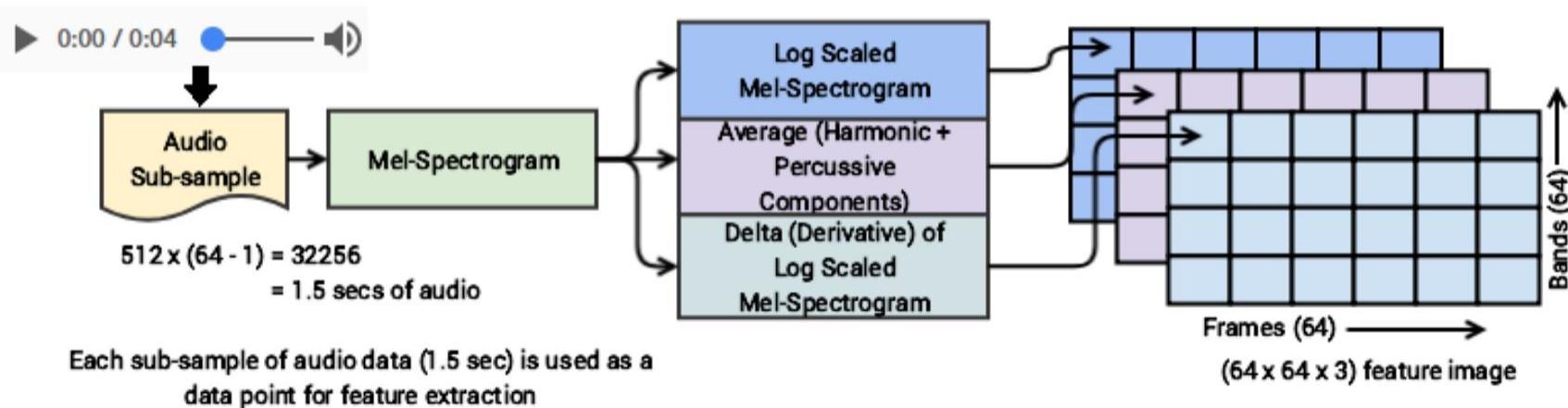
plt.subplot(2,1,1)
librosa.display.specshow(log_Sh, sr=sr, y_axis='mel')
plt.title('mel power spectrogram (Harmonic)')
plt.colorbar(format='%.+02.0f dB')

plt.subplot(2,1,2)
librosa.display.specshow(log_Sp, sr=sr, x_axis='time',
                        y_axis='mel')
plt.title('mel power spectrogram (Percussive)')
plt.colorbar(format='%.+02.0f dB')
plt.tight_layout()
```



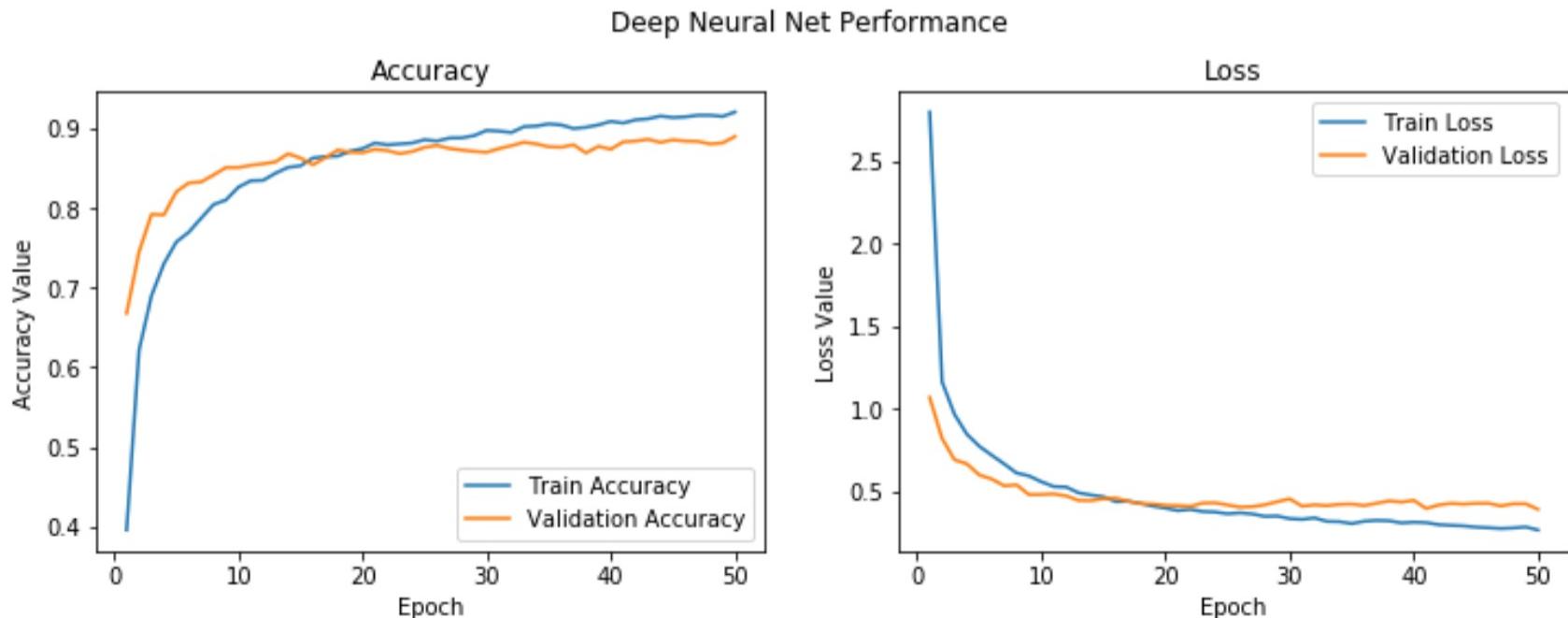
# Visualizing Audio Data - Why we do it?

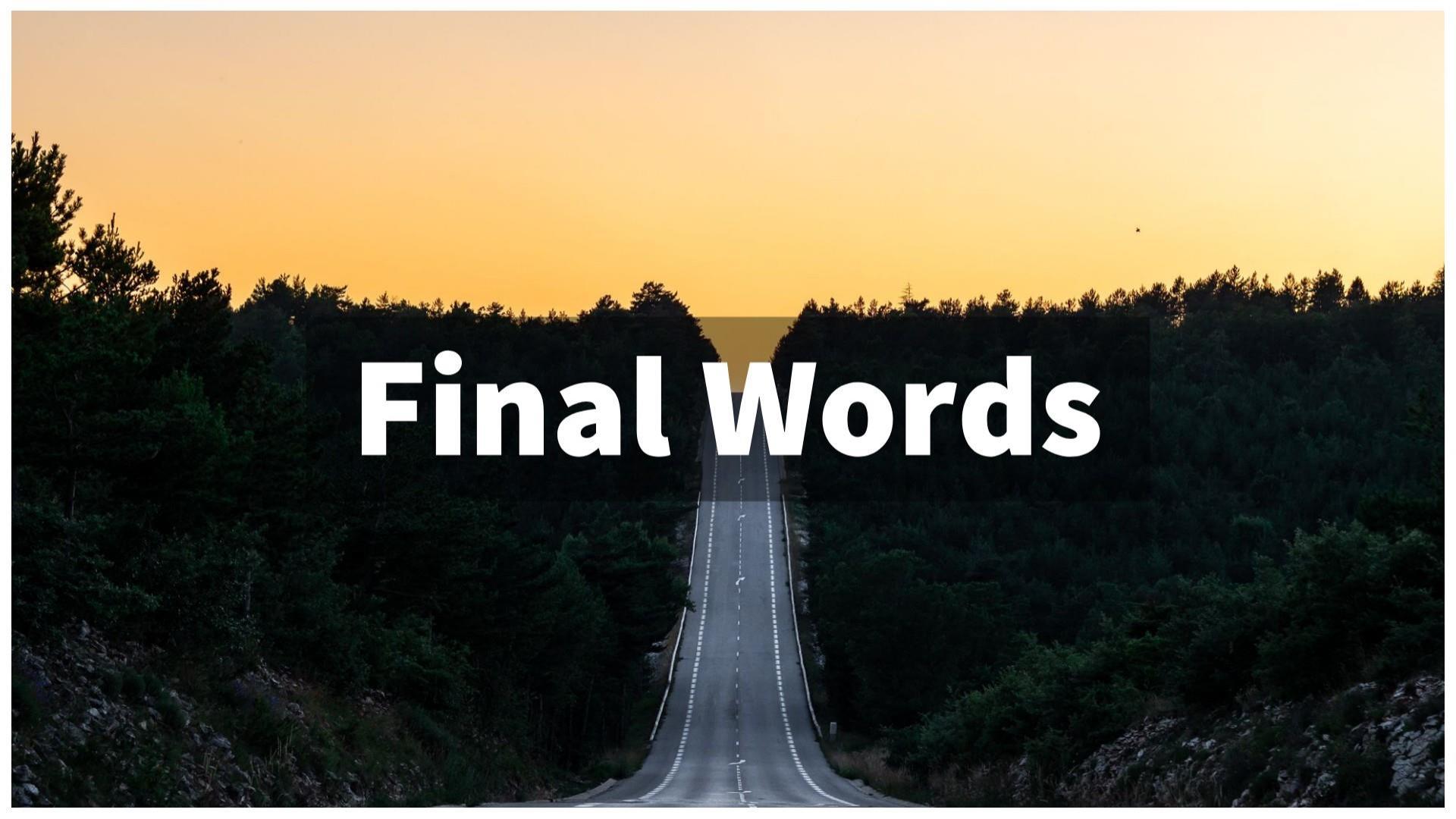
Visual depiction & Feature engineering for Deep Learning Models



# Visualizing Audio Data - Why we do it?

Audio Classification with audio data + pre-trained image models (e.g VGG\Inception)

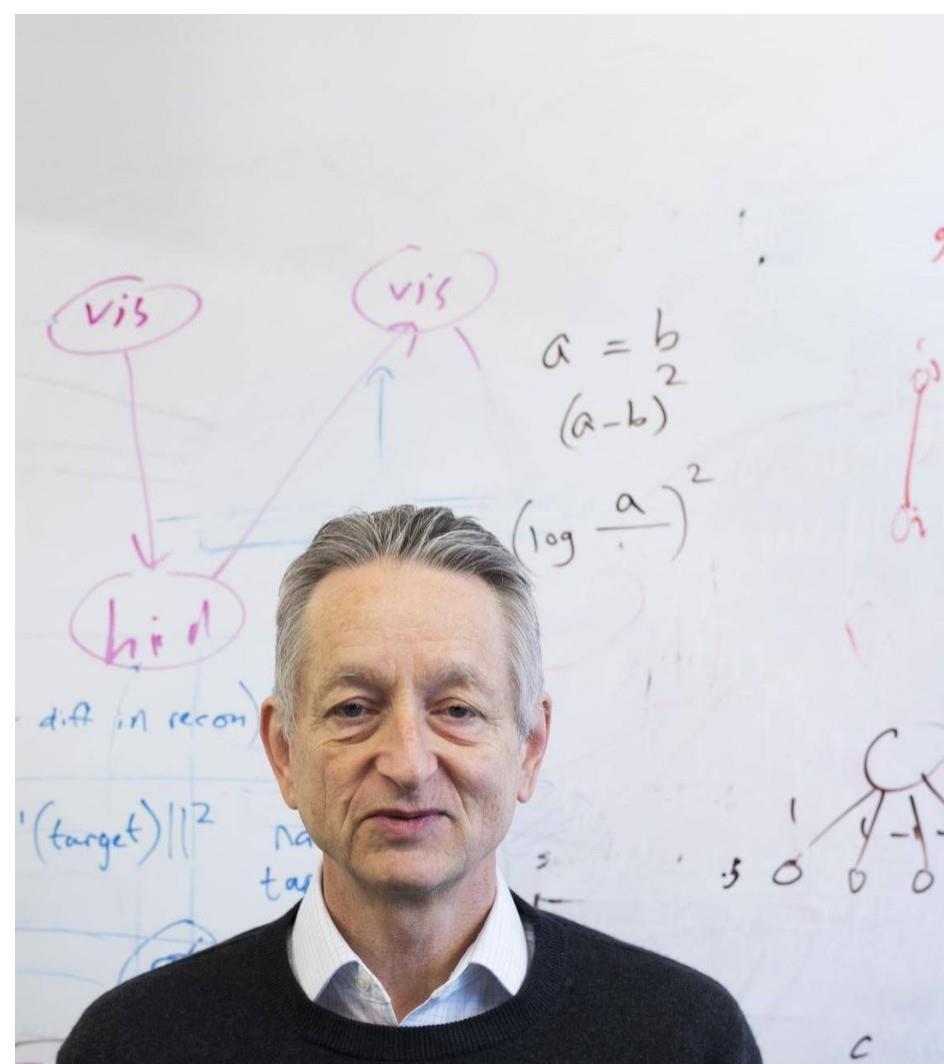


A photograph of a two-lane asphalt road curving through a dense forest. The road is marked with white dashed lines. The sky above is a warm, golden-yellow color, suggesting either sunrise or sunset. The foreground and sides of the road are filled with dark green trees and shrubs.

# Final Words

# What if we have a lot of features?

- Use domain expertise to focus on important features
- Use modeling to get the most important features and visualize to see if they really make sense
- Dimension reduction techniques including PCA, SVD, t-SNE
- Remember to scale features and remove outliers as needed



**“If you want to visualize 17 dimensions, visualize 3 dimensions and say 17 over and over again. That's how everybody does it.”**

**-Geoffrey Hinton**

# References and Content

## ○ Article Links

<https://towardsdatascience.com/a-comprehensive-guide-to-the-grammar-of-graphics-for-effective-visualization-of-multi-dimensional-1f92b4ed4149>

<https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57>

<https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa>

<https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72>

## ○ Papers

<http://vita.had.co.nz/papers/layered-grammar.pdf>

## ○ Code & Examples

[https://github.com/dipanjanS/art\\_of\\_data\\_visualization](https://github.com/dipanjanS/art_of_data_visualization)

<https://seaborn.pydata.org/>

<https://plotnine.readthedocs.io/en/stable/>

<https://github.com/d3/d3/wiki/Gallery>

# Contact Me

Interested in collaborating, research or writing?



LinkedIn

<https://www.linkedin.com/in/dipanzan>



GitHub

<https://github.com/dipanjanS>

Towards  
Data Science

Towards Data Science

<https://towardsdatascience.com/@dipanzan.sarkar>