Altair Exercises

This notebook will explore multiple different visualizations in Altair.

Part 5

The following exercise is with artwork created by Bob Ross (https://en.wikipedia.org/wiki/Bob_Ross). Bob was a very famous painter who had a televised painting show from 1983 to 1994. Over 13 seasons and approximately 400 paintings, Bob would walk the audience through a painting project. Often these were landscape images. Bob was famous for telling his audience to paint "happy trees" and sayings like, "We don't make mistakes, just happy little accidents." His soothing voice and bushy hair are well known to many generations of viewers.

We'll be starting with the dataset created by 538 for their article on a Statistical Analysis of Bob Ross (https://fivethirtyeight.com/features/a-statistical-analysis-of-the-work-of-bob-ross/). The authors of the article coded each painting to indicate what features the image contained (e.g., one tree, more than one tree, what kinds of clouds, etc.).

In [1]:
```
import zipfile as zip
import urllib.request
import os.path
from os import path
import pandas as pd
import altair as alt
import numpy as np
from sklearn import manifold
from sklearn.metrics import euclidean_distances
from sklearn.decomposition import PCA
import ipywidgets as widgets
from IPython.display import display
from PIL import Image
```

Load dataset

In [2]:
```
# the paints Bob used
rosspaints = ['alizarin crimson','bright red','burnt umber','cadmium y
              'indian yellow','indian red','liquid black','liquid clea
              'midnight black','phthalo blue','phthalo green','prussia
              'titanium white','van dyke brown','yellow ochre']

# hex values for the paints above
rosspainthex = ['#94261f','#c06341','#614f4b','#f8ed57','#5c2f08','#e0
                '#000000','#ffffff','#000000','#36373c','#2a64ad','#21
                '#364e00','#f9f7eb','#2d1a0c','#b28426']

# boolean features about what an image includes
imgfeatures = ['Apple frame', 'Aurora borealis', 'Barn', 'Beach', 'Boa
               'Bridge', 'Building', 'Bushes', 'Cabin', 'Cactus',
               'Circle frame', 'Cirrus clouds', 'Cliff', 'Clouds',
               'Coniferous tree', 'Cumulis clouds', 'Decidious tree',
               'Diane andre', 'Dock', 'Double oval frame', 'Farm',
```

```
                'Fence', 'Fire', 'Florida frame', 'Flowers', 'Fog',
                'Framed', 'Grass', 'Guest', 'Half circle frame',
                'Half oval frame', 'Hills', 'Lake', 'Lakes', 'Lighthous
                'Mill', 'Moon', 'At least one mountain', 'At least two
                'Nighttime', 'Ocean', 'Oval frame', 'Palm trees', 'Path
                'Person', 'Portrait', 'Rectangle 3d frame', 'Rectangula
                'River or stream', 'Rocks', 'Seashell frame', 'Snow',
                'Snow-covered mountain', 'Split frame', 'Steve ross',
                'Man-made structure', 'Sun', 'Tomb frame', 'At least or
                'At least two trees', 'Triple frame', 'Waterfall', 'Wav
                'Windmill', 'Window frame', 'Winter setting', 'Wood fra

# load the data frame
bobross = pd.read_csv("../assets/bobross.csv")

# enable correct rendering (unnecessary in later versions of Altair)
alt.renderers.enable('default')

# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

Out[2]:
```
DataTransformerRegistry.enable('json')
```

We have a few variables defined for you that you might find useful for the rest of this exercise. First is the `bobross` dataframe which, has a row for every painting created by Bob (we've removed those created by guest artists).

In [3]:
```
bobross.sample(5)
```

Out[3]:

	EPISODE	TITLE	RELEASE_DATE	Apple frame	Aurora borealis	Barn	Beach	Boat	Bridge	B
244	S21E01	"VALLEY VIEW"	9/5/90	0	0	0	0	0	0	
371	S31E03	"WINDING STREAM"	3/8/94	0	0	0	0	0	0	
237	S20E07	"AUTUMN FANTASY"	5/16/90	0	0	0	0	0	0	
57	S05E13	"MEADOW STREAM"	3/27/85	0	0	1	0	0	0	
61	S06E04	"WHISPERING STREAM"	5/22/85	0	0	0	0	0	0	

5 rows × 114 columns

In the dataframe you will see an episode identifier (EPISODE, which contains the season and episode number), the image title (TITLE), the release date (RELEASE_DATE as well as another column for the year). There are also a number of boolean columns for the features coded by FiveThirtyEight. A '1' means the feature is present, a '0' means it is not. A list of those columns is available in the `imgfeatures` variable.

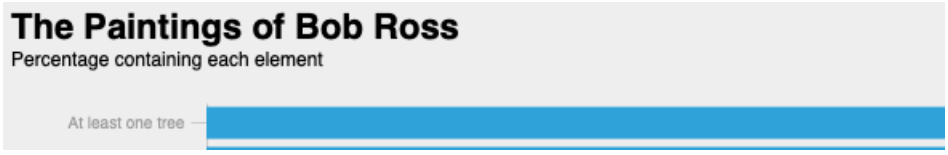
Bob Ross Bar Chart

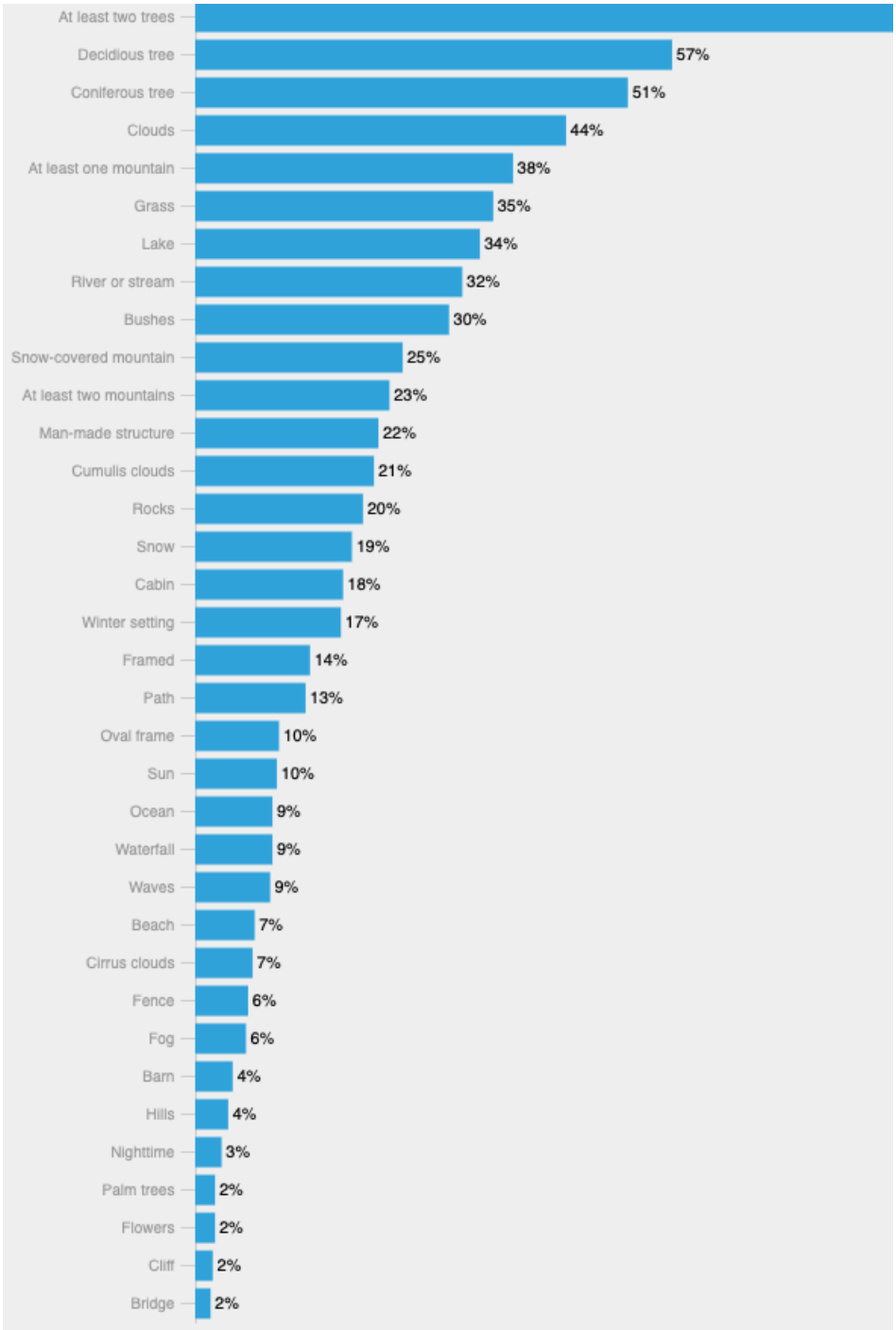
We're going to recreate the first chart from the Bob Ross article (assets/bob_ross_538.png) (source: Statistical Analysis of Bob Ross (https://fivethirtyeight.com/features/a-statistical-analysis-of-the-work-of-bob-ross/)). This one simply shows a bar chart for the percent of images that have certain features.

In [4]:
```
def makeBobRossBar():
    data = bobross[imgfeatures]
    data = data.sum() / len(data)
    data = data.to_frame()
    data.reset_index(inplace=True)
    data.rename(columns={'index' : 'feature', 0 : 'pct'}, inplace=True
    data.sort_values(by='pct',ascending=False,inplace=True)
    data = data.head(36) # both visualizations show top 36 features
    vals = list(data['feature'])

    #     return data

    barsq1 = alt.Chart(data).mark_bar(size=20).encode(
        x=alt.X('pct',
                axis=None),
        y=alt.Y('feature:N',
                axis=alt.Axis(tickCount=5, title=''),
                sort=vals
               )
    )

    # barsq1

    textq1 = barsq1.mark_text(
        align='left',
        baseline='middle',
        dx=3
    ).encode(
        text=alt.Text('pct:Q', format=',.0%')
    )

    # textq1

    bobross_features = (barsq1 + textq1).configure(
        background='#eeeeee',
        padding=5
    ).configure_axis(
        labelFontSize=10,
        labelFont='Helvetica',
        labelOpacity=1
    ).configure_mark(
        color='#008fd5'
    ).configure_view(
        strokeWidth=0
    ).configure_scale(
        bandPaddingInner=0.1
    ).configure_title(
        anchor='start',
        font='Helvetica',
        fontSize=22,
        fontWeight='bold',
        offset=20
    ).properties(
        width=500,
        height=900
    ).properties(
        title={"text":"The Paintings of Bob Ross",
               "subtitle" : ["Percentage containing each element"]}
    )

    return bobross_features
```

In [5]:
```
alt.themes.enable('fivethirtyeight')
makeBobRossBar()
```

Out[5]:





Conditional Probabilities

The 538 article (Statistical Analysis of Bob Ross (https://fivethirtyeight.com/features/a-statistical-analysis-of-the-work-of-bob-ross/)) has a long analysis of conditional probabilities. Essentially, we want to know the probability of one feature given another (e.g., what is the probability of Snow given Trees?). The article calculates this over the entire history of the show, but we would like to visualize these probabilities over time. Have they been constant? or evolving? We will only be doing this for a few variables (otherwise, we'll have a matrix of over 3000 small charts). Specifically, we care about images that contain: 'At least one tree','At least two trees','Clouds','Grass','At least one mountain','Lake.' Each small multiple plot will be a line chart corresponding to the conditional probability over time. The matrix "cell" indicates which pairs of variables are being considered (e.g., probability of at least two trees given the probability of at least one tree is the 2nd row, first column in our example).

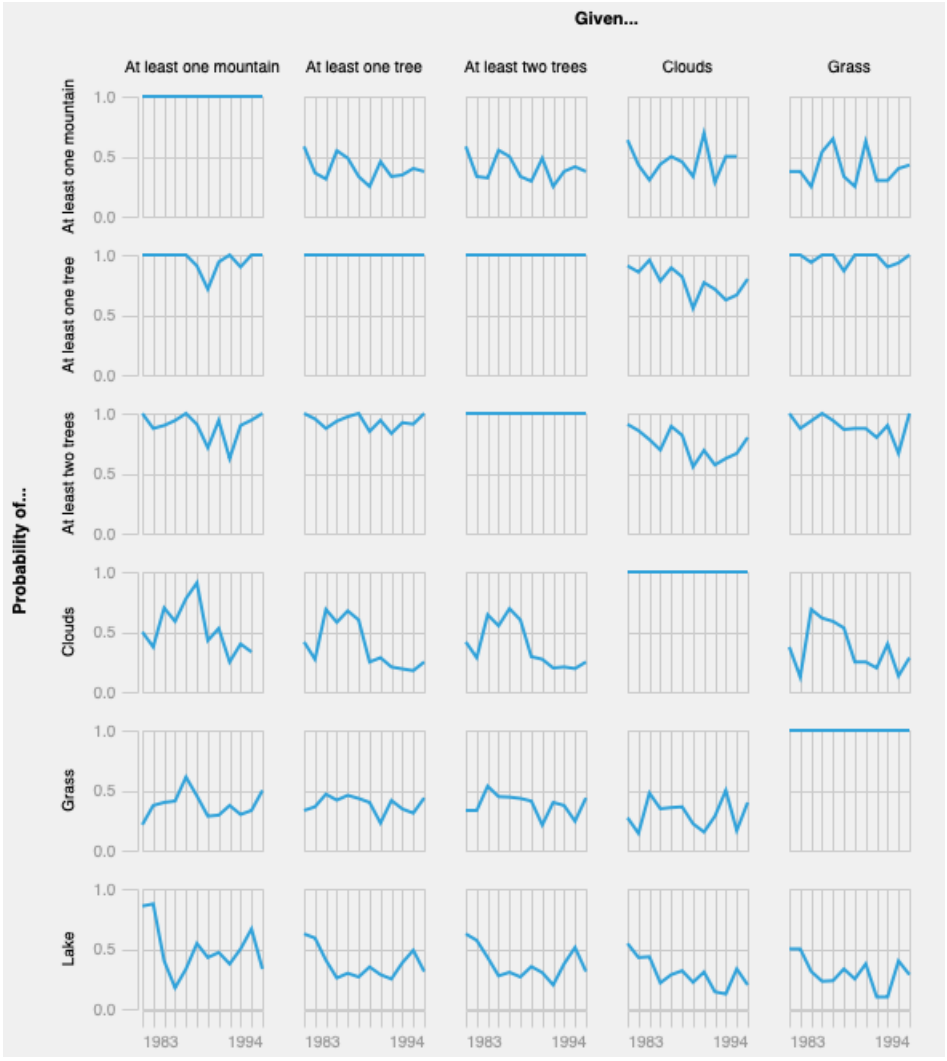
In [6]:
```
def condprobability(frame,column1,column2,year):
    # we suggest you implement this function to make your life easier.
    # the two columns we want the conditional probability for, and the
    # you can make variants of this function as you see fit


    df = frame[frame['year'] == year]

    if column1 == column2:
        cond_prob = 1.0
    else:
        probs = df.groupby(column2).size().div(len(df))
        df = df.groupby([column2, column1]).size().div(len(df)).div(pr
        cond_prob = float(df.xs(1, level=0, axis=0, drop_level=False).

    return [column1, column2, year, cond_prob]
```

In [7]:
```
def makeBobRossCondProb(totest=['At least one tree','At least two tree
    # implement this function to return an altair chart
    # note that we have created a default 'totest' variable that has t
    # we want the pairwise analysis

    # generate and format data table

    df = {}
    count=0
    for yr in list(bobross['year'].unique()):
        for c1 in totest:
            for c2 in totest:
                try:
                    df[count] = condprobability(bobross,c1,c2,yr)
                    count+=1
                except:
                    pass

    df = pd.DataFrame(data=df).T
    df.rename(columns={0:'key1',1:'key2',2:'year',3:'prob'}, inplace=T
    df['year'] = pd.to_datetime((df['year']).apply(str), format='%Y')

    # generate visualization
    line_charts = alt.Chart(df).transform_fold(
        ['key1','key2'], as_=['key','value']
    ).mark_line().encode(
        x=alt.X('year(year):T',
                title=None,
                axis=alt.Axis(grid=True, tickCount=12)
               ),
        y=alt.Y('prob:Q',
                title=None
               )
    ).properties(
        width=75,
```

```
        height=75
    ).facet(
        column=alt.Column('key2:N',header=alt.Header(labelOrient='top'
        row=alt.Row('key1:N', title="Probability of...", sort='ascend
    )


    return line_charts
```

In [8]:
```
makeBobRossCondProb()
```

Out[8]:



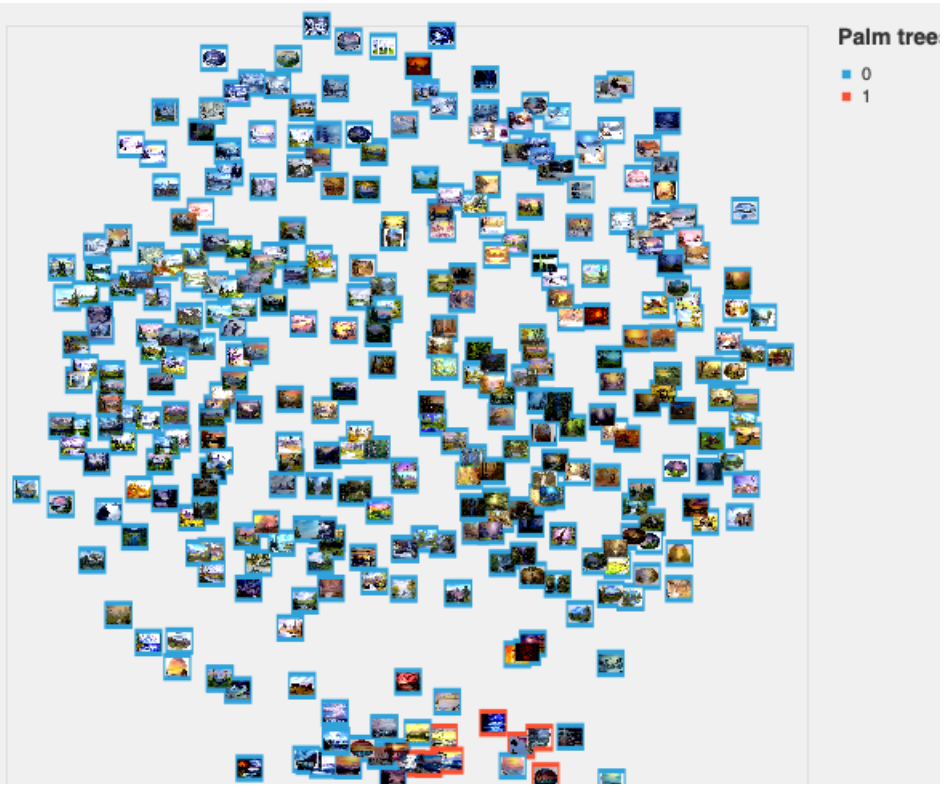
Create 2D MDS plot

We are going to create an interactive widget that allows you to select the feature you want to be highlighted. The plot should change when you select new items from the list.

In [9]:
```
# create the seed
seed = np.random.RandomState(seed=3)

# generate the MDS configuration, we want 2 components, etc. You can t
# the settings change the layout
mds = manifold.MDS(n_components=2, max_iter=3000, eps=1e-9, random_sta

# fit the data. At the end, 'pos' will hold the x,y coordinates
pos = mds.fit(bobross[imgfeatures]).embedding_

# we'll now load those values into the bobross data frame, giving us a
bobross['x'] = [x[0] for x in pos]
bobross['y'] = [x[1] for x in pos]
```

In [10]:
```
def genMDSPlot(key):
    # return an altair chart (e.g., return alt.Chart(...))
    # key is a string indicating which images should be visually highl
    # should be made salient)

#     source = bobross[['x', 'y', 'img_url']]

    imgs = alt.Chart(bobross).mark_image(
        width=15,
        height=15,
    ).encode(
        x=alt.X('x', axis=None),
        y=alt.Y('y', axis=None),
        url='img_url'
    )

    borders = alt.Chart(bobross).mark_square(size=300,opacity=1).encod
        x=alt.X('x', axis=None),
        y=alt.Y('y', axis=None),
        color=alt.Color(str(key+':N')))


    plot = (borders + imgs).properties(width=500,height=500)

    return plot

genMDSPlot('Palm trees')
```

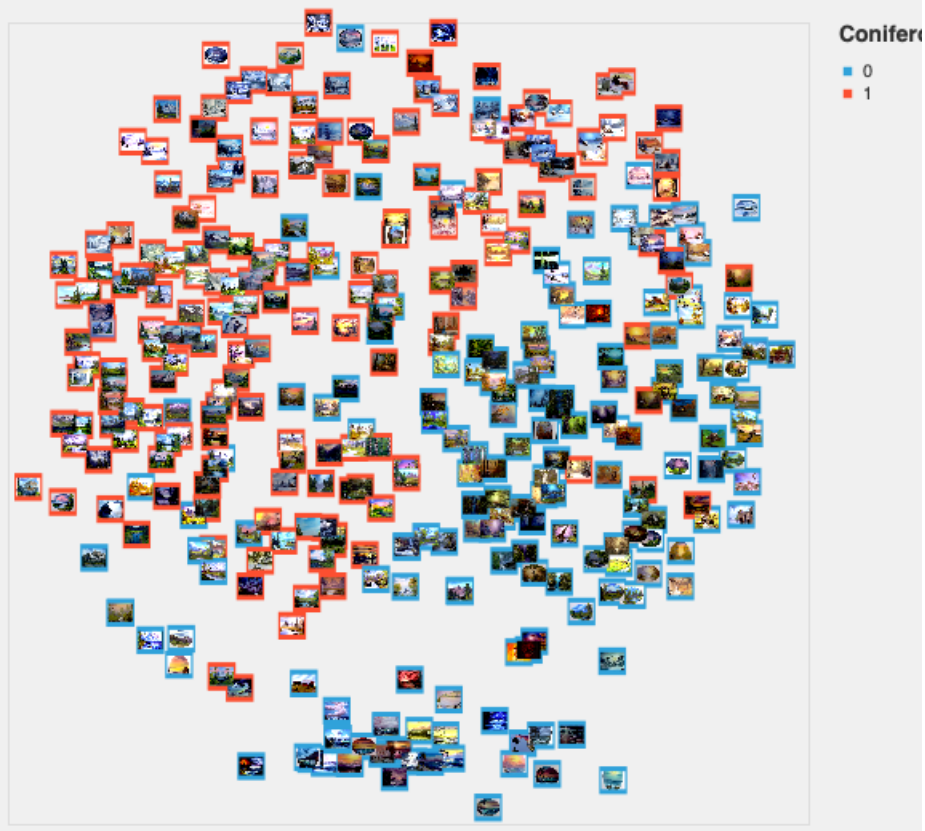
Out[10]:



Dropdown Capabilities:

In [11]:
```
# note that it might take a few seconds for the images to download
# depending on your internet connection

output = widgets.Output()

def clicked(b):
    output.clear_output()
    with output:
        highlight = filterdrop.value
        if (highlight == ""):
            print("please enter a query")
        else:
            genMDSPlot(highlight).display()


featurecount = bobross[imgfeatures].sum()

filterdrop = widgets.Dropdown(
    options=list(featurecount[featurecount > 2].keys()),
    description='Highlight:',
    disabled=False,
)

filterdrop.observe(clicked)

display(filterdrop,output)

with output:
    genMDSPlot('Barn').display()
```

Highlight: Coniferous tree



Exercise adapted and modified from UMSI homework assignment for SIADS 622.