In [1]:
```
import pandas as pd
import numpy as np
from scipy.stats import f_oneway, ranksums

from saac.evaluation.eval_utils import rgb_sorter, rgb_intensity, generate_histplot
from saac.statistics import ks2sample_test

%matplotlib inline
```

In [2]:
```
respath='./data/processed/'
```

Trait Sentiment (TDA) Evaluation

In [3]:
```
tda_res_all = pd.read_csv(respath+'TDA_Results.csv' )
print(f'Total rows: {len(tda_res_all)}')

sentcheck = tda_res_all[tda_res_all['tda_compound']==tda_res_all['compound']]
print(f'Total rows where tda sentiment is equal to prompt sentiment : {len(sentchec

print('Counts of sampled sentiment categories for all possible gender detected valu
sentiment_order = ['very negative', 'negative', 'neutral','positive','very positive
gender_order = ['man', 'woman', 'unknown','no face']
pd.crosstab(tda_res_all['gender_detected_val'], tda_res_all['tda_sentiment_val']).r
```

```
Total rows: 1483
Total rows where tda sentiment is equal to prompt sentiment : 1483
Counts of sampled sentiment categories for all possible gender detected values
```

Out[3]:

tda_sentiment_val	very negative	negative	neutral	positive	very positive
gender_detected_val					
man	149	121	121	82	95
woman	107	141	132	172	175
unknown	7	7	4	11	2
no face	30	31	38	31	27

In [4]:
```
tda_res = tda_res_all[~tda_res_all['gender_detected_val'].isin(['unknown','no face'
print(f"Total rows after removing faceless and unknown gender detected results: {le
```

```
Total rows after removing faceless and unknown gender detected results: 1295
```

Trait (TDA) Sentiment by Detected Gender

Two Sample Kolmogorov-Smirnov Test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html

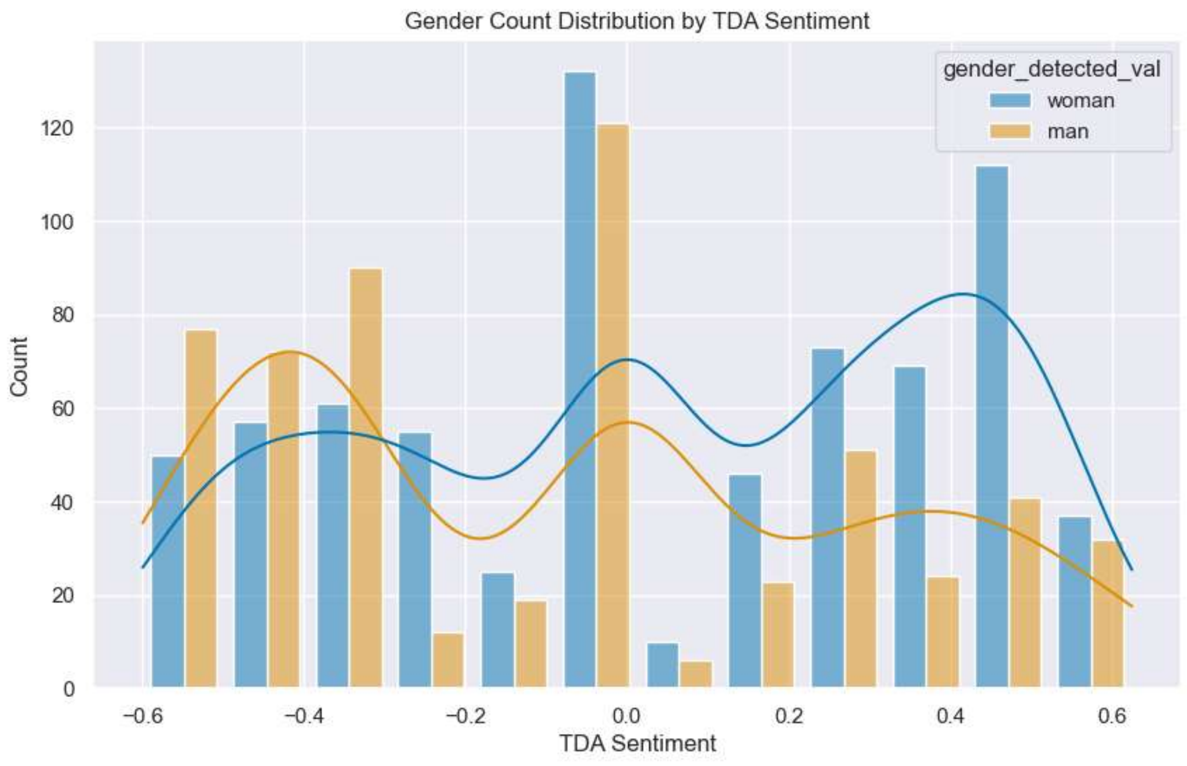
Using the default two-sided parameter for alternative, the null hypothesis is that the two distributions are identical and the alternative is that they are not identical.

If the p-value is lower than our confidence level of 95%, we can reject the null hypothesis in favor of the alternative and conclude that the data were not drawn from the same distribution.

```
In [5]: t = [x for x in ks2sample_test(tda_res, group_col='gender_detected_val', value_col=
        t
```

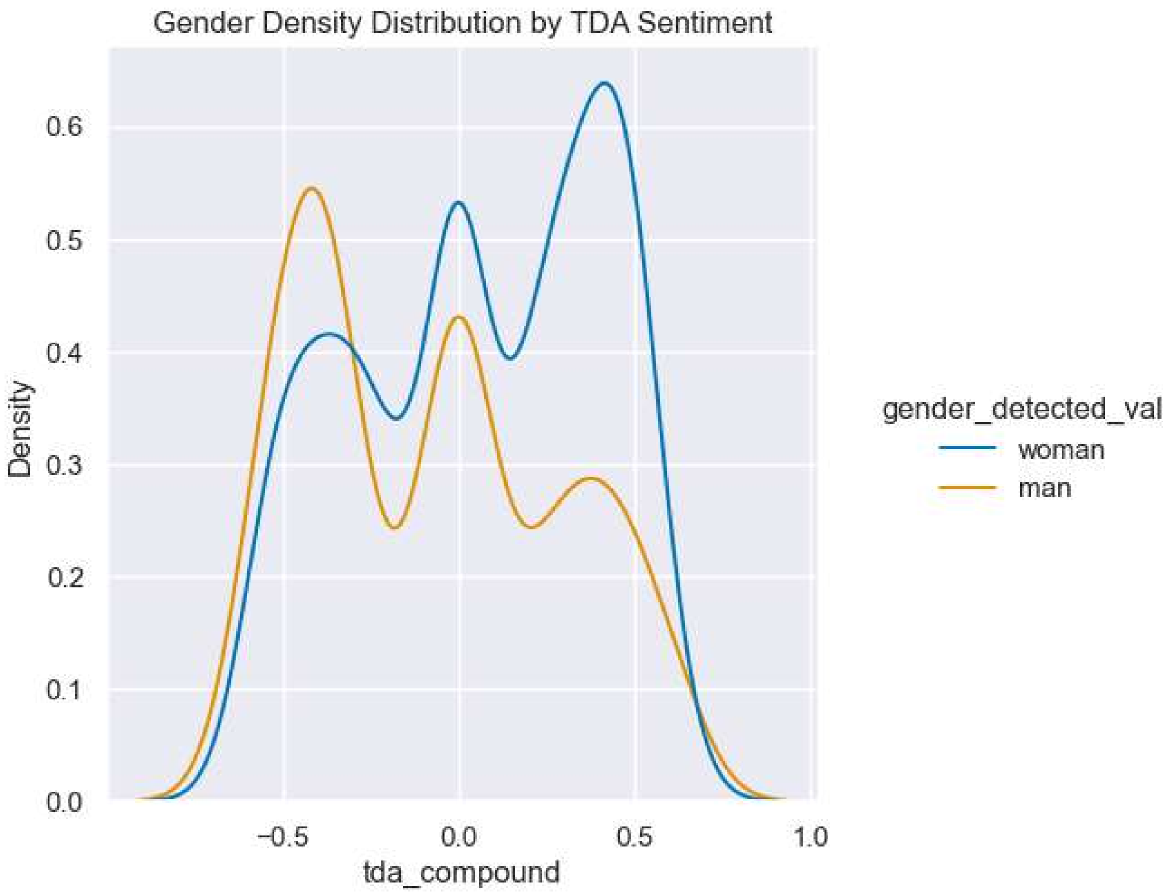
```
Out[5]: [{'group1': 'man',
          'group2': 'woman',
          'statistic': 0.19304686440513785,
          'pvalue': 7.02933607694096e-11}]
```

```
In [6]: tda_hist = generate_histplot(tda_res,
                           'tda_compound',
                           'gender_detected_val',
                            hue_order = ['woman', 'man'],
                            title = 'Gender Count Distribution by TDA Sentiment',
                            xlabel = 'TDA Sentiment',
                            ylabel = 'Count')
```



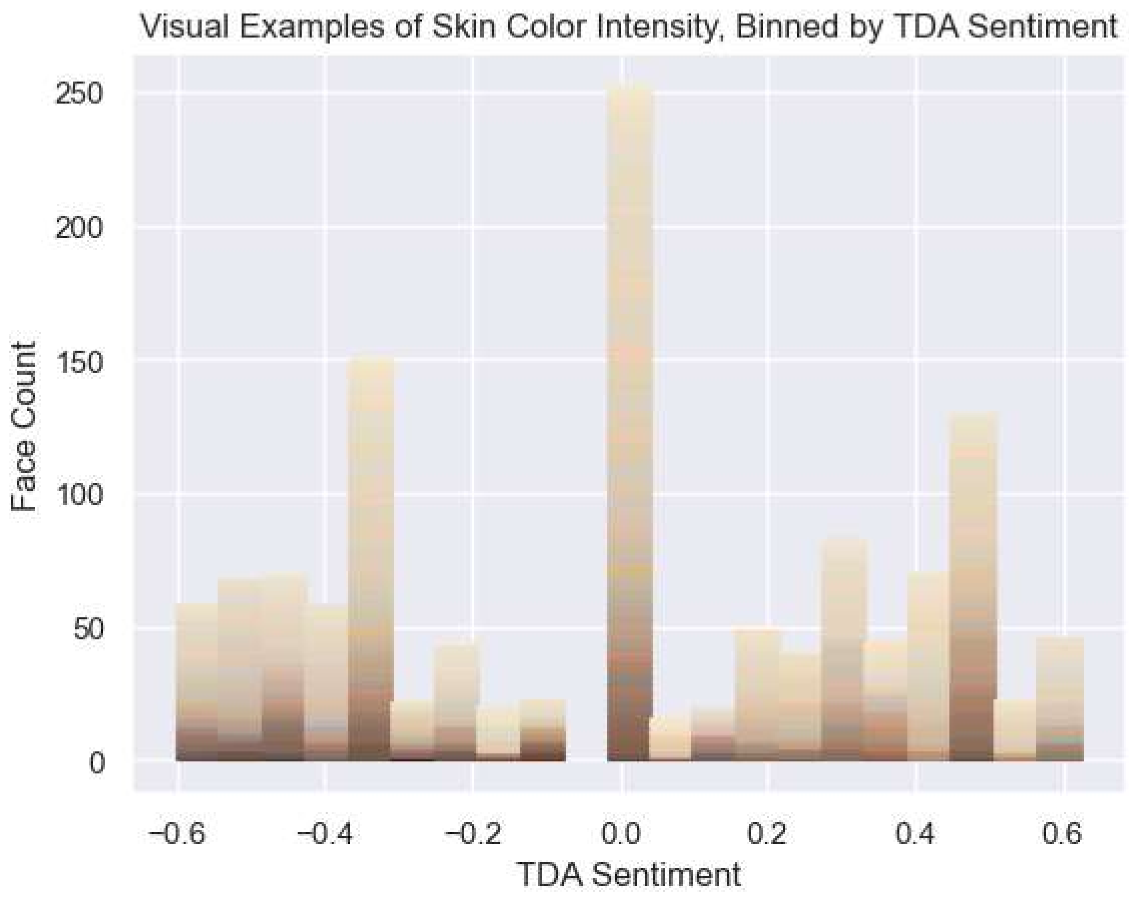
```
In [7]: tda_dis = generate_displot(tda_res,
                        'tda_compound',
                        'gender_detected_val',
                        kind = 'kde',
                        title = 'Gender Density Distribution by TDA Sentiment')
```

```
<Figure size 1000x600 with 0 Axes>
```

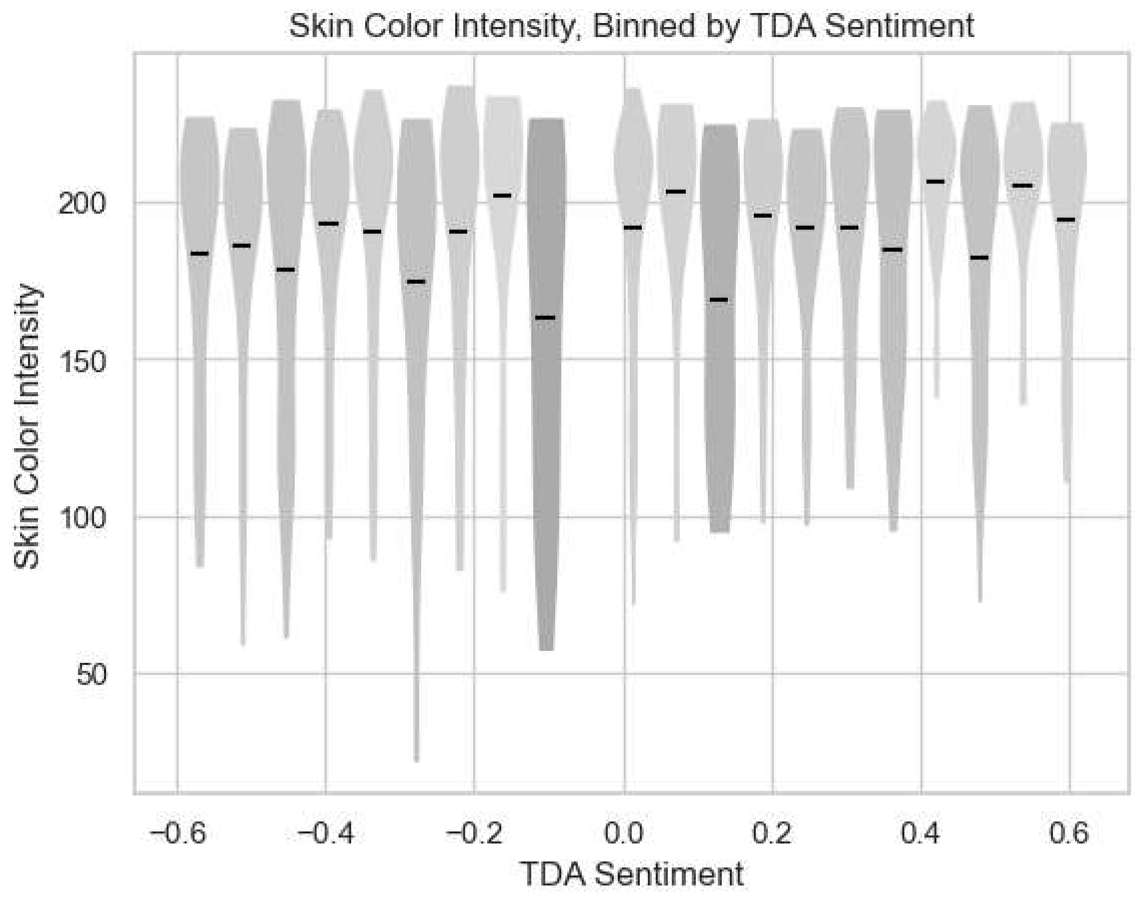


Trait Sentiment by Skin Color and Intensity(Lumia)

```
In [8]: ## Visual demonstration of intensity sorted by trait sentiment bin

        tda_rgb = rgb_histogram(tda_res,'tda_compound',
                      'skin color',
                      n_bins=21,
                      x_label='TDA Sentiment',
                      y_label='Face Count',
                      title='Visual Examples of Skin Color Intensity, Binned by TDA Sentime
```



```
In [9]: # Violin plots of skin intensity per trait sentiment bin

        tda_vp = lumia_violinplot(df=tda_res,
        x_col = 'tda_compound',
        rgb_col = 'skin color',
        n_bins = 21,
        widths_val = 0.05,
        points_val = 100,
        x_label = 'TDA Sentiment',
        y_label = 'Skin Color Intensity',
        title = 'Skin Color Intensity, Binned by TDA Sentiment')
```



One-way ANOVA Test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

The one-way ANOVA tests the null hypothesis that two or more groups have the same population mean.

In [10]:
```
#Getting all rgb intensities
n_bins=21
tda_count, tda_division = np.histogram(tda_res['tda_compound'], bins=n_bins)

all_rgb_intensities = []

for idx in range(1, len(tda_division)):
    if idx + 1 == len(tda_division):
        mask = (tda_res['tda_compound'] >= tda_division[idx - 1]) & (tda_res['tda_c
    else:
        mask = (tda_res['tda_compound'] >= tda_division[idx - 1]) & (tda_res['tda_c

    if sum(mask) <= 0:
        continue

    rgb_intensities = tda_res[mask]['skin color'].apply(eval).apply(rgb_intensity)
    all_rgb_intensities.append(list(rgb_intensities.values))
```

```
In [11]: F, p = f_oneway(*all_rgb_intensities)
         print(F)
         print(p)
```

```
3.459577938826144
7.309227757179605e-07
```

Occupation/Annual Median Salary Evaluation

```
In [12]: occ_res_all = pd.read_csv(respath +'Occupation_Results.csv').sort_values('a_median'
         print(f'Total rows: {len(occ_res_all)}')
         print('Counts of sampled wage categories for median annual wage for all possible ge

         wage_order = ['very low', 'low', 'medium','high','very high'] # Presetting order of
         gender_order = ['man', 'woman', 'unknown','no face']
         pd.crosstab(occ_res_all['gender_detected_val'], occ_res_all['wage_val']).reindex(ge
```

```
Total rows: 1375
Counts of sampled wage categories for median annual wage for all possible gender d
etected values
```

Out[12]:

wage_val gender_detected_val	very low	low	medium	high	very high
man	93	51	100	166	213
woman	90	123	99	26	46
unknown	5	7	5	5	4
no face	104	83	67	56	32

```
In [13]: #For the case of this evaluation we will not be including images where a face could
         #or where the gender could not be determined

         occ_res = occ_res_all[~occ_res_all['gender_detected_val'].isin(['unknown','no face'
         print(f"Total rows after removing faceless and unknown gender detected results: {le
```

```
Total rows after removing faceless and unknown gender detected results: 1007
```

Occupation/Annual Median Salary by Detected Gender

Two Sample Kolmogorov-Smirnov Test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html

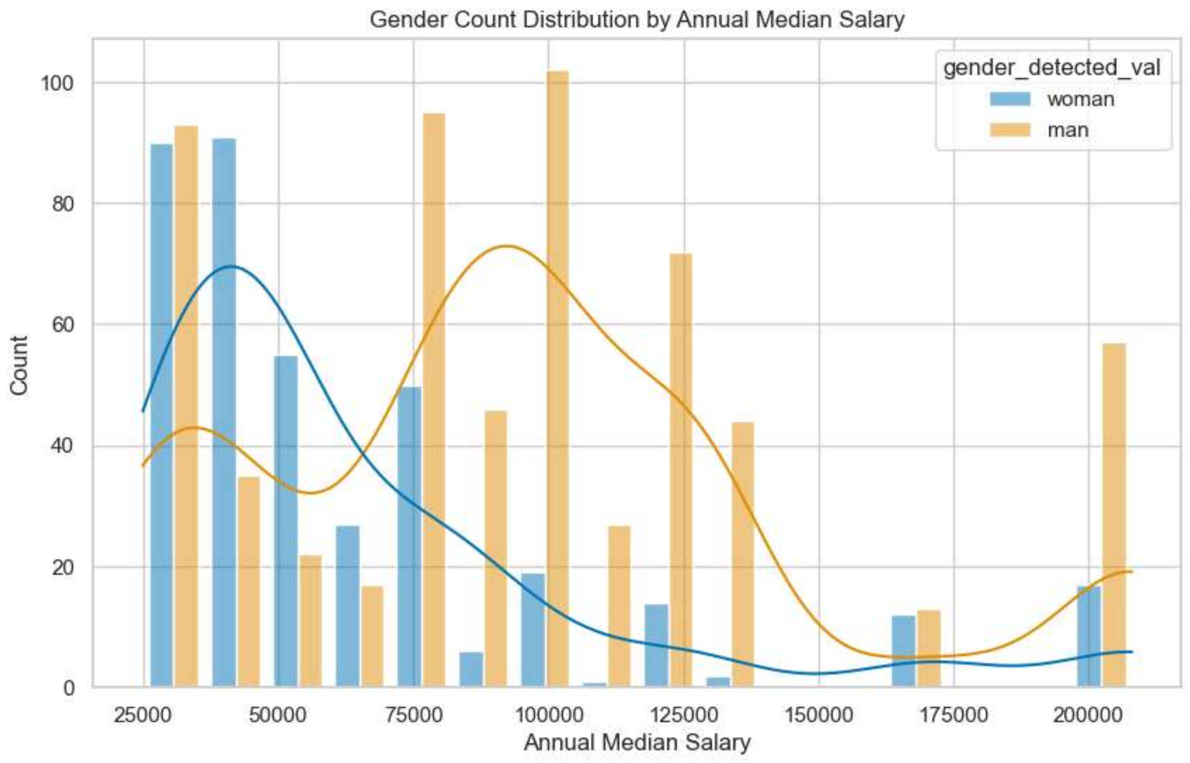
Using the default two-sided parameter for alternative, the null hypothesis is that the two distributions are identical and the alternative is that they are not identical.

If the p-value is lower than our confidence level of 95%, we can reject the null hypothesis in favor of the alternative and conclude that the data were not drawn from the same distribution.

```
In [14]: o = [x for x in ks2sample_test(occ_res, group_col='gender_detected_val', value_col=
         o
```

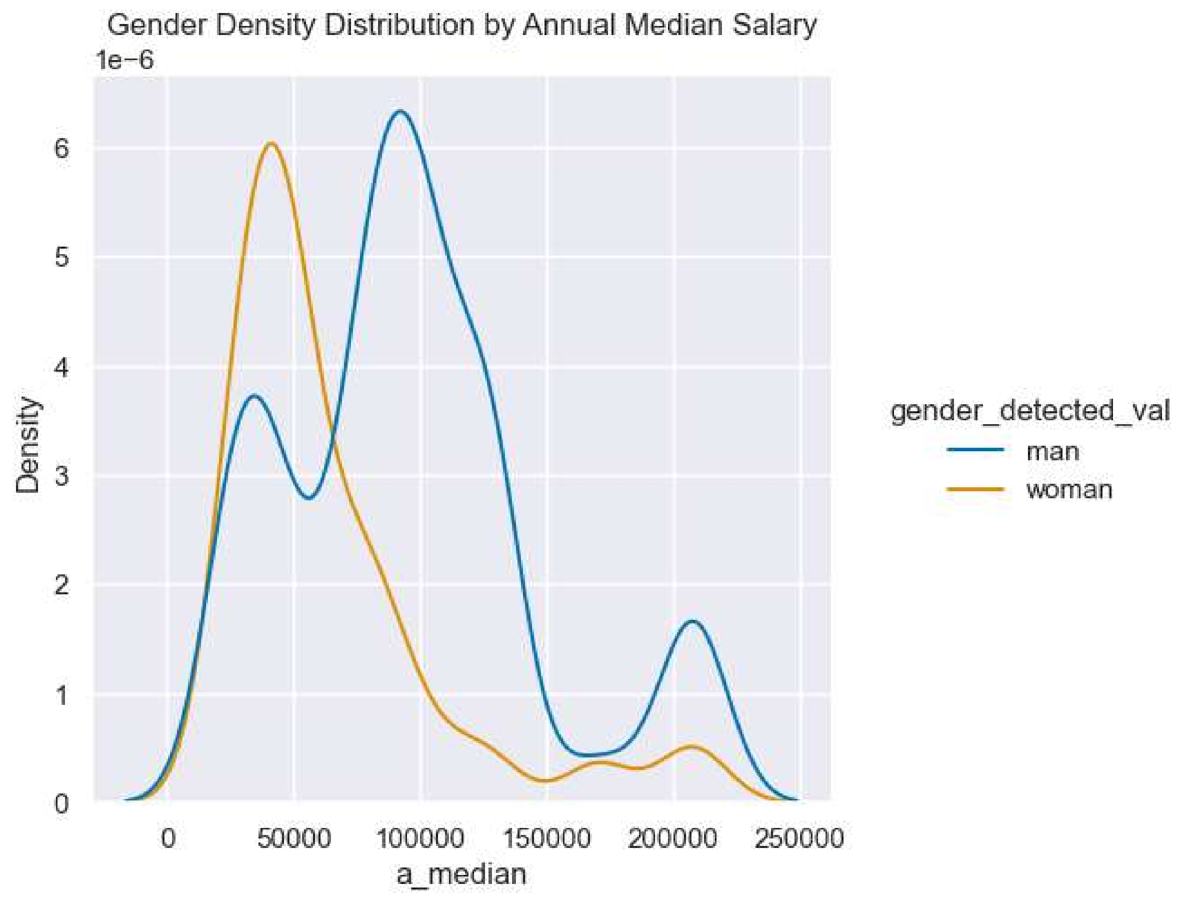
```
Out[14]: [{'group1': 'man',
           'group2': 'woman',
           'statistic': 0.4756345304975923,
           'pvalue': 3.220655285416218e-49}]
```

```
In [15]: occ_hist= generate_histplot(df=occ_res,
                                     x_col='a_median',
                                     hue_col='gender_detected_val',
                                     hue_order=['woman','man'],
                                     title='Gender Count Distribution by Annual Median Salary',
                                     xlabel= 'Annual Median Salary',
                                     ylabel= 'Count')
```



```
In [16]: occ_dis = generate_displot(df=occ_res,
                                    x_col='a_median',
                                    hue_col='gender_detected_val',
                                    kind='kde',
                                    title='Gender Density Distribution by Annual Median Salary')
```

```
<Figure size 1000x600 with 0 Axes>
```



Wilcoxon signed-rank test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

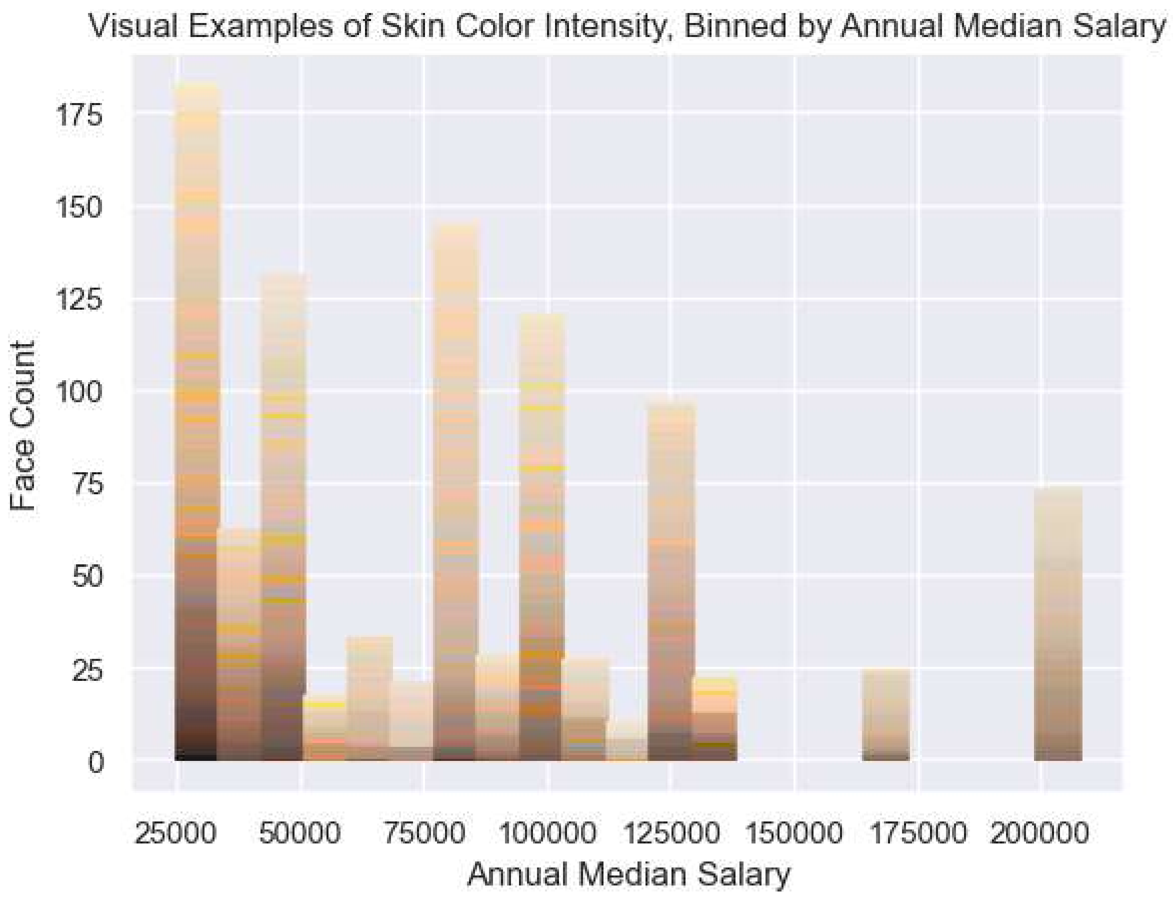
Tests the null hypothesis that two related paired samples come from the same distribution.

In [17]:
```
mask_male = occ_res['gender_detected_cat'] == 4
mask_female = occ_res['gender_detected_cat'] == 3
male_salary = occ_res[mask_male]['a_median'].median()
female_salary = occ_res[mask_female]['a_median'].median()

print(f"Annual median salary for male faces: {male_salary:0.2f}")
print(f"Annual median salary for female faces: {female_salary:0.2f}")

wcox_results = ranksums(occ_res[mask_male]['a_median'], occ_res[mask_female]['a_med

print(wcox_results.statistic)
print(wcox_results.pvalue)
```

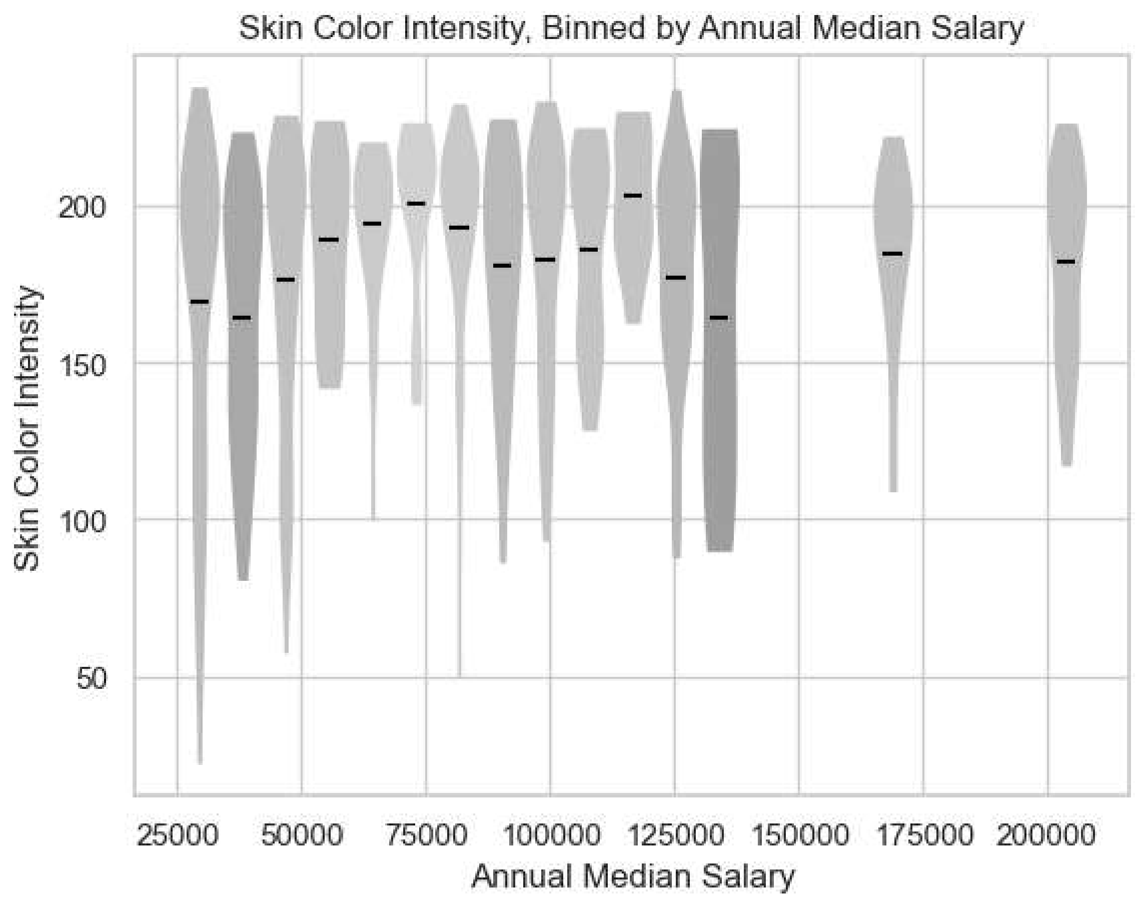
```
Annual median salary for male faces: 95300.00
Annual median salary for female faces: 48260.00
10.92420129220724
8.831393584734473e-28
```

Occupation/Annual Median Salary by Skin Color and Intensity(Lumia)

```
In [18]: # Visual test of intensity sorting per salary bin
         occ_rgb = rgb_histogram(df= occ_res,
                       x_col= 'a_median',
                       rgb_col='skin color',
                       n_bins=21,
                       x_label='Annual Median Salary',
                       y_label='Face Count',
                       title='Visual Examples of Skin Color Intensity, Binned by Annual Medi
```



```
In [19]: # Violin plots of skin intensity per yearly median salary bin
         occ_vp = lumia_violinplot(df = occ_res,
                         x_col ='a_median',
                         rgb_col='skin color',
                         n_bins= 21,
                         widths_val=7500.0,
                         points_val =100,
                         x_label ='Annual Median Salary',
                         y_label = 'Skin Color Intensity',
                         title='Skin Color Intensity, Binned by Annual Median Salary')
```



One-way ANOVA Test

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

The one-way ANOVA tests the null hypothesis that two or more groups have the same population mean.

In [20]:
```
# Getting all rgb intensities
n_bins=21

occ_count, occ_division = np.histogram(occ_res['a_median'], bins=n_bins)

all_rgb_intensities = []

for idx in range(1, len(occ_division)):
    if idx + 1 == len(occ_division):
        mask = (occ_res['a_median'] >= occ_division[idx - 1]) & (occ_res['a_median'
    else:
        mask = (occ_res['a_median'] >= occ_division[idx - 1]) & (occ_res['a_median'

    if sum(mask) <= 0:
        continue

    rgb_intensities = occ_res[mask]['skin color'].apply(eval).apply(rgb_intensity)
    all_rgb_intensities.append(list(rgb_intensities.values))
```

In [21]:
```
F, p = f_oneway(*all_rgb_intensities)
print(F)
print(p)
```

```
4.739047830234663
1.7969689852908102e-08
```