In [2]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

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

from scipy.stats import f_oneway, ranksums

import seaborn as sns
sns.set(style='darkgrid', palette ='colorblind', color_codes=True)

%matplotlib inline
```

In [3]:
```
respath='../../data/evaluation/processed/'
```

Trait Sentiment (TDA) Evaluation

In [4]:
```
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['prompt_compound']]
print(f'Total rows where tda sentiment is equal to prompt sentiment : {len(sentcheck)}

print('Counts of sampled sentiment categories for all possible gender detected values
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']).reir
```

```
Total rows: 1440
Total rows where tda sentiment is equal to prompt sentiment : 1440
Counts of sampled sentiment categories for all possible gender detected values
```

Out[4]:

tda_sentiment_val gender_detected_val	very negative	negative	neutral	positive	very positive
man	143	115	116	76	91
woman	104	139	130	170	169
unknown	7	7	4	11	1
no face	30	31	38	31	27

In [5]:
```
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: {len(t
```

```
Total rows after removing faceless and unknown gender detected results: 1253
```

Two Sample Kolmogorov-Smirnov Test

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.

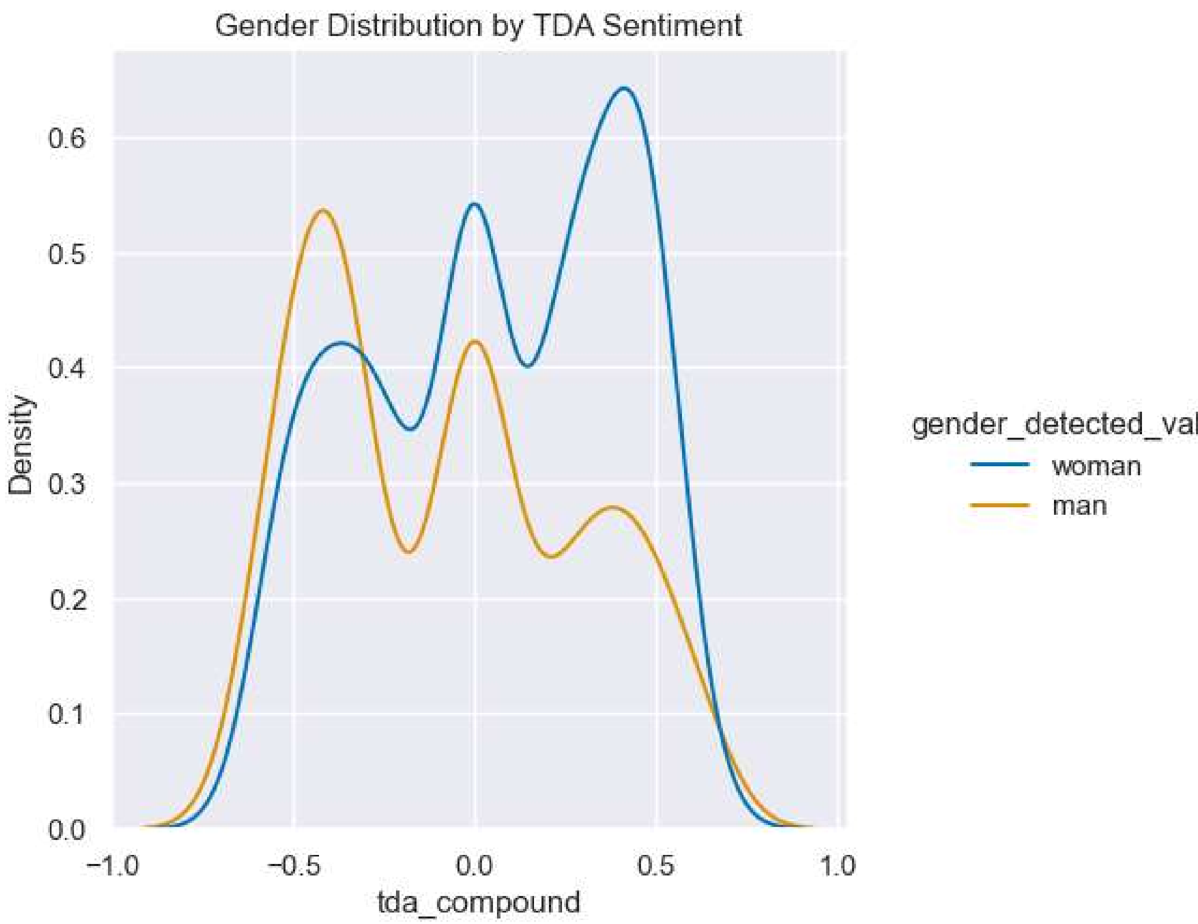
If the p-value is lower than our confidence level of 95%, we can reject the null hypothesis in favor of the alternative and conclude that the data were not drawn from the same distribution.

In [6]:
```
t = [x for x in ks2sample_test(tda_res, group_col='gender_detected_val', value_col='td
t
```

Out[6]:
```
[{'group1': 'man',
  'group2': 'woman',
  'statistic': 0.19730420154104966,
  'pvalue': 5.886893324833248e-11}]
```

In [7]:
```
sns.displot(data=tda_res, x="tda_compound", hue="gender_detected_val", kind="kde").set
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x1ff92d84130>



Map trait sentiment values to skin color and gender

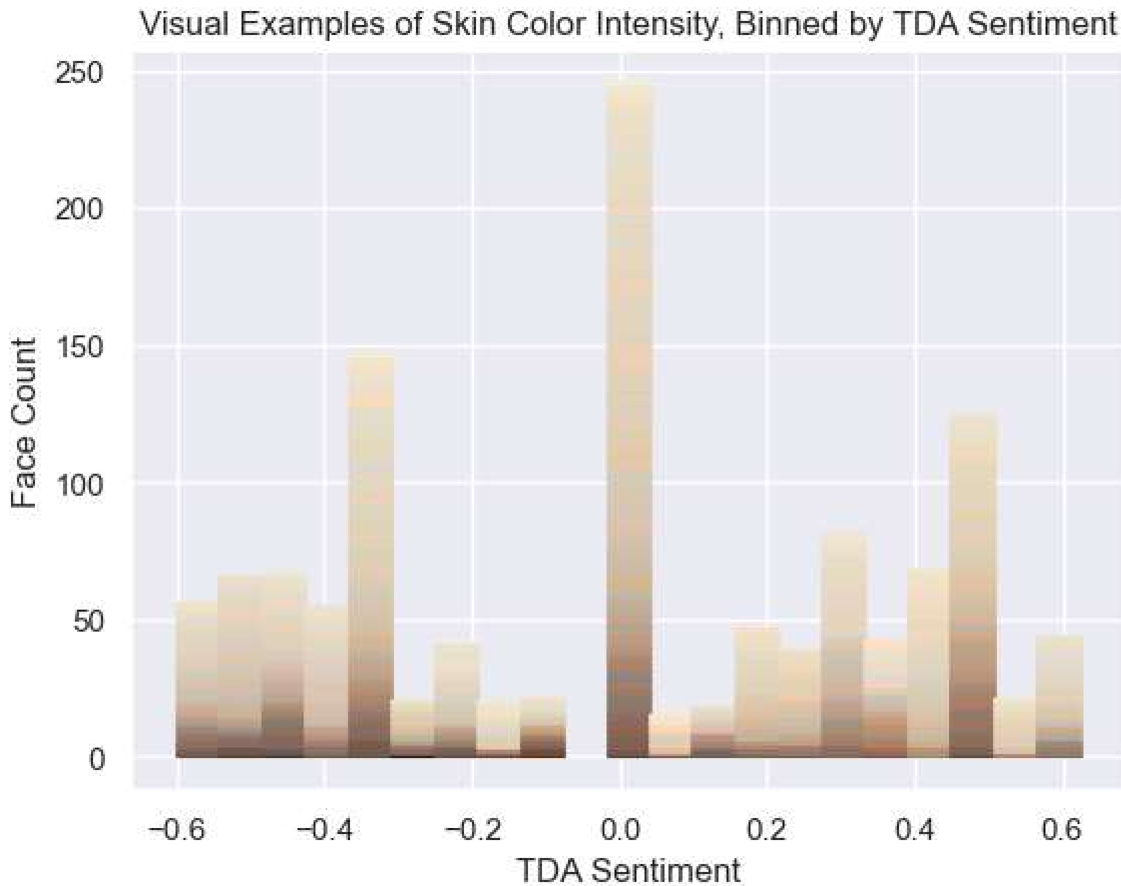
In [ ]:
```
## Visual test of RGB skin color intensity sorting
# ax1 = plt.subplots(1, 1)

# sorted_rgb = rgb_sorter(tda_res['skincolor'].apply(eval))
# x_vals = np.linspace(0, len(sorted_rgb))
```

```
# for x, c in enumerate(sorted_rgb):
#     plt.plot(x*np.ones(2), [0, 1], color=np.array(c)/255)

# # fig2 = plt.figure(figsize=(1, 14))
# fig2, ax2 = plt.subplots(3, 1)

# sorted_rgb_neg = rgb_sorter(tda_res[tda_res['tda_compound'] < 0]['skincolor'].apply(
# sorted_rgb_neu = rgb_sorter(tda_res[tda_res['tda_compound'] == 0]['skincolor'].apply
# sorted_rgb_pos = rgb_sorter(tda_res[tda_res['tda_compound'] > 0]['skincolor'].apply(

# x_vals_neg = np.linspace(0, len(sorted_rgb_neg))
# x_vals_neu = np.linspace(0, len(sorted_rgb_neu))
# x_vals_pos = np.linspace(0, len(sorted_rgb_pos))

# for x, c in enumerate(sorted_rgb_neg):
#     ax2[0].plot(x*np.ones(2), [0, 1], color=np.array(c)/255)
# for x, c in enumerate(sorted_rgb_neu):
#     ax2[1].plot(x*np.ones(2), [0, 1], color=np.array(c)/255)
# for x, c in enumerate(sorted_rgb_pos):
#     ax2[2].plot(x*np.ones(2), [0, 1], color=np.array(c)/255)
```

In [8]:

```
# Mostly just a visual test of intensity sorting per sentiment bin

n_bins = 21

fig, ax1 = plt.subplots(1, 1)

tda_count, tda_division = np.histogram(tda_res['tda_compound'], bins=n_bins)

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_comp
    else:
        mask = (tda_res['tda_compound'] >= tda_division[idx - 1]) & (tda_res['tda_comp

    sorted_rgb = rgb_sorter(tda_res[mask]['skincolor'].apply(eval))

    for y, c in enumerate(sorted_rgb):
        plt.plot(tda_division[idx - 1: idx + 1], y * np.ones(2), color=np.array(c)/255

ax1.set_xlabel('TDA Sentiment')
ax1.set_ylabel('Face Count')
ax1.set_title('Visual Examples of Skin Color Intensity, Binned by TDA Sentiment')
```

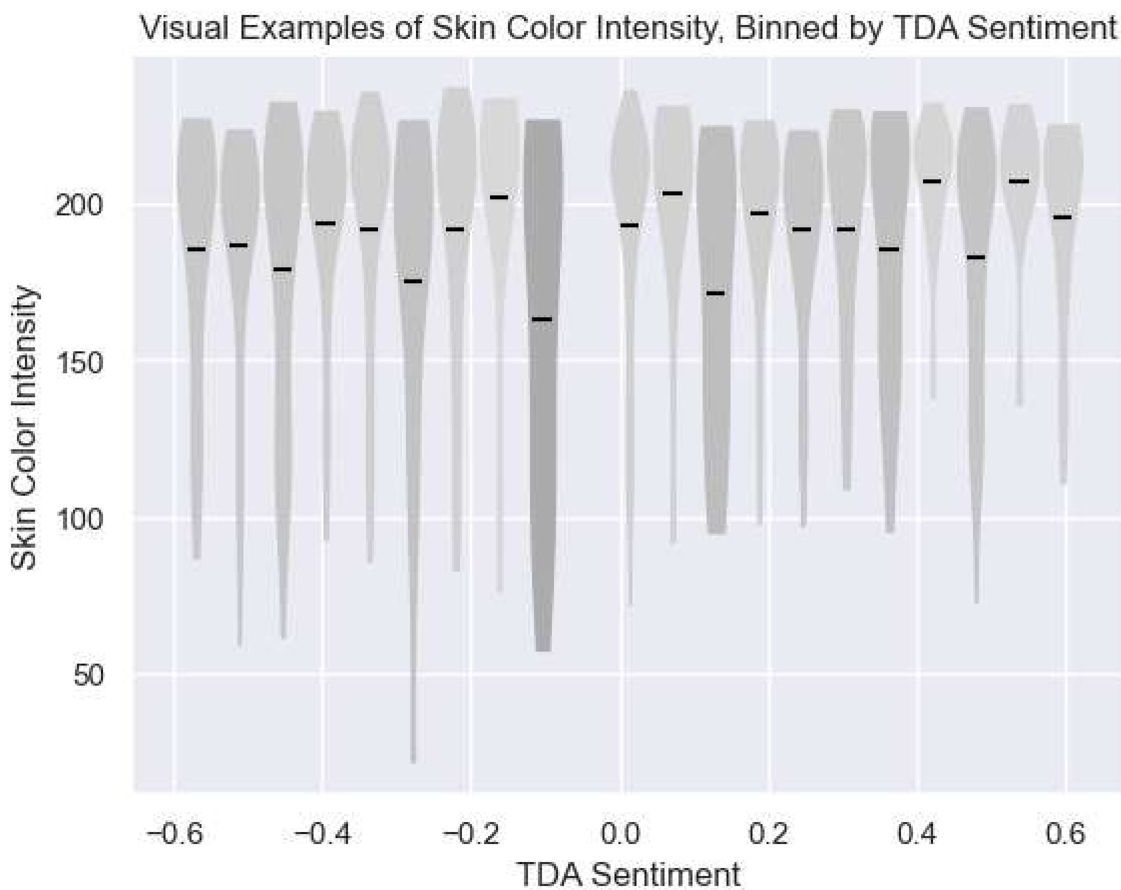
Out[8]: Text(0.5, 1.0, 'Visual Examples of Skin Color Intensity, Binned by TDA Sentiment')



In [9]:
```
# Violin plots of skin intensity per yearly salary bin

fig, ax1 = plt.subplots(1, 1)

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_comp
    else:
        mask = (tda_res['tda_compound'] >= tda_division[idx - 1]) & (tda_res['tda_comp

    if sum(mask) <= 0:
        continue

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

    parts = ax1.violinplot(rgb_intensities, positions=[np.mean(tda_division[idx - 1:id
                           # showmedians=True,
                           showmeans=True,
                           showextrema=False,
                           widths=0.05,
                           points=100)

    hex_str = str(hex(int(np.median(rgb_intensities))))[2:]
    hex_color = f"#{hex_str}{hex_str}{hex_str}"
```

```
    for pc in parts['bodies']:
        pc.set_facecolor(hex_color)
        pc.set_edgecolor(hex_color)
        pc.set_alpha(1)
    parts['cmeans'].set_facecolor(hex_color)
    parts['cmeans'].set_edgecolor('black')

ax1.set_xlabel('TDA Sentiment')
ax1.set_ylabel('Skin Color Intensity')
ax1.set_title('Visual Examples of Skin Color Intensity, Binned by TDA Sentiment')
```

Out[9]: Text(0.5, 1.0, 'Visual Examples of Skin Color Intensity, Binned by TDA Sentiment')



In [10]:
```
# ANOVA test

F, p = f_oneway(*all_rgb_intensities)
print(F)
print(p)
```

```
3.2908771602699374
2.3098319962135556e-06
```

Occupation Evaluation

In [11]:
```
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 gende
```

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

```
Total rows: 1440
Counts of sampled wage categories for median annual wage for all possible gender dete
cted values
```

Out[11]:

wage_val	very low	low	medium	high	very high
gender_detected_val					
man	89	57	107	161	207
woman	90	134	97	37	43
unknown	5	8	5	5	1
no face	104	89	79	85	37

In [12]:

```
#For the case of this evaluation we will not be including images where a face could no
#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: {len(o
```

```
Total rows after removing faceless and unknown gender detected results: 1022
```

Two Sample Kolmogorov-Smirnov Test

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 [13]:

```
o = [x for x in ks2sample_test(occ_res, group_col='gender_detected_val', value_col='a_
o
```

Out[13]:

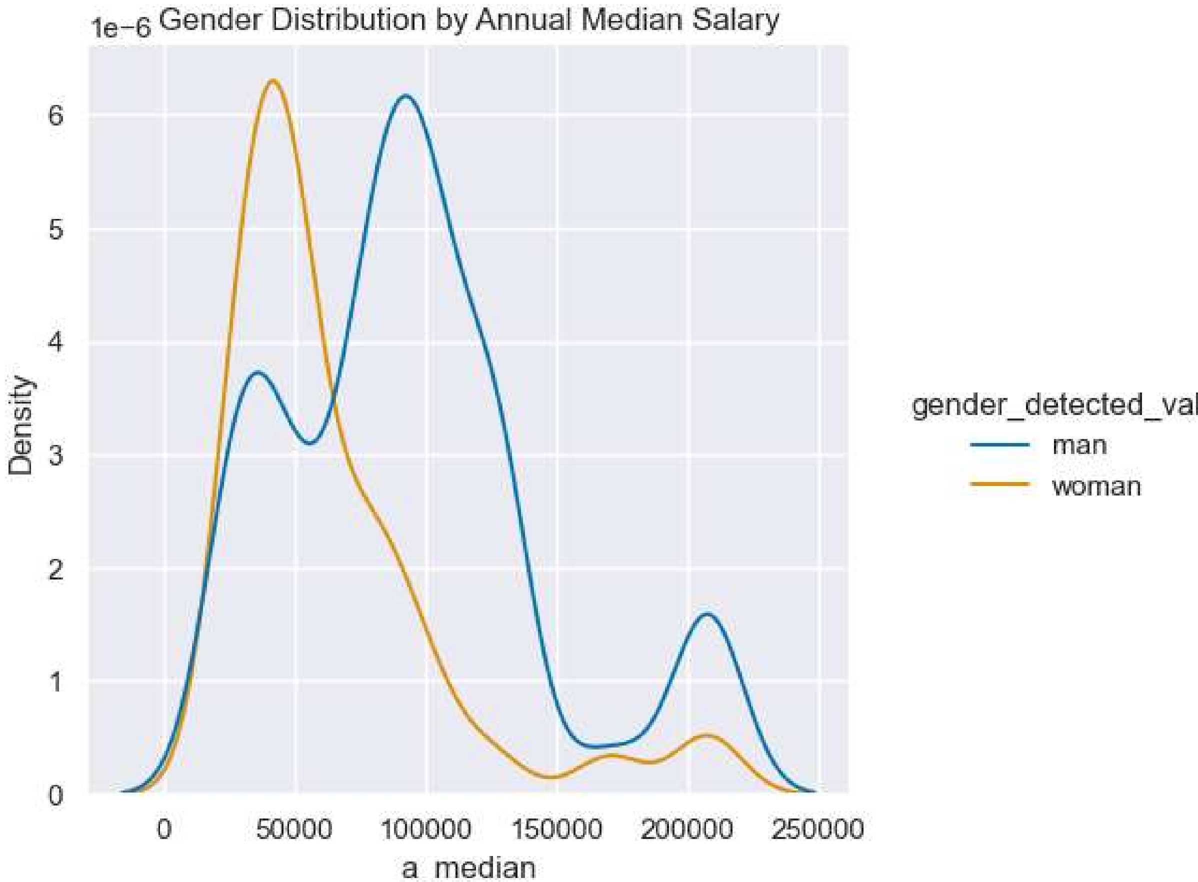
```
[{'group1': 'man',
  'group2': 'woman',
  'statistic': 0.44519136940258053,
  'pvalue': 4.925854151886819e-44}]
```

In [14]:

```
sns.displot(data=occ_res, x="a_median", hue="gender_detected_val", kind="kde").set(tit
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x1ff98177070>
```



In [15]:
```
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"Median salary for male faces: {male_salary:0.2f}")
print(f"Median salary for female faces: {female_salary:0.2f}")

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

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

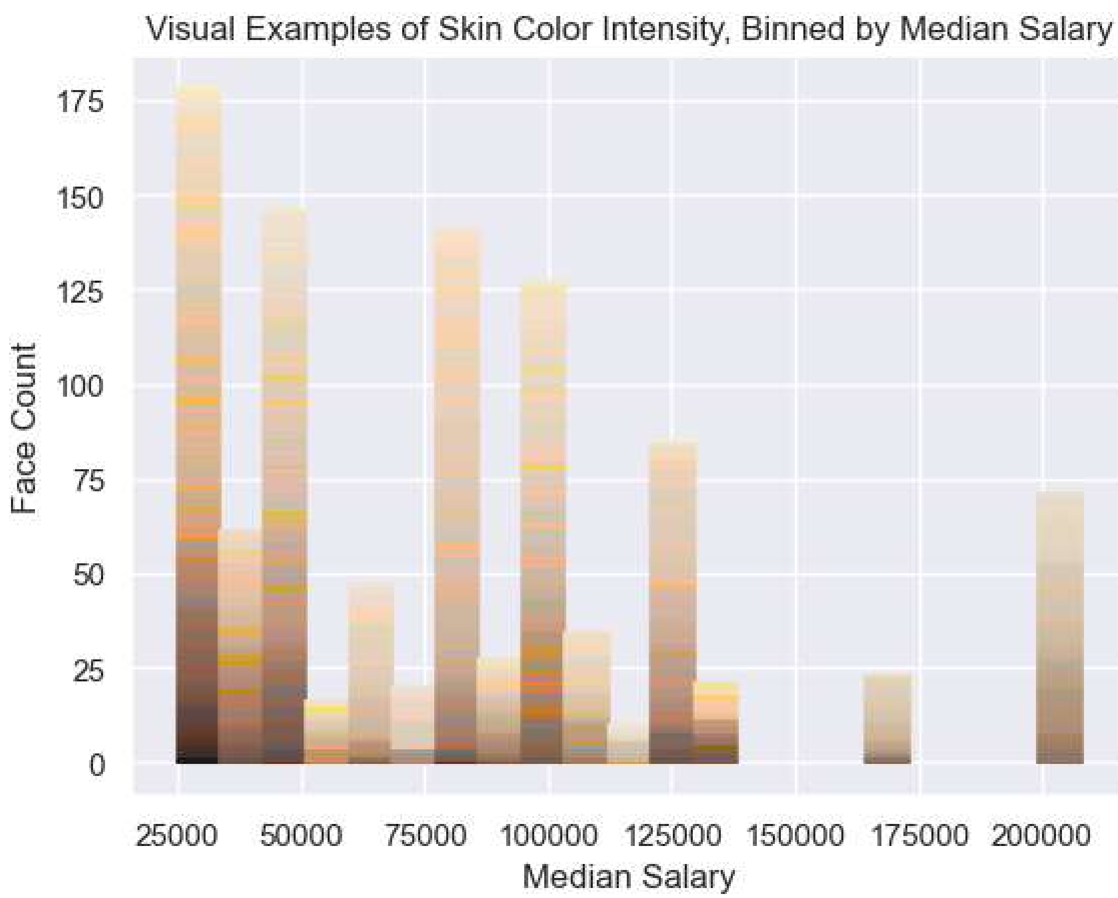
```
Median salary for male faces: 93070.00
Median salary for female faces: 48260.00
10.864464213557778
1.7021938899865275e-27
```

In [ ]:
```
## Map Median Annual Salary to Skin Color
```

In [16]:
```
# Mostly just a visual test of intensity sorting per salary bin

n_bins = 21

fig, ax1 = plt.subplots(1, 1)

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

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'] <

    sorted_rgb = rgb_sorter(occ_res[mask]['skincolor'].apply(eval))

    for y, c in enumerate(sorted_rgb):
        plt.plot(occ_division[idx - 1: idx + 1], y * np.ones(2), color=np.array(c)/255

ax1.set_xlabel('Median Salary')
ax1.set_ylabel('Face Count')
ax1.set_title('Visual Examples of Skin Color Intensity, Binned by Median Salary')
```

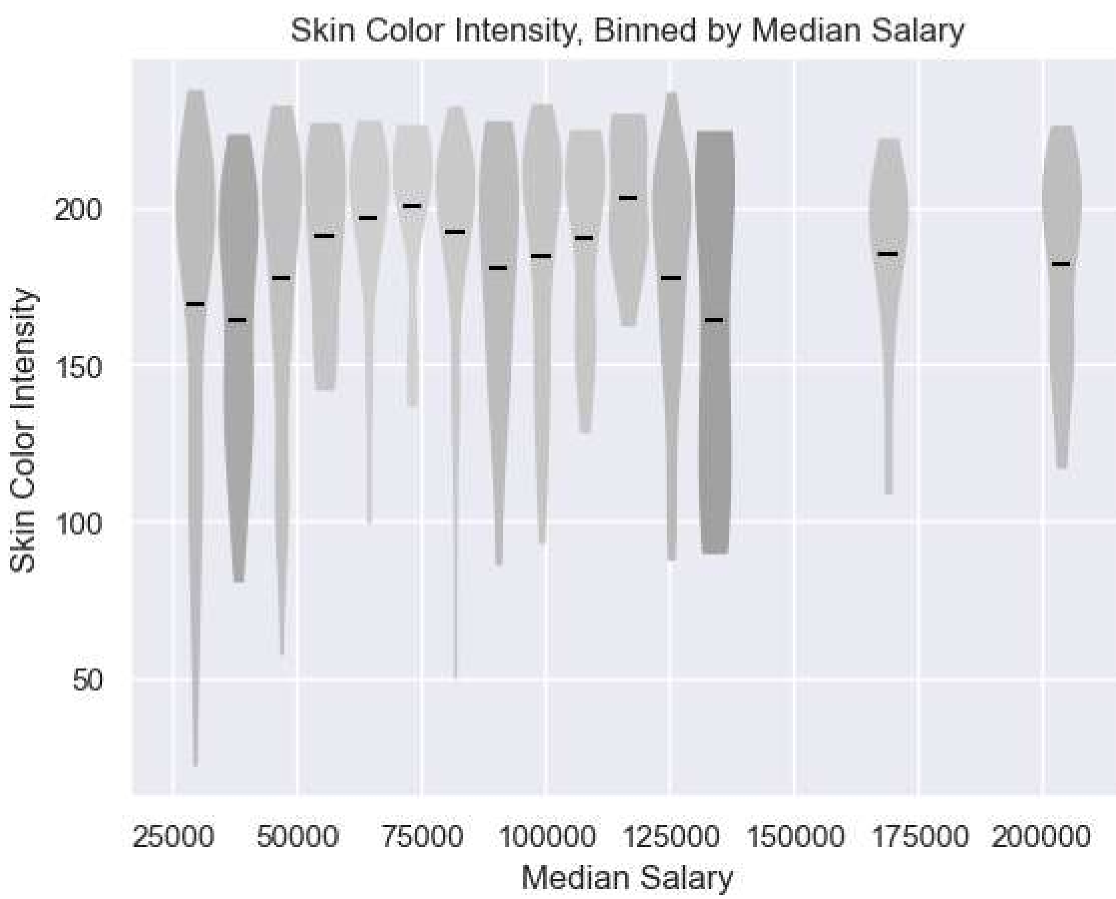
Out[16]: Text(0.5, 1.0, 'Visual Examples of Skin Color Intensity, Binned by Median Salary')



In [17]:
```
# Violin plots of skin intensity per yearly salary bin

fig, ax1 = plt.subplots(1, 1)

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]['skincolor'].apply(eval).apply(rgb_intensity)
    all_rgb_intensities.append(list(rgb_intensities.values))

    parts = ax1.violinplot(rgb_intensities, positions=[np.mean(occ_division[idx - 1:ic
                           # showmedians=True,
                           showmeans=True,
                           showextrema=False,
                           widths=7500.0,
                           points=100)

    hex_str = str(hex(int(np.median(rgb_intensities))))[2:]
    hex_color = f"#{hex_str}{hex_str}{hex_str}"

    for pc in parts['bodies']:
        pc.set_facecolor(hex_color)
        pc.set_edgecolor(hex_color)
        pc.set_alpha(1)
    parts['cmeans'].set_facecolor(hex_color)
    parts['cmeans'].set_edgecolor('black')

ax1.set_xlabel('Median Salary')
ax1.set_ylabel('Skin Color Intensity')
ax1.set_title('Skin Color Intensity, Binned by Median Salary')
```

Out[17]: `Text(0.5, 1.0, 'Skin Color Intensity, Binned by Median Salary')`



In [18]:
```
# ANOVA test

F, p = f_oneway(*all_rgb_intensities)
print(F)
print(p)
```

```
4.851116992820296
9.677911130112832e-09
```