Python_Code

November 10, 2022

0.1 Data Exploration for CodeDay Data Set

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statistics

import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
import statsmodels.api as sm

%matplotlib inline
```

[2]:
```
data = pd.read_csv("combinedData_individual.csv")
```

NOTE: gender_pct and ethnicity_pct were calculated using the ENTIRE dataset. This means that the attendees who did not provide their demographics data were included.

[ ]:
```
data.head()
```

[3]:
```
# Dates of events
dates = pd.to_datetime(data["startsAt"])
date1 = min(dates)
date2 = max(dates)
print(date1) # Nov 8, 2014
print(date2) # Nov 12, 2016
```

```
2014-11-08 00:00:00
2016-11-12 00:00:00
```

0.1.1 Individual Attendee Demographics

The attendees voluntarily provided demographics data, along with their registrations.

[4]:
```
# Total number of attendees:
print("Total number of attendees:",len(data))
```

```
# Total number of attendees where gender is known:
known_gender = data[data["gender"] != "unknown"]

# Total number of attendees where gender and ethnicity is known:
known_gender_and_ethnicity = known_gender[known_gender["ethnicity"] !=␣
 ↪"unknown"]
print("Attendees for whom demographics are known:",␣
 ↪len(known_gender_and_ethnicity))

# Total number of attendees where gender and ethnicity are NOT known:
not_known = data[data["gender"] == "unknown"]
print("Attendees for whom demographics are NOT known:", len(not_known))

# Percent of attendees providing demographics data
print("Percent of attendees providing demographics data",␣
 ↪len(known_gender_and_ethnicity)/len(data))
```

```
Total number of attendees: 14057
Attendees for whom demographics are known: 7207
Attendees for whom demographics are NOT known: 6850
Percent of attendees providing demographics data 0.5126982997794693
```

Removing participants who didn't report their gender leaves us with a dataset of 7207 individuals. When this subset of participants is further filtered for unknown ethnicity, no further individuals are removed, meaning the same individuals who did not report their gender ALSO did not report their ethnicity.

[5]:
```
# Saving a file with data ONLY for participants
# who provided complete gender and ethnicity data.

known_gender_and_ethnicity.to_csv('no_unknowns.csv')
```

[6]:
```
# for ease of typing will use shorter name
no_unknowns = known_gender_and_ethnicity
```

Follow-up surveys were sent to all attendees who indicated their interest in coding prior to attending the event was "low" or "none".

[7]:
```
# Number of attendees with low prior interest in coding:
low_interest = data[data["codingInterest"] == "low"]

# Number of attendees with no prior interest in coding:
no_interest = data[data["codingInterest"] == "none"]

# Total number of attendees sent follow-up (i.e. post-event surveys):
print("Total number of attendees sent follow-up surveys:",␣
 ↪len(low_interest)+len(no_interest))
```

```
# Total number of follow-up surveys returned:
low_interest_fu = low_interest[low_interest["outcome"] != "unknown"]
no_interest_fu = no_interest[no_interest["outcome"] != "unknown"]
print("Total number of follow-up surveys returned:",␣
 ↪len(low_interest_fu)+len(no_interest_fu))
print("Percent of follow-up surveys returned:",␣
 ↪(len(low_interest_fu)+len(no_interest_fu))/
 ↪(len(low_interest)+len(no_interest)))
```

```
Total number of attendees sent follow-up surveys: 4703
Total number of follow-up surveys returned: 3190
Percent of follow-up surveys returned: 0.6782904529024028
```

[8]:
```
# Checking to see how many attendees with prior 'low' interest in coding
# did not report gender and ethnicity.
low_interest_unk = low_interest[low_interest['gender'] == 'unknown']
print("Low interest unknowns:", len(low_interest_unk))

# Checking to see how many attendees with prior 'no' interest in coding
# did not report gender and ethnicity.
no_interest_unk = no_interest[no_interest['gender'] == 'unknown']
print("No interest unknowns:", len(no_interest_unk))
```

```
Low interest unknowns: 0
No interest unknowns: 0
```

[ ]:
```
not_known.head()
```

[9]:
```
# The attendees who did not provide information on their gender and ethnicity
# also did not provide information on their prior level of interest in coding.
# Thus, none of these attendees were sent follow-up surveys,
# so outcome in unknown by default.
print(len(not_known[not_known['gender'] != 'unknown']))
print(len(not_known[not_known['ethnicity'] != 'unknown']))
print(len(not_known[not_known['codingInterest'] != 'unknown']))
print(len(not_known[not_known['outcome'] != 'unknown']))
```

```
0
0
0
0
```

How many students reporting ‘low’ or ‘no’ prior interest in coding presented their work at the event?

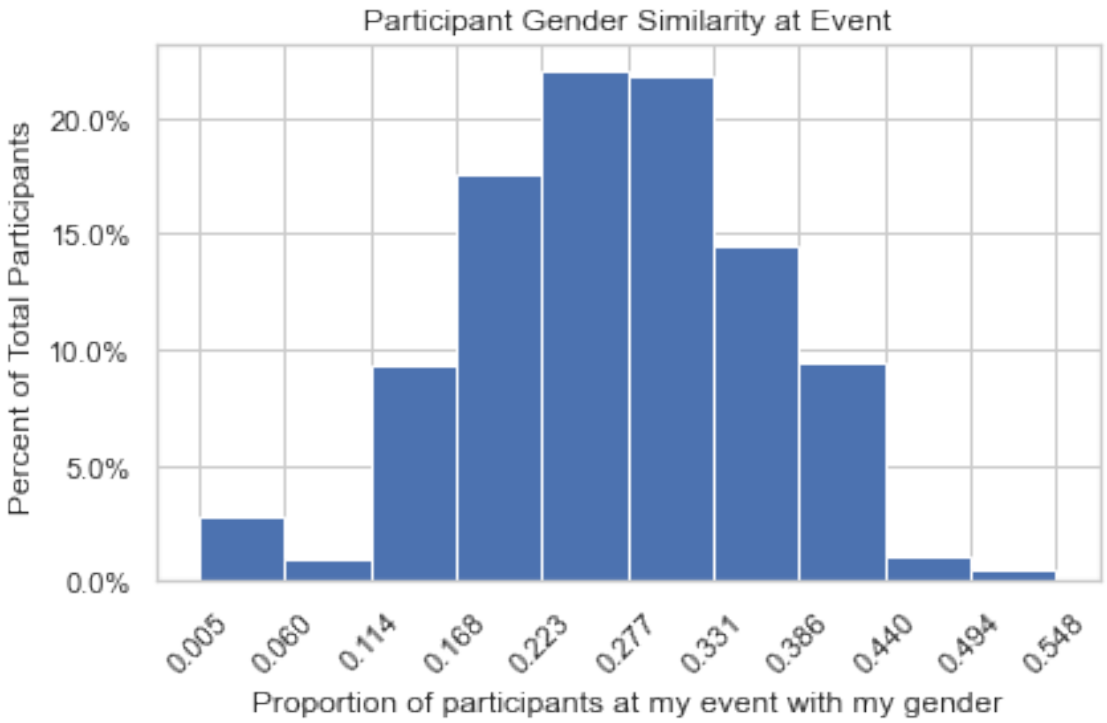
[10]:
```
# Number of attendees who responded to f/u survey who have data on whether they␣
 ↪presented:
```

```
low_interest_presented = low_interest_fu[low_interest_fu["presented"] !=␣
 ↪"unknown"]
no_interest_presented = no_interest_fu[no_interest_fu["presented"] != "unknown"]

# Total number:
print("Total number of attendees sent follow-up surveys \n who have info on␣
 ↪whether they presented:",␣
 ↪len(low_interest_presented)+len(no_interest_presented))
```

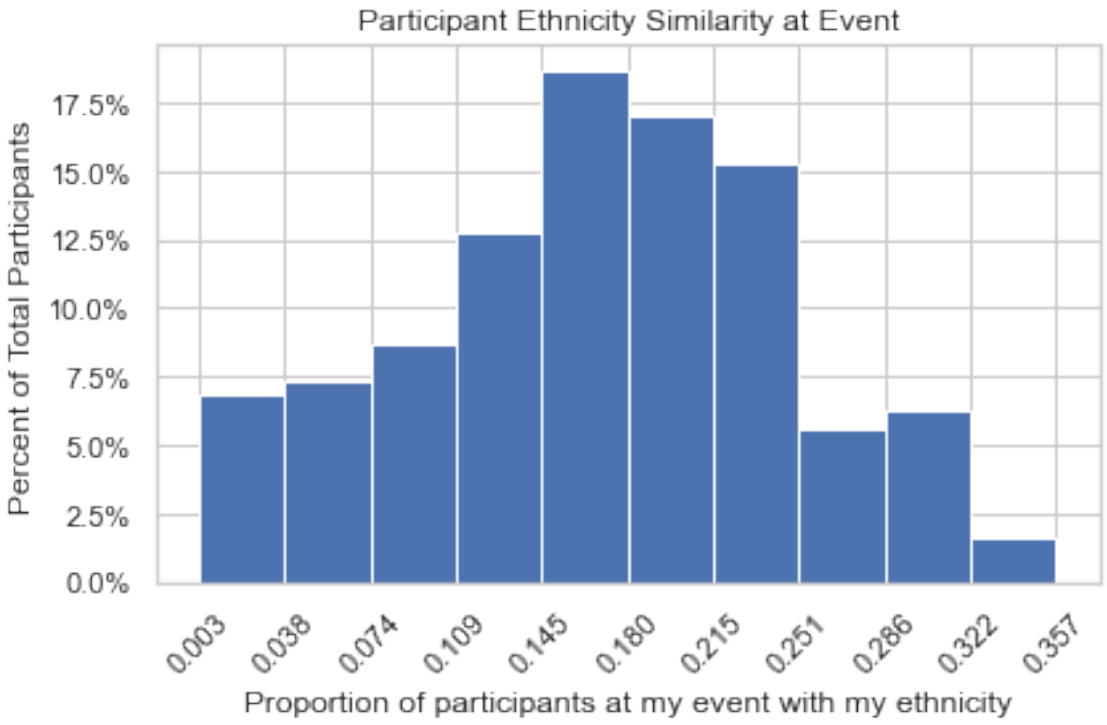
```
Total number of attendees sent follow-up surveys
 who have info on whether they presented: 3107
```

Overall, looking at attendees for whom we have data (i.e. excluding unknowns):

[11]:
```
from matplotlib.ticker import PercentFormatter
from matplotlib.ticker import StrMethodFormatter

plotdata = no_unknowns['gender_pct']

n, bins, edges = plt.hist(plotdata, weights=np.ones(len(plotdata)) /␣
 ↪len(plotdata))

plt.xlabel("Proportion of participants at my event with my gender")
plt.ylabel("Percent of Total Participants")

plt.xticks(bins, rotation = 45)

plt.grid(True)

plt.gca().xaxis.set_major_formatter(StrMethodFormatter('{x:,.3f}')) # 3 decimal␣
 ↪places

plt.gca().yaxis.set_major_formatter(PercentFormatter(1))

plt.title("Participant Gender Similarity at Event")
plt.tight_layout()
plt.savefig('gender_sim_hist.png', dpi=300)
plt.show()
```



[12]:
```
plotdata = no_unknowns['ethnicity_pct']

n, bins, edges = plt.hist(plotdata, weights=np.ones(len(plotdata)) /␣
 ↪len(plotdata))

plt.xlabel("Proportion of participants at my event with my ethnicity")
plt.ylabel("Percent of Total Participants")

plt.xticks(bins, rotation = 45)

plt.grid(True)

plt.gca().xaxis.set_major_formatter(StrMethodFormatter('{x:,.3f}')) # 3 decimal␣
 ↪places

plt.gca().yaxis.set_major_formatter(PercentFormatter(1))

plt.title("Participant Ethnicity Similarity at Event")
plt.tight_layout()
plt.savefig('ethnicity_sim_hist.png', dpi=300)

plt.show()
```



0.2 Event Demographics

We are interested in how attendees of different gender and ethnicity persist with coding activities after attending a CodeDay event. Since part of feeling as though one "belongs" to a community is affected by whether or not a person sees others like them in that community, we calculated "Gender Proportion" and "Ethnicity Proportion" to look at how the overall event demographics matched the demographics of the attendee from the point of view of each attendee.

Since we don't know the ethnicity and gender of 6,850 attendees, the percentages for all of these categories represents minimum values, rather than the precise values.

[13]:
```
# Construct heatmaps
eth = list(set(data["ethnicity"]))
print(eth)

gen = list(set(data["gender"]))
print(gen)
```

```
['latinx', 'nativeAmerican', 'other', 'black', 'white', 'unknown', 'asian']
['nonBinary', 'male', 'female', 'unknown']
```

[14]:
```
gen_eth_combinations_LI = []
for each_e in eth:
```

```
    for each_g in gen:
        tup = (each_e, each_g)
        gen_eth_combinations_LI.append(tup)

print(gen_eth_combinations_LI)
gen_eth_combi_counts_LI = {}

for index in range(len(low_interest)):
    curr_ind = low_interest.iloc[index]
    curr_ind_eth = curr_ind["ethnicity"]
    curr_ind_gen = curr_ind["gender"]
    curr_ind_tup = (curr_ind_eth, curr_ind_gen)
    if(curr_ind_tup not in gen_eth_combi_counts_LI.keys()):
        gen_eth_combi_counts_LI[curr_ind_tup] = 1
    else:
        gen_eth_combi_counts_LI[curr_ind_tup] += 1

gen_eth_combi_counts_LI
```

```
[('latinx', 'nonBinary'), ('latinx', 'male'), ('latinx', 'female'), ('latinx',
'unknown'), ('nativeAmerican', 'nonBinary'), ('nativeAmerican', 'male'),
('nativeAmerican', 'female'), ('nativeAmerican', 'unknown'), ('other',
'nonBinary'), ('other', 'male'), ('other', 'female'), ('other', 'unknown'),
('black', 'nonBinary'), ('black', 'male'), ('black', 'female'), ('black',
'unknown'), ('white', 'nonBinary'), ('white', 'male'), ('white', 'female'),
('white', 'unknown'), ('unknown', 'nonBinary'), ('unknown', 'male'), ('unknown',
'female'), ('unknown', 'unknown'), ('asian', 'nonBinary'), ('asian', 'male'),
('asian', 'female'), ('asian', 'unknown')]
```

[14]:
```
{('black', 'nonBinary'): 17,
 ('white', 'male'): 703,
 ('other', 'female'): 59,
 ('white', 'female'): 469,
 ('latinx', 'female'): 233,
 ('nativeAmerican', 'male'): 56,
 ('asian', 'female'): 427,
 ('latinx', 'male'): 346,
 ('asian', 'male'): 541,
 ('black', 'female'): 138,
 ('white', 'nonBinary'): 37,
 ('other', 'male'): 96,
 ('black', 'male'): 241,
 ('other', 'nonBinary'): 6,
 ('asian', 'nonBinary'): 34,
 ('latinx', 'nonBinary'): 10,
 ('nativeAmerican', 'female'): 37,
 ('nativeAmerican', 'nonBinary'): 1}
```

[15]:
```
ethnicities = ['white', 'asian', 'latinx', 'black','nativeAmerican', 'other']
genders = ['male', 'female', 'non-binary']
```

[16]:
```
# Hand transcribed from gen_eth_combi_counts_LI
# [white, asian, black, nativeAmerican, other]
row1_M_LI = [703, 541, 346, 241, 56, 96]
row2_F_LI = [469, 427, 233, 138, 37, 59]
row3_NB_LI = [37, 34, 10, 17, 1, 6]
```

[17]:
```
# Gender breakdowns for low- and no-interest groups

total_M_LI = low_interest[low_interest["gender"] == "male"]
total_F_LI = low_interest[low_interest["gender"] == "female"]
total_NB_LI = low_interest[low_interest["gender"] == "nonBinary"]

cont_M_LI = total_M_LI[total_M_LI["outcome"] == "yes"]
cont_F_LI = total_F_LI[total_F_LI["outcome"] == "yes"]
cont_NB_LI = total_NB_LI[total_NB_LI["outcome"] == "yes"]

print("Number of low-interest participants who are male:", len(total_M_LI), "-␣
 ↪Percent:", len(total_M_LI)/len(low_interest))
print("\tNumber of low-interest males who continued:", len(cont_M_LI), "-␣
 ↪Percent:", len(cont_M_LI)/len(total_M_LI))
print("Number of low-interest participants who are female:", len(total_F_LI),␣
 ↪"- Percent:", len(total_F_LI)/len(low_interest))
print("\tNumber of low-interest females who continued:", len(cont_F_LI), "-␣
 ↪Percent:", len(cont_F_LI)/len(total_F_LI))
print("Number of low-interest participants who are non-binary:",␣
 ↪len(total_NB_LI), "- Percent:", len(total_NB_LI)/len(low_interest))
print("\tNumber of low-interest non-binary ind. who continued:",␣
 ↪len(cont_NB_LI), "- Percent:", len(cont_NB_LI)/len(total_NB_LI))

print()

total_M_NI = no_interest[no_interest["gender"] == "male"]
total_F_NI = no_interest[no_interest["gender"] == "female"]
total_NB_NI = no_interest[no_interest["gender"] == "nonBinary"]

cont_M_NI = total_M_NI[total_M_NI["outcome"] == "yes"]
cont_F_NI = total_F_NI[total_F_NI["outcome"] == "yes"]
cont_NB_NI = total_NB_NI[total_NB_NI["outcome"] == "yes"]

print("Number of no-interest participants who are male:", len(total_M_NI), "-␣
 ↪Percent:", len(total_M_NI)/len(no_interest))
print("\tNumber of low-interest males who continued:", len(cont_M_NI), "-␣
 ↪Percent:", len(cont_M_NI)/len(total_M_NI))
```

```
print("Number of no-interest participants who are female:", len(total_F_NI), "-␣
 ↪Percent:", len(total_F_NI)/len(no_interest))
print("\tNumber of low-interest females who continued:", len(cont_F_NI), "-␣
 ↪Percent:", len(cont_F_NI)/len(total_F_NI))
print("Number of no-interest participants who are non-binary:",␣
 ↪len(total_NB_NI), "- Percent:", len(total_NB_NI)/len(no_interest))
print("\tNumber of low-interest non-binary ind. who continued:",␣
 ↪len(cont_NB_NI), "- Percent:", len(cont_NB_NI)/len(total_NB_NI))
```

```
Number of low-interest participants who are male: 1983 - Percent:
0.5746160533178789
        Number of low-interest males who continued: 954 - Percent:
0.481089258698941
Number of low-interest participants who are female: 1363 - Percent:
0.3949579831932773
        Number of low-interest females who continued: 629 - Percent:
0.4614820249449743
Number of low-interest participants who are non-binary: 105 - Percent:
0.030425963488843813
        Number of low-interest non-binary ind. who continued: 54 - Percent:
0.5142857142857142

Number of no-interest participants who are male: 734 - Percent:
0.5862619808306709
        Number of low-interest males who continued: 256 - Percent:
0.34877384196185285
Number of no-interest participants who are female: 488 - Percent:
0.38977635782747605
        Number of low-interest females who continued: 156 - Percent:
0.319672131147541
Number of no-interest participants who are non-binary: 30 - Percent:
0.023961661341853034
        Number of low-interest non-binary ind. who continued: 7 - Percent:
0.23333333333333334
```

[18]:
```
tot_LI = [row1_M_LI, row2_F_LI, row3_NB_LI]
tot_LI
```

[18]:
```
[[703, 541, 346, 241, 56, 96],
 [469, 427, 233, 138, 37, 59],
 [37, 34, 10, 17, 1, 6]]
```

[19]:
```
df_LI = pd.DataFrame(tot_LI, index=genders, columns=ethnicities)
total_LI = sum(row1_M_LI) + sum(row2_F_LI) + sum(row3_NB_LI)
```

[20]:
```
gen_eth_combinations_NI = []
for each_e in eth:
```

```
    for each_g in gen:
        tup = (each_e, each_g)
        gen_eth_combinations_NI.append(tup)

gen_eth_combi_counts_NI = {}



for index in range(len(no_interest)):
    curr_ind = no_interest.iloc[index]
    curr_ind_eth = curr_ind["ethnicity"]
    curr_ind_gen = curr_ind["gender"]
    curr_ind_tup = (curr_ind_eth, curr_ind_gen)
    if(curr_ind_tup not in gen_eth_combi_counts_NI.keys()):
        gen_eth_combi_counts_NI[curr_ind_tup] = 1
    else:
        gen_eth_combi_counts_NI[curr_ind_tup] += 1

#gen_eth_combi_counts_NI
```

[21]:
```
# Hand transcribed: [white, asian, black, nativeAmerican, other]
row1_M_NI = [254, 221, 132, 81, 19, 27]
row2_F_NI = [191, 132, 67, 67, 12, 19]
row3_NB_NI = [13, 6, 4, 5, 1, 1]
```

[22]:
```
tot_NI = [row1_M_NI, row2_F_NI, row3_NB_NI]
tot_NI
```

[22]:
```
[[254, 221, 132, 81, 19, 27], [191, 132, 67, 67, 12, 19], [13, 6, 4, 5, 1, 1]]
```

[23]:
```
df_NI = pd.DataFrame(tot_NI, index=genders, columns=ethnicities)
total_NI = sum(row1_M_NI) + sum(row2_F_NI) + sum(row3_NB_NI)
```

[24]:
```
# all participants

gen_eth_combinations = []
for each_e in eth:
    for each_g in gen:
        tup = (each_e, each_g)
        gen_eth_combinations.append(tup)

gen_eth_combi_counts = {}



for index in range(len(data)):
    curr_ind = data.iloc[index]
```

```
    curr_ind_eth = curr_ind["ethnicity"]
    curr_ind_gen = curr_ind["gender"]
    curr_ind_tup = (curr_ind_eth, curr_ind_gen)
    if(curr_ind_tup not in gen_eth_combi_counts.keys()):
        gen_eth_combi_counts[curr_ind_tup] = 1
    else:
        gen_eth_combi_counts[curr_ind_tup] += 1

gen_eth_combi_counts
```

[24]:
```
{('black', 'nonBinary'): 25,
 ('white', 'male'): 1652,
 ('asian', 'male'): 1310,
 ('other', 'female'): 108,
 ('unknown', 'unknown'): 6850,
 ('black', 'male'): 390,
 ('latinx', 'male'): 587,
 ('white', 'female'): 1101,
 ('latinx', 'female'): 354,
 ('nativeAmerican', 'male'): 96,
 ('asian', 'female'): 923,
 ('white', 'nonBinary'): 84,
 ('nativeAmerican', 'female'): 59,
 ('black', 'female'): 257,
 ('other', 'male'): 170,
 ('asian', 'nonBinary'): 63,
 ('other', 'nonBinary'): 8,
 ('latinx', 'nonBinary'): 17,
 ('nativeAmerican', 'nonBinary'): 3}
```

[25]:
```
# Hand transcribed: [white, asian, black, nativeAmerican, other]
row1_M_all = [1652, 1310, 587, 390, 96, 170]
row2_F_all = [1101, 923, 354, 257, 59, 108]
row3_NB_all = [84, 63, 17, 25, 3, 8]
```

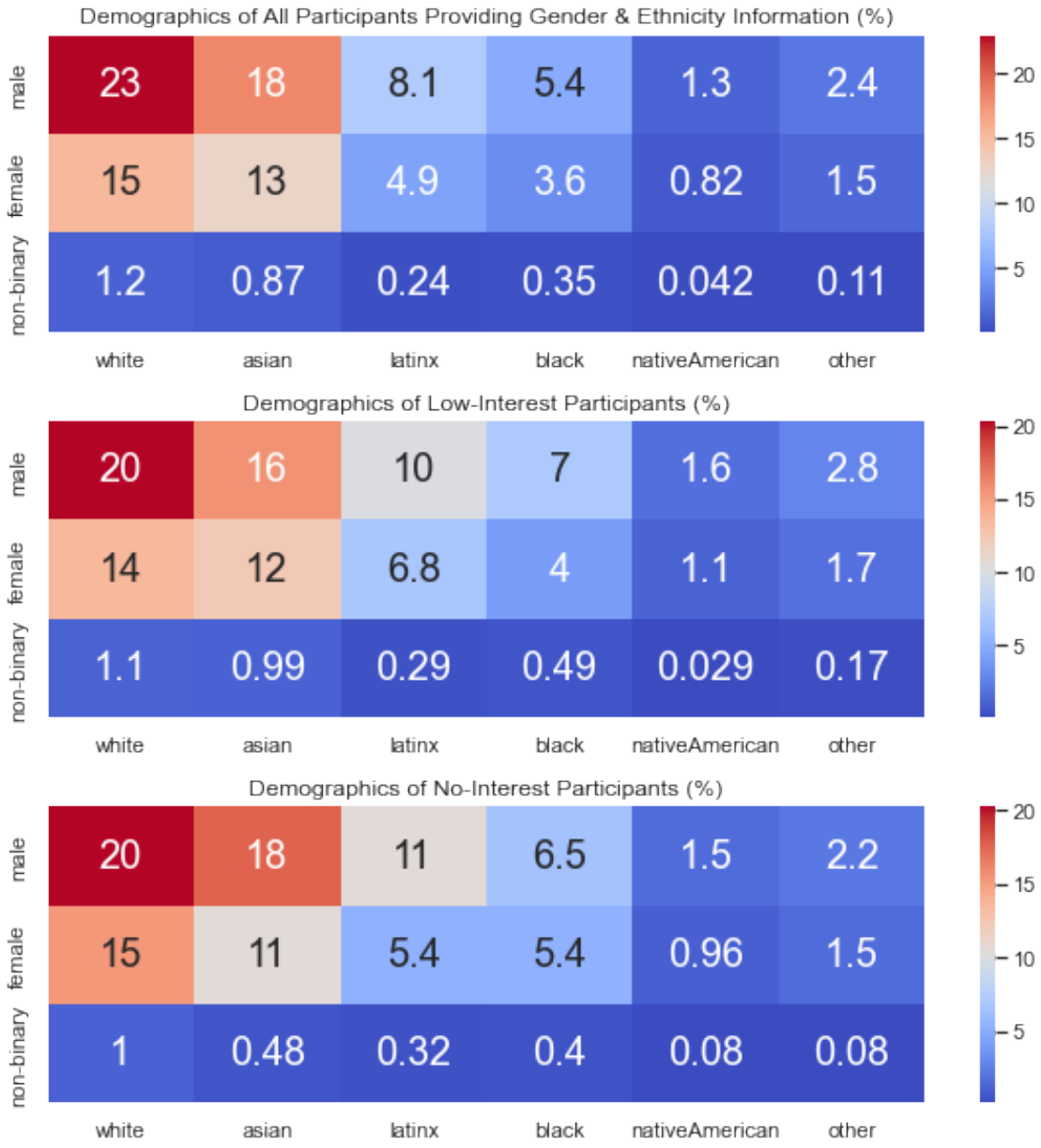
[26]:
```
tot_all = [row1_M_all, row2_F_all, row3_NB_all]
tot_all
```

[26]:
```
[[1652, 1310, 587, 390, 96, 170],
 [1101, 923, 354, 257, 59, 108],
 [84, 63, 17, 25, 3, 8]]
```

[27]:
```
df_all = pd.DataFrame(tot_all, index=genders, columns=ethnicities)
total_all = sum(row1_M_all) + sum(row2_F_all) + sum(row3_NB_all)
```

[28]:
```
fig, axes = plt.subplots(3, 1, figsize=(10, 10), sharey=True)
#fig.suptitle('Demographics of Participants')
```

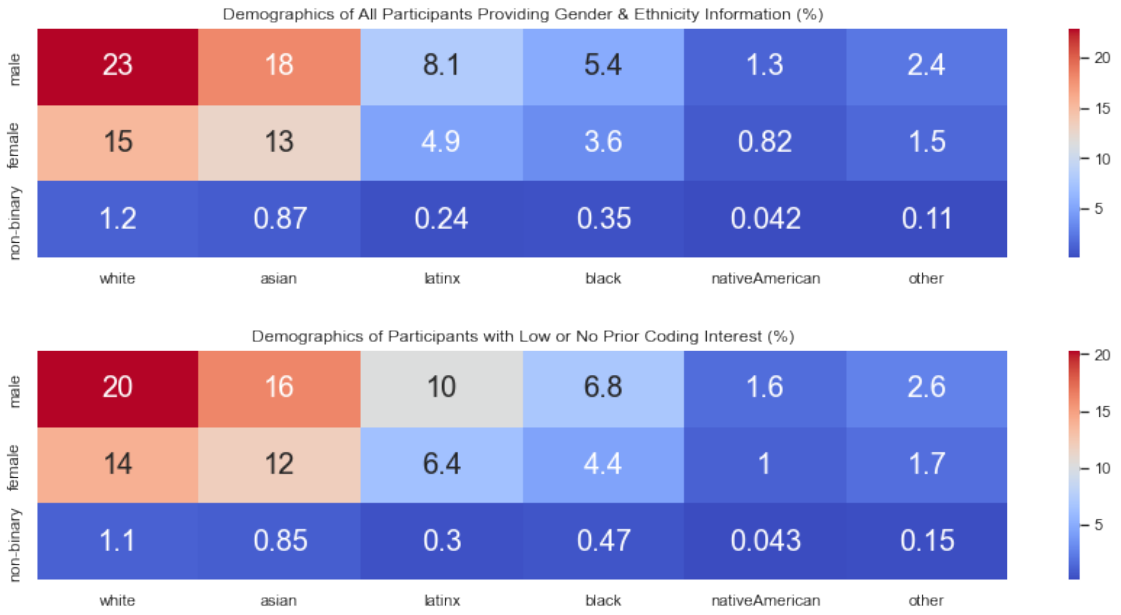
```
# All Percents
sns.heatmap((df_all/total_all)*100, ax=axes[0],annot = True,cmap= 'coolwarm', ␣
 ↪annot_kws={"fontsize":20})
axes[0].set_title('Demographics of All Participants Providing Gender &␣
 ↪Ethnicity Information (%)')

# Low-Interest Percents
sns.heatmap((df_LI/total_LI)*100, ax=axes[1], annot = True,cmap=␣
 ↪'coolwarm',annot_kws={"fontsize":20})
axes[1].set_title('Demographics of Low-Interest Participants (%)')

# No-Interest Percents
sns.heatmap((df_NI/total_NI)*100, ax=axes[2], annot = True,cmap=␣
 ↪'coolwarm',annot_kws={"fontsize":20}, square=False)
axes[2].set_title('Demographics of No-Interest Participants (%)')

plt.subplots_adjust(hspace = 0.3)
plt.savefig('heatplot.png', dpi=300, bbox_inches='tight')
plt.show()
```



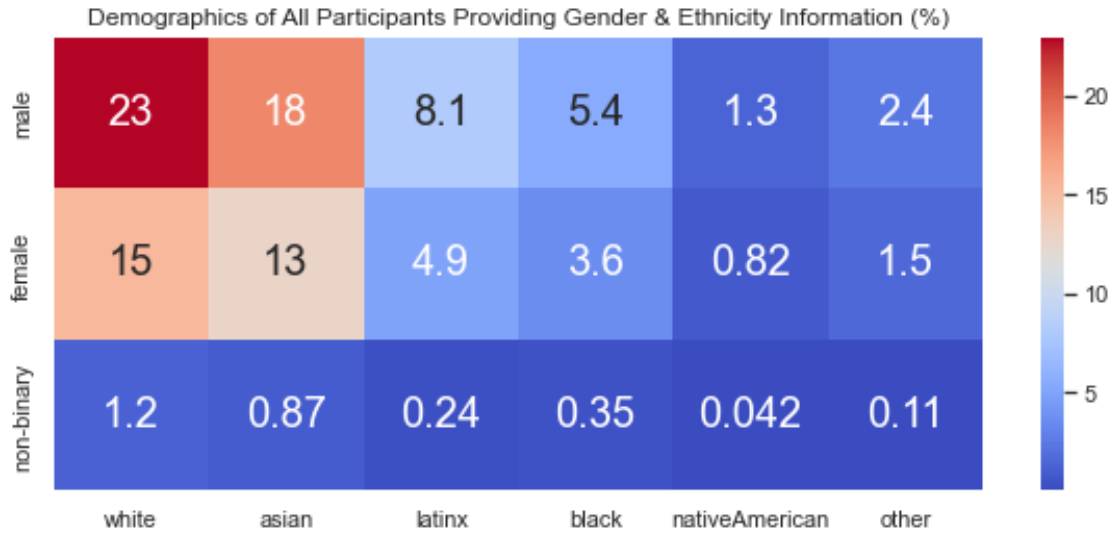
[29]:
```
# All us (low + no interest)
fig, axes = plt.subplots(2, 1, figsize=(15, 7), sharey=True)

# Adjust label pos
pos1 = axes[0].get_position() # get the original position
print(pos1)
pos2 = [pos1.x0, pos1.y0 - 0.5,  pos1.width, pos1.height - 0.5]
axes[0].set_position(pos2) # set a new position
print(axes[0].get_position())
```

```
#fig.suptitle('Demographics of Participants with Low or No Coding Interest')

# All Percents
sns.heatmap((df_all/total_all)*100, ax=axes[0],annot = True,cmap= 'coolwarm', ␣
 ↪annot_kws={"fontsize":20}, square=False)
axes[0].set_title('Demographics of All Participants Providing Gender &␣
 ↪Ethnicity Information (%)')


# Low-No-Interest Percents
sns.heatmap(((df_LI+df_NI)/(total_LI+total_NI))*100, ax=axes[1], annot =␣
 ↪True,cmap= 'coolwarm',annot_kws={"fontsize":20}, square=False)
axes[1].set_title('Demographics of Participants with Low or No Prior Coding␣
 ↪Interest (%)')

# No-Interest Percents
#sns.heatmap((df_NI/total_NI)*100, ax=axes[2], annot = True,cmap=␣
 ↪'coolwarm',annot_kws={"fontsize":20})
#axes[2].set_title('Demographics of No-Interest Participants (%)')

plt.subplots_adjust(hspace = 0.4)
plt.savefig('heatplot.png', dpi=300, bbox_inches='tight')
plt.show()
```

```
Bbox(x0=0.125, y0=0.5368181818181819, x1=0.9, y1=0.88)
Bbox(x0=0.125, y0=0.03681818181818186, x1=0.9, y1=-0.12)
```



[30]:
```
# All vs (low + no interest)
#fig, axes = plt.subplots(1, 1, figsize=(15, 7), sharey=True)

# Adjust label pos
#pos1 = axes[0].get_position() # get the original position
#print(pos1)
#pos2 = [pos1.x0, pos1.y0 - 0.5,  pos1.width, pos1.height - 0.5]
#axes[0].set_position(pos2) # set a new position
#print(axes[0].get_position())

#fig.suptitle('Demographics of Participants with Low or No Coding Interest')

# All Percents
sns.set(rc = {'figure.figsize':(10,4)})
ax = sns.heatmap((df_all/total_all)*100,annot = True,cmap= 'coolwarm', ␣
 ↪annot_kws={"fontsize":20}, square=False)
ax.set_title('Demographics of All Participants Providing Gender & Ethnicity␣
 ↪Information (%)')


# Low-No-Interest Percents
#sns.heatmap(((df_LI+df_NI)/(total_LI+total_NI))*100, ax=axes[1], annot =␣
 ↪True,cmap= 'coolwarm',annot_kws={"fontsize":20}, square=False)
#axes[1].set_title('Demographics of Participants with Low or No Prior Coding␣
 ↪Interest (%)')

# No-Interest Percents
#sns.heatmap((df_NI/total_NI)*100, ax=axes[2], annot = True,cmap=␣
 ↪'coolwarm',annot_kws={"fontsize":20})
#axes[2].set_title('Demographics of No-Interest Participants (%)')

plt.subplots_adjust(hspace = 0.8)
plt.savefig('heatplot.png', dpi=600, bbox_inches='tight')
plt.show()
```

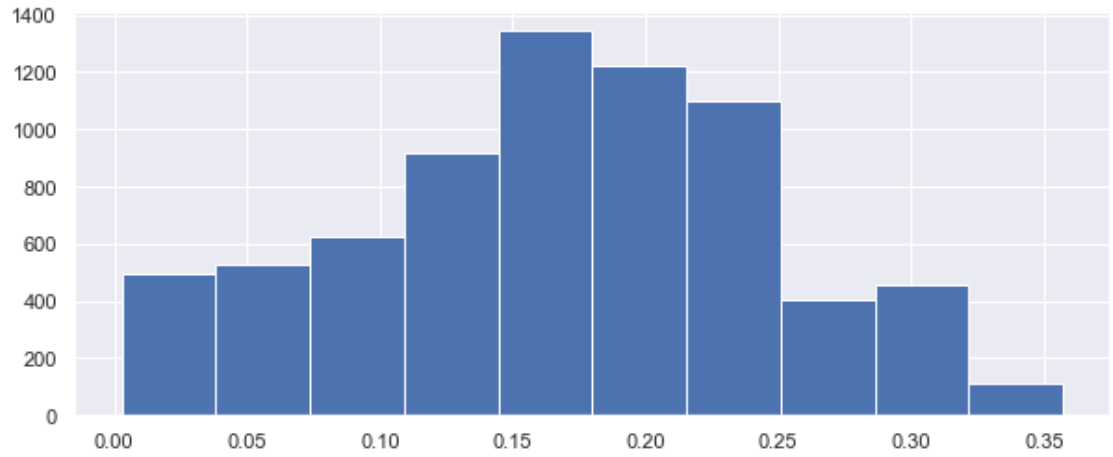


0.2.1 Histograms: Ethnicity Proportions

NOTE: Remember that the 'ethnicity_pct' measure includes the attendees for whome gender and ethnicity are 'unknown', so the ethnicity proportions represented in the histograms below are **minimum** values.

[31]:
```
plt.hist(no_unknowns['ethnicity_pct'])
```

[31]:
```
(array([ 492.,  530.,  628.,  919., 1344., 1225., 1100.,  402.,  454.,
         113.]),
 array([0.002849  , 0.03827839, 0.07370777, 0.10913716, 0.14456654,
        0.17999593, 0.21542532, 0.2508547 , 0.28628409, 0.32171347,
        0.35714286]),
 <BarContainer object of 10 artists>)
```

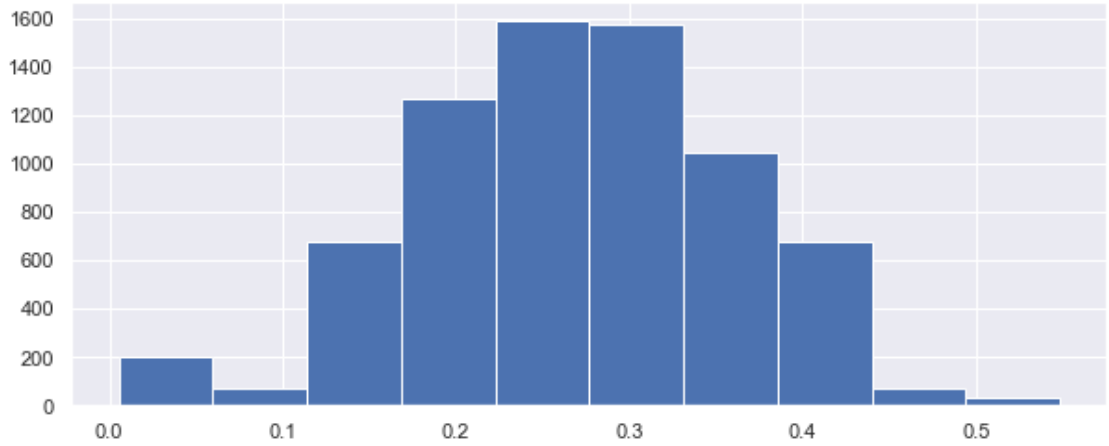


[32]:
```
print(min(no_unknowns['ethnicity_pct']))
print(max(no_unknowns['ethnicity_pct']))
```

```
0.0028490028490028
0.3571428571428571
```

[33]:
```
plt.hist(no_unknowns['gender_pct'])
```

[33]:
```
(array([ 198.,   73.,  674., 1264., 1588., 1576., 1045.,  679.,   74.,
          36.]),
 array([0.00543478, 0.05973001, 0.11402525, 0.16832048, 0.22261571,
        0.27691094, 0.33120617, 0.3855014 , 0.43979663, 0.49409187,
        0.5483871 ]),
 <BarContainer object of 10 artists>)
```

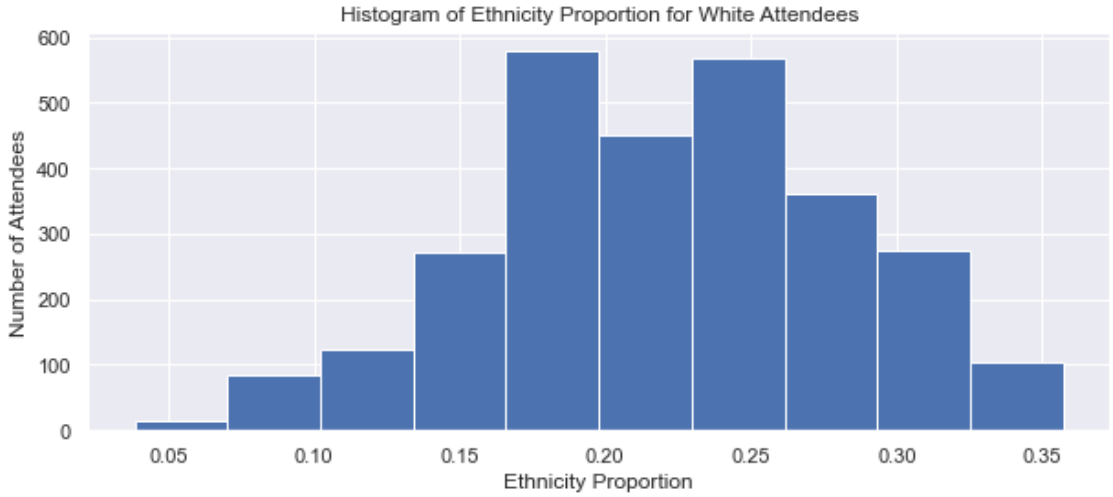


[34]:
```
print(min(no_unknowns['gender_pct']))
print(max(no_unknowns['gender_pct']))
```

```
0.0054347826086956
0.5483870967741935
```

[35]:
```
white = data[(data['ethnicity'] == "white")]
black = data[(data['ethnicity'] == "black")]
latinx = data[(data['ethnicity'] == "latinx")]
asian = data[(data['ethnicity'] == "asian")]
nativeam = data[(data['ethnicity'] == "nativeAmerican")]
other = data[(data['ethnicity'] == "other")]
```

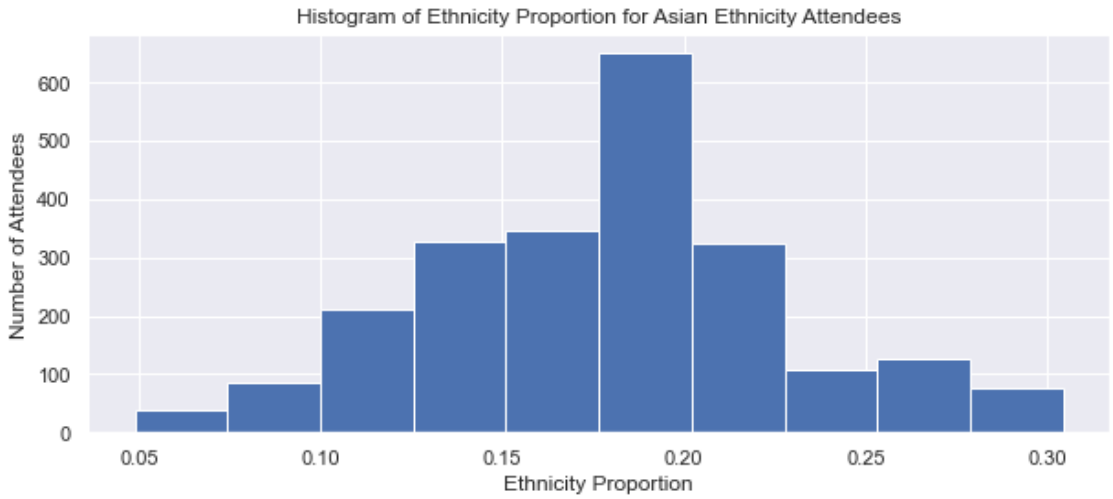
[36]:
```
_ = plt.hist(white['ethnicity_pct'], bins=10)  # arguments are passed to np.
  ↪histogram
plt.title("Histogram of Ethnicity Proportion for White Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



[37]:
```
print("(mean:", white['ethnicity_pct'].mean(),", min:
  ↪",min(white['ethnicity_pct']),", max:",max(white['ethnicity_pct']),")")
```

```
(mean: 0.22197984630472864 , min: 0.0384615384615384 , max: 0.3571428571428571 )
```

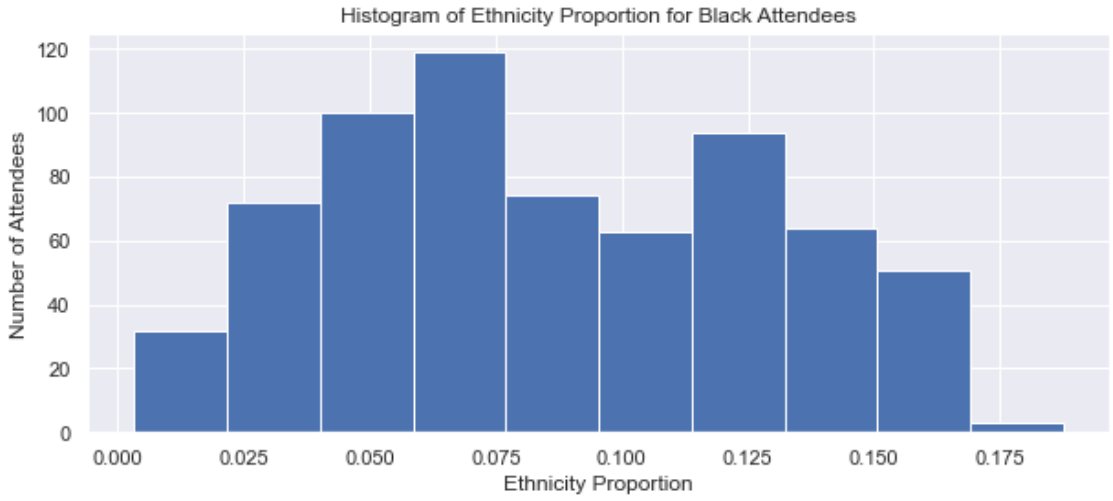
[38]:
```
_ = plt.hist(asian['ethnicity_pct'], bins=10)  # arguments are passed to np.
  ↪histogram
plt.title("Histogram of Ethnicity Proportion for Asian Ethnicity Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



[39]:
```
print("( mean:", asian['ethnicity_pct'].mean(),", min:
  ↪",min(asian['ethnicity_pct']),", max:",max(asian['ethnicity_pct']),")")
```

```
( mean: 0.17841280054620842 , min: 0.048780487804878 , max: 0.3043478260869565 )
```

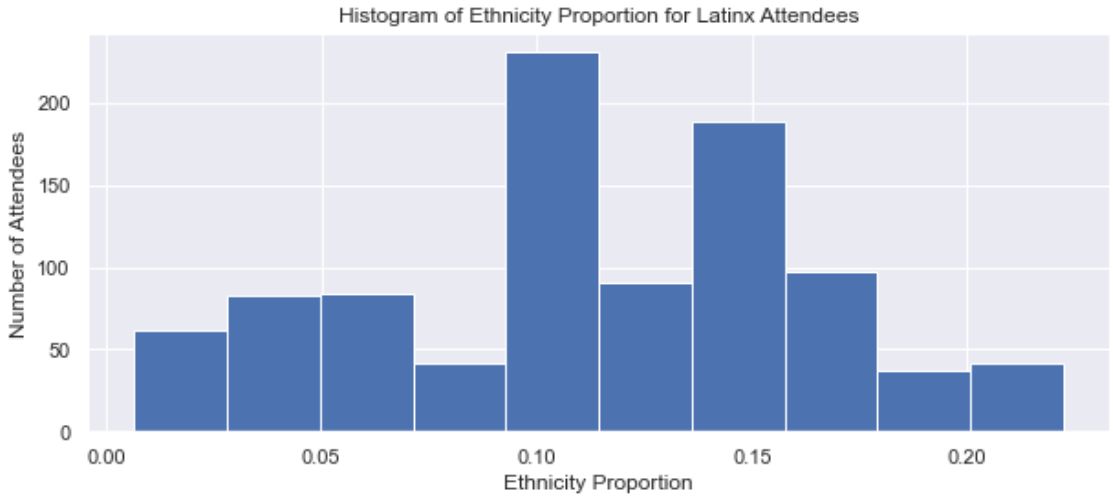
[40]:
```
_ = plt.hist(black['ethnicity_pct'], bins=10)  # arguments are passed to np.
  ↪histogram
plt.title("Histogram of Ethnicity Proportion for Black Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



[41]:
```
print("(mean:", black['ethnicity_pct'].mean(),", min:
  ↪",min(black['ethnicity_pct']),", max:",max(black['ethnicity_pct']),")")
```

```
(mean: 0.0853148361933327 , min: 0.0032894736842105 , max: 0.1875 )
```

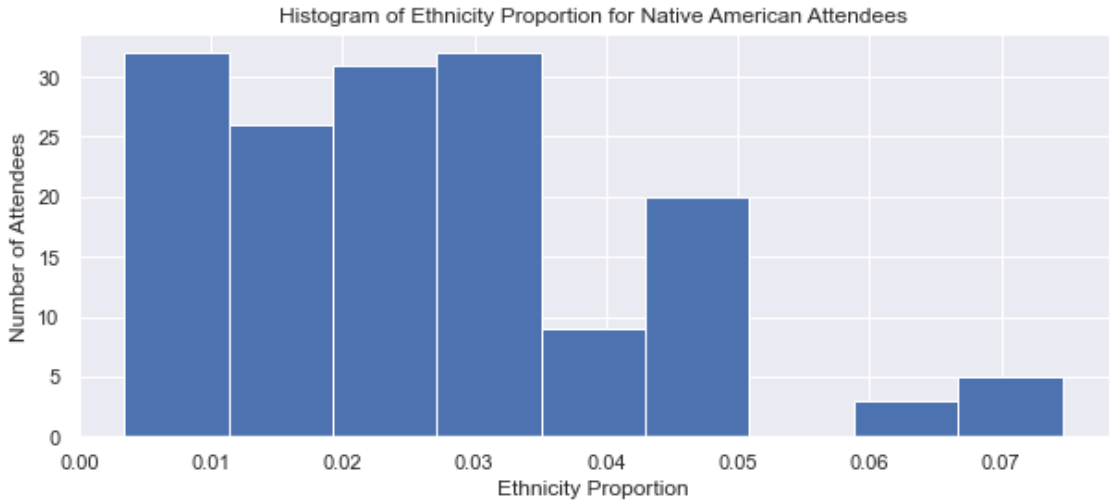
[42]:
```
_ = plt.hist(latinx['ethnicity_pct'], bins=10)  # arguments are passed to np.
  ↪histogram
plt.title("Histogram of Ethnicity Proportion for Latinx Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



[43]:
```
print("(mean:", latinx['ethnicity_pct'].mean(),", min:
  ↪",min(latinx['ethnicity_pct']),", max:",max(latinx['ethnicity_pct']),")")
```

```
(mean: 0.11288875840344274 , min: 0.006578947368421 , max: 0.2222222222222222 )
```

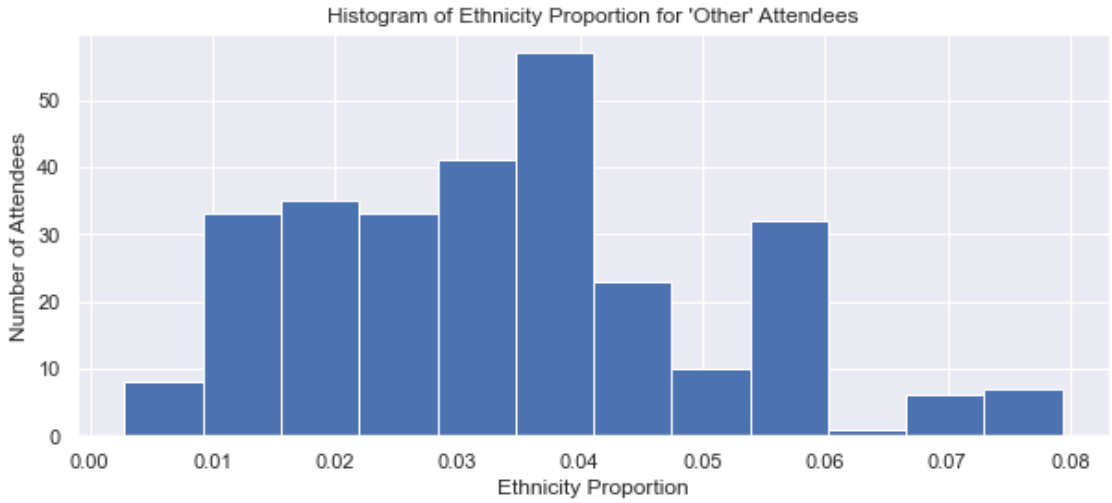
[44]:
```
_ = plt.hist(nativeam['ethnicity_pct'], bins="auto")  # arguments are passed to␣
  ↪np.histogram
plt.title("Histogram of Ethnicity Proportion for Native American Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



[45]:
```
print("(mean:", nativeam['ethnicity_pct'].mean(),", min:
  ↪",min(nativeam['ethnicity_pct']),", max:",max(nativeam['ethnicity_pct']),")")
```

```
(mean: 0.026371237645922956 , min: 0.0034364261168384 , max: 0.0746268656716417
)
```

[46]:
```
_ = plt.hist(other['ethnicity_pct'], bins="auto")  # arguments are passed to np.
  ↪histogram
plt.title("Histogram of Ethnicity Proportion for 'Other' Attendees")
plt.xlabel("Ethnicity Proportion")
plt.ylabel("Number of Attendees")
plt.show()
```



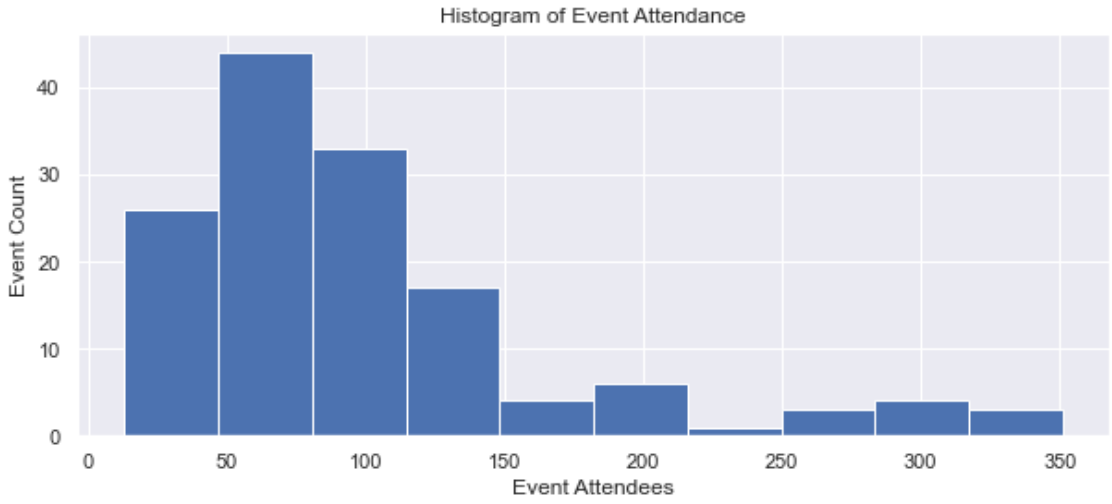
[47]:
```
print("(mean:", other['ethnicity_pct'].mean(),", min:
 ↪",min(other['ethnicity_pct']),", max:",max(other['ethnicity_pct']),")")
```

```
(mean: 0.034464735358817285 , min: 0.0028490028490028 , max: 0.0793650793650793
)
```

0.2.2 Histogram of Event Sizes

[48]:
```
data["eventregion"] = data["newEventId"].astype(str) + "_" +␣
 ↪data["newRegionId"].astype(str)
```

[50]:
```
_ = plt.hist(data['eventregion'].value_counts(), bins=10)  # arguments are␣
 ↪passed to np.histogram
plt.title("Histogram of Event Attendance")
plt.xlabel("Event Attendees")
plt.ylabel("Event Count")
plt.show()
```



[51]:
```
data['eventregion'].value_counts()
```

[51]:
```
1_0      351
0_1      349
1_2      320
6_0      304
5_1      293
        ...
5_10      26
2_26      25
3_24      18
```

```
2_25     16
2_24     13
Name: eventregion, Length: 141, dtype: int64
```

[52]:
```
print('Range of event attendance values:')
print('\tSmallest event:', min(data['eventregion'].value_counts()))
print('\tLargest event:', max(data['eventregion'].value_counts()))
```

```
Range of event attendance values:
        Smallest event: 13
        Largest event: 351
```

[ ]:
```

```