Result_analysis

May 25, 2021

1 Creative car design

In this Jupyter Notebook the results of the external evaluation performed for the creative car design system is performed.

1.1 Table of contents

1.2 Student info

- **Name**: Bontinck Lennert
- **StudentID**: 568702
- **Affiliation**: VUB - Master Computer Science: AI

1.3 Required libraries

The following code block loads in the required libraries. Anaconda environments should have them installed by default.

[1]:
```
# Numerical manipulations libraries
import pandas as pd
import numpy as np

# plotting libraries
import matplotlib.pyplot as plt
import seaborn as sn
```

1.4 Loading in the results

The following code block loads in the final results as they were retreived from the export page of the external evaluation tool.

[2]:
```
bias = pd.read_csv(r'Data\bias.csv')
images = pd.read_csv(r'Data\images.csv')
participants = pd.read_csv(r'Data\participants.csv')
ratings_grouped = pd.read_csv(r'Data\ratings_grouped.csv')
ratings_single = pd.read_csv(r'Data\ratings_single.csv')
```

The following code blocks will validate the loaded in data is correct.

[3]:
```
print("-------- bias dataset --------")
display(bias.head())
print("-------- images dataset --------")
display(images.head())
print("-------- participants dataset --------")
display(participants.head())
print("-------- ratings_grouped dataset --------")
display(ratings_grouped.head())
print("-------- ratings_single dataset --------")
display(ratings_single.head())
```

```
-------- bias dataset --------
```

	participant_id	biased	note
0	1	nonbiased	Got me
1	2	nonbiased	I think I wasnt biased, but wouldnt know really
2	3	nonbiased	NaN
3	5	nonbiased	NaN
4	6	nonbiased	NaN

```
-------- images dataset --------
```

	image_id	filename	path	grouped
0	1	grouped_1	images/grouped/grouped_1.png	1
1	2	grouped_2	images/grouped/grouped_2.png	1
2	3	grouped_3	images/grouped/grouped_3.png	1
3	4	grouped_4	images/grouped/grouped_4.png	1
4	5	single1	images/single/single1.png	0

```
-------- participants dataset --------
```

```
   participant_id gender  age  expertise  colorblind  bad_vision
0               1   male   20          0           0           0
1               2   male   20          1           0           0
2               3   male   30          0           1           0
3               4   male   20          0           0           0
4               5   male   20          1           0           0

-------- ratings_grouped dataset --------

   participant_id  image_id  correspondence  realism  creative   made_by  \
0               1         1               5        4         3     known
1               1         2               5        4         4  computer
2               1         3               5        4         4     known
3               1         4               2        3         2     human
4               2         1               5        5         1     human

                                          note
0        Just turns red but cool it is the car only
1                            Nose changes i think
2                           Nice change of wheels
3  Looks like completely different cars to me wou…
4                  Its just a car that turns red…

-------- ratings_single dataset --------

   participant_id  image_id  carlike  detail  realism  resemblence  creative  \
0               1         5        1       1        1            1         3
1               1         6        5       4        3            3         4
2               1         7        5       5        5            5         4
3               1         8        5       4        4            5         3
4               1         9        5       5        4            4         3

   general_impression   made_by                                    note
0                   1  computer                    Unique but not a car
1                   2     known  Hmmm not sure if that is two fronts haha
2                   4     human                                     NaN
3                   4  computer                                     NaN
4                   3     known                       cool text, meh car
```

1.5 Cleaning dataset and analysing participants data

The following code blocks remove entrees of participants that didn't complete the survey completely. It also gives some basic statistics about the participants that did complete the survey.

[4]:
```
# Determine amount of participants that registered
# and show some other statistics
participants_started = participants.shape[0]
singles_ratings_started = ratings_single.shape[0]
grouped_ratings_started = ratings_grouped.shape[0]
biased_ratings_started = bias.shape[0]
```

```
print(f"There are {participants_started} participants that started the survey.")
print(f"There are a total of {singles_ratings_started} singles_ratings_started␣
 ↪entrees in the original dataset.")
print(f"There are a total of {grouped_ratings_started} grouped_ratings_started␣
 ↪entrees in the original dataset.")
print(f"There are a total of {biased_ratings_started} biased_ratings_started␣
 ↪entrees in the original dataset.")
```

```
There are 32 participants that started the survey.
There are a total of 183 singles_ratings_started entrees in the original
dataset.
There are a total of 93 grouped_ratings_started entrees in the original dataset.
There are a total of 23 biased_ratings_started entrees in the original dataset.
```

[5]:
```
# Get a list of participant_ids that rated all grouped images
participants_rated_all_grouped = ratings_grouped.
 ↪groupby('participant_id')['participant_id'].filter(lambda x: len(x) == 4)

# Get a list of participant_ids that rated all single images
participants_rated_all_single = ratings_single.
 ↪groupby('participant_id')['participant_id'].filter(lambda x: len(x) == 8)

# Get a list of participant_ids that submitted bias
participants_gave_bias = bias.participant_id

# Remove participants that didn't fill in all ratings and bias
participants = participants[participants['participant_id'].
 ↪isin(participants_rated_all_grouped) &
                             (participants['participant_id'].
 ↪isin(participants_rated_all_single)) &
                             (participants['participant_id'].
 ↪isin(participants_gave_bias))]

# Get simple list of all ok participants
finished_participants = participants.participant_id

# Only keep results that come from ok participants
bias = bias[bias['participant_id'].isin(finished_participants)]
ratings_grouped = ratings_grouped[ratings_grouped['participant_id'].
 ↪isin(finished_participants)]
ratings_single = ratings_single[ratings_single['participant_id'].
 ↪isin(finished_participants)]
```

[6]:
```
# Determine amount of participants that completed the survey
# and show some other statistics
participants_finished = participants.shape[0]
```

```
singles_ratings_finished = ratings_single.shape[0]
grouped_ratings_finished = ratings_grouped.shape[0]
biased_ratings_finished = bias.shape[0]
expert_ratings_finished = participants[participants.expertise == 1].shape[0]
expert_ratings_male_finished = participants[(participants.expertise == 1) &␣
 ↪(participants.gender == "male")].shape[0]
males_finished = participants[participants.gender == "male"].shape[0]
females_finished = participants[participants.gender == "female"].shape[0]
others_finished = participants[participants.gender == "other"].shape[0]
age_mode = participants.age.mode()[0]
age_avg = participants.age.mean()
age_std = participants.age.std()
colorblind_count = participants[participants.colorblind == 1].shape[0]
badvision_count = participants[participants.bad_vision == 1].shape[0]


print(f"There are {participants_started} participants that completed the survey.
 ↪")
print(f"There are a total of {singles_ratings_started} singles_ratings_started␣
 ↪entrees in the cleaned dataset.")
print(f"There are a total of {grouped_ratings_started} grouped_ratings_started␣
 ↪entrees in the cleaned dataset.")
print(f"There are a total of {biased_ratings_started} biased_ratings_started␣
 ↪entrees in the cleaned dataset.")
print(f"There are a total of {expert_ratings_finished} expertised ratingsinfor␣
 ↪the cleaned dataset, {expert_ratings_male_finished} males.")
print(f"There are a total of {males_finished} male participants in the cleaned␣
 ↪dataset.")
print(f"There are a total of {females_finished} female participants in the␣
 ↪cleaned dataset.")
print(f"There are a total of {others_finished} other gendered participants in␣
 ↪the cleaned dataset.")
print(f"The mode of the participants age is {age_mode} with an average of␣
 ↪{age_avg} having a standard deviation of {age_std}.")
print(f"{colorblind_count} participant(s) were colorblind and {badvision_count}␣
 ↪had bad vision.")
```

```
There are 32 participants that completed the survey.
There are a total of 183 singles_ratings_started entrees in the cleaned dataset.
There are a total of 93 grouped_ratings_started entrees in the cleaned dataset.
There are a total of 23 biased_ratings_started entrees in the cleaned dataset.
There are a total of 7 expertised ratingsinfor the cleaned dataset, 7 males.
There are a total of 16 male participants in the cleaned dataset.
There are a total of 5 female participants in the cleaned dataset.
There are a total of 0 other gendered participants in the cleaned dataset.
The mode of the participants age is 20 with an average of 23.80952380952381
having a standard deviation of 6.690433824641326.
```

```
1 participant(s) were colorblind and 0 had bad vision.
```

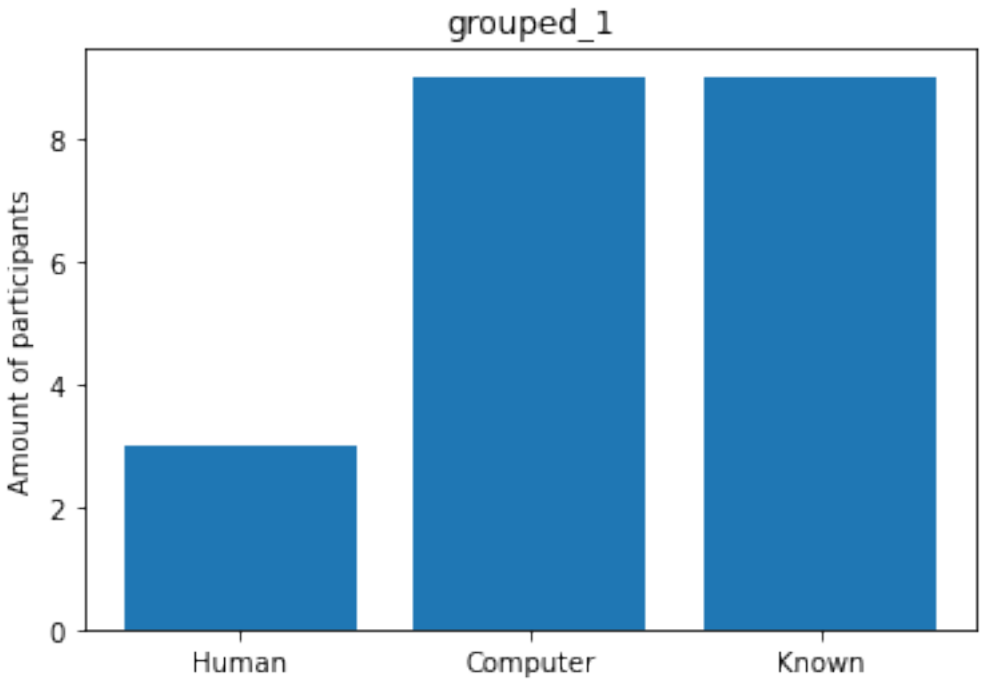
It becomes apparent there were quite a lot of participants who didn't complete the survey, this is likely due to the time it takes to complete a survey being around 15 minutes as well as some early participants having issues with the survey that were fixed later on. The total of 21 ratings of which 7 are expertised is enough to conclude interesting things according to Nick Harley.

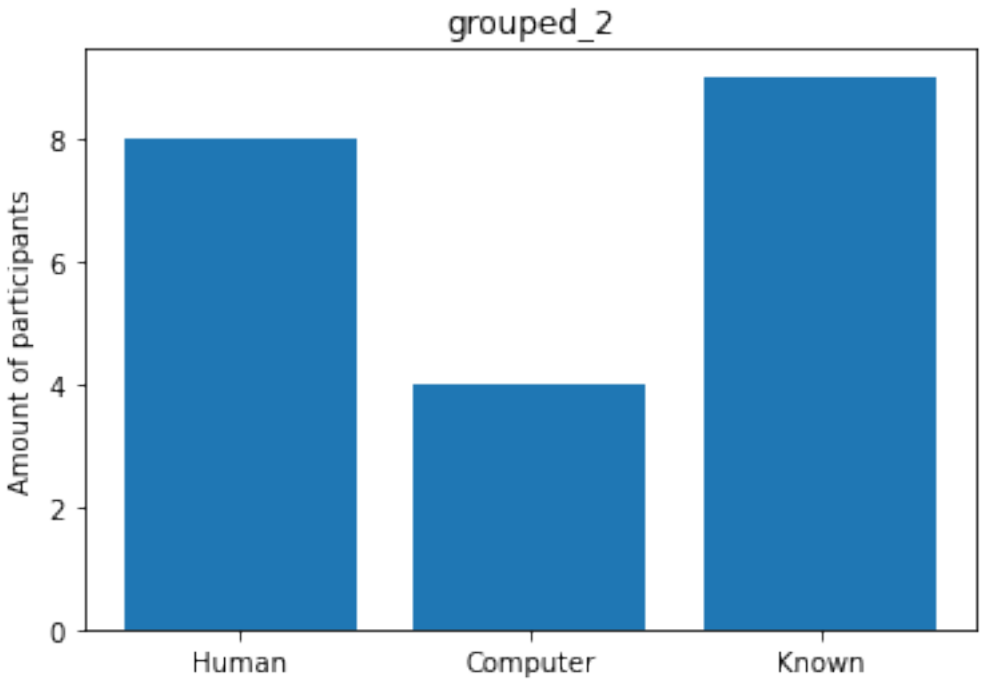
1.6 Human VS computer analysis

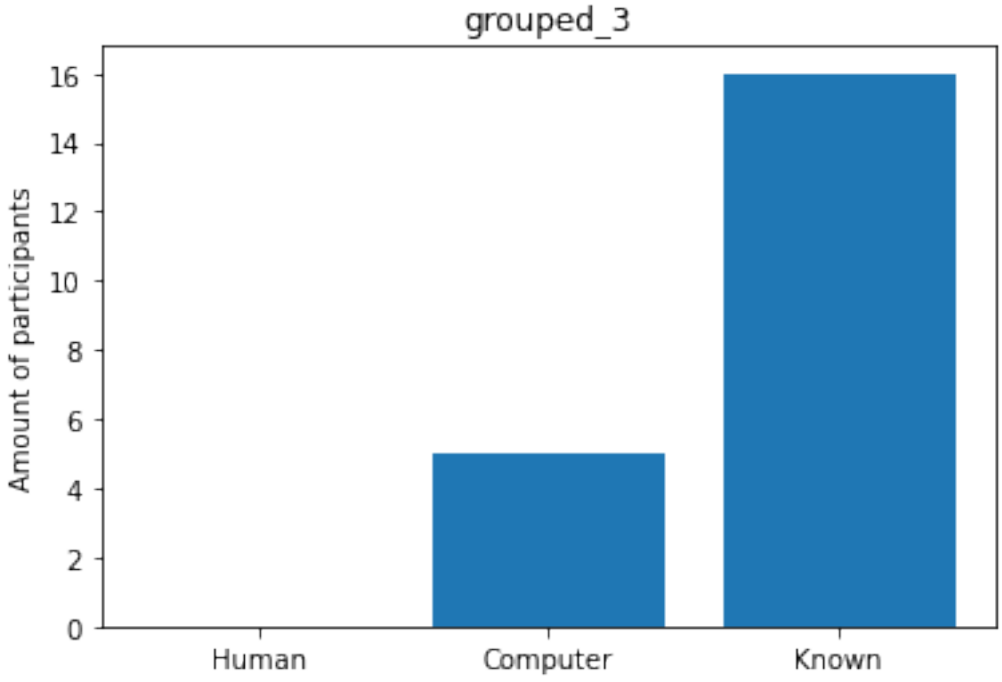
The following code blocks will give the distribution of human, computer and known assignments for the made_by field of each image.

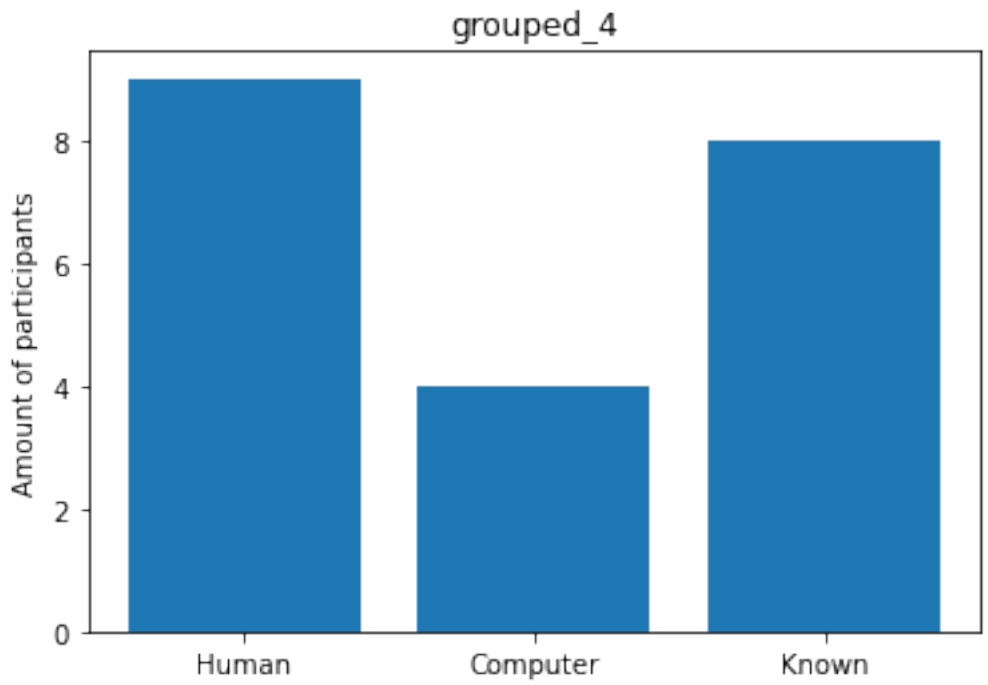
[7]:
```
# Get all images
image_ids = images.image_id

for image_id in image_ids:
    # Get image name
    image_name = images[images.image_id == image_id].filename.item()

    # Determine grouped or single image
    is_grouped = images[images.image_id == image_id].grouped.item()

    # Calculate values
    if (is_grouped):
        values = [ratings_grouped[(ratings_grouped.image_id == image_id) &␣
↪(ratings_grouped.made_by == "human")].shape[0],
                  ratings_grouped[(ratings_grouped.image_id == image_id) &␣
↪(ratings_grouped.made_by == "computer")].shape[0],
                  ratings_grouped[(ratings_grouped.image_id == image_id) &␣
↪(ratings_grouped.made_by == "known")].shape[0]]
    else:
        values = [ratings_single[(ratings_single.image_id == image_id) &␣
↪(ratings_single.made_by == "human")].shape[0],
                  ratings_single[(ratings_single.image_id == image_id) &␣
↪(ratings_single.made_by == "computer")].shape[0],
                  ratings_single[(ratings_single.image_id == image_id) &␣
↪(ratings_single.made_by == "known")].shape[0]]

    x_ticks = ["Human", "Computer", "Known"]

    # Replace NaN with 0 - occurs when certain rating doesn't occur
    values = pd.Series(values).fillna(0).tolist()

    x = np.arange(len(values))
    y = values

    # Show plot
    plt.title(image_name)
```

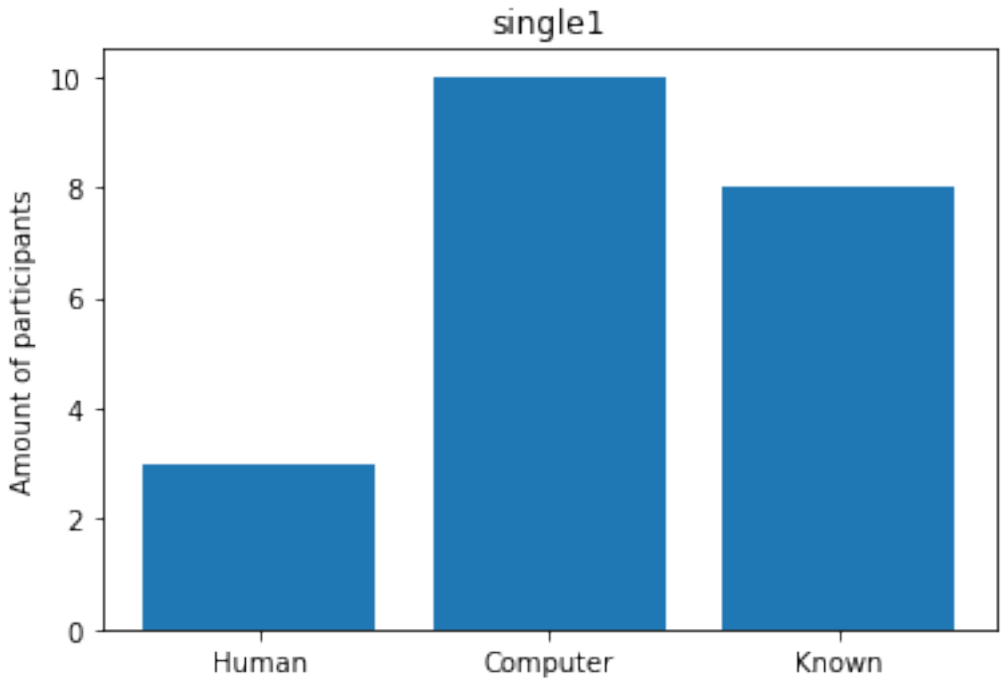
```
    plt.ylabel('Amount of participants')
    plt.xticks(x, x_ticks)

    plt.bar(x, y, align='center', capsize=5)

    plt.savefig("Graphs/Made_by/" + image_name, dpi=300, bbox_inches='tight')
    plt.show()
```

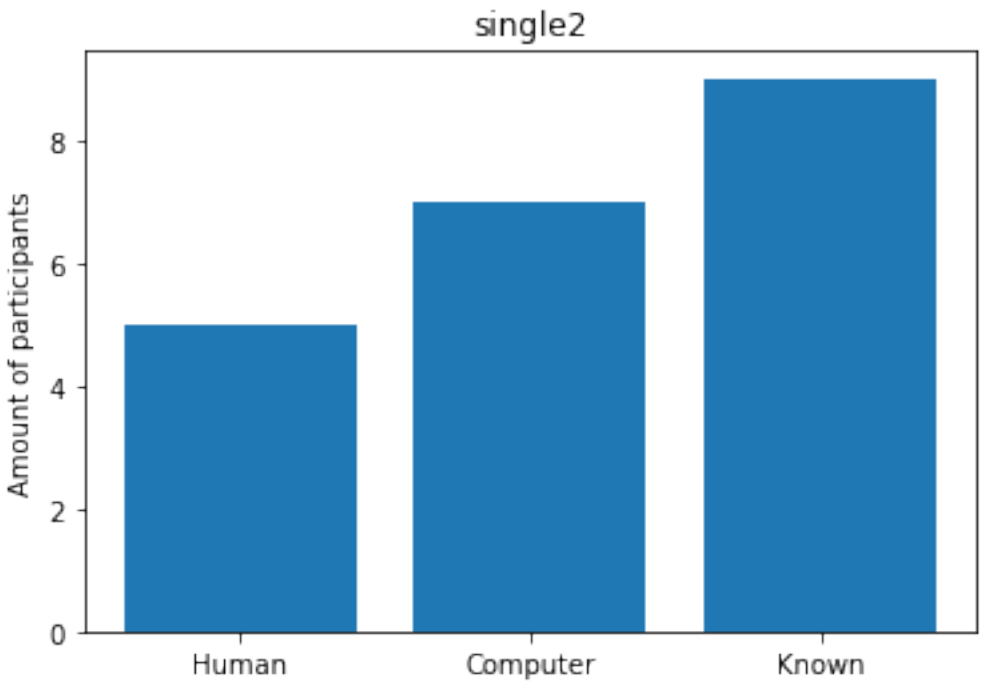


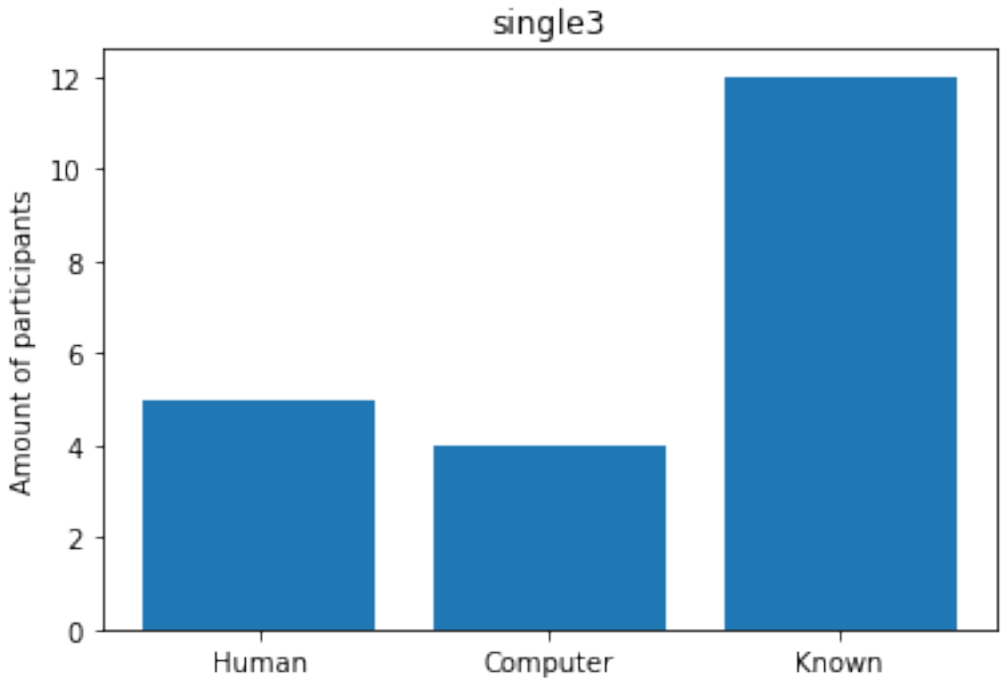

grouped_2
Amount of participants
0
2
4
6
8
Human
Computer
Known

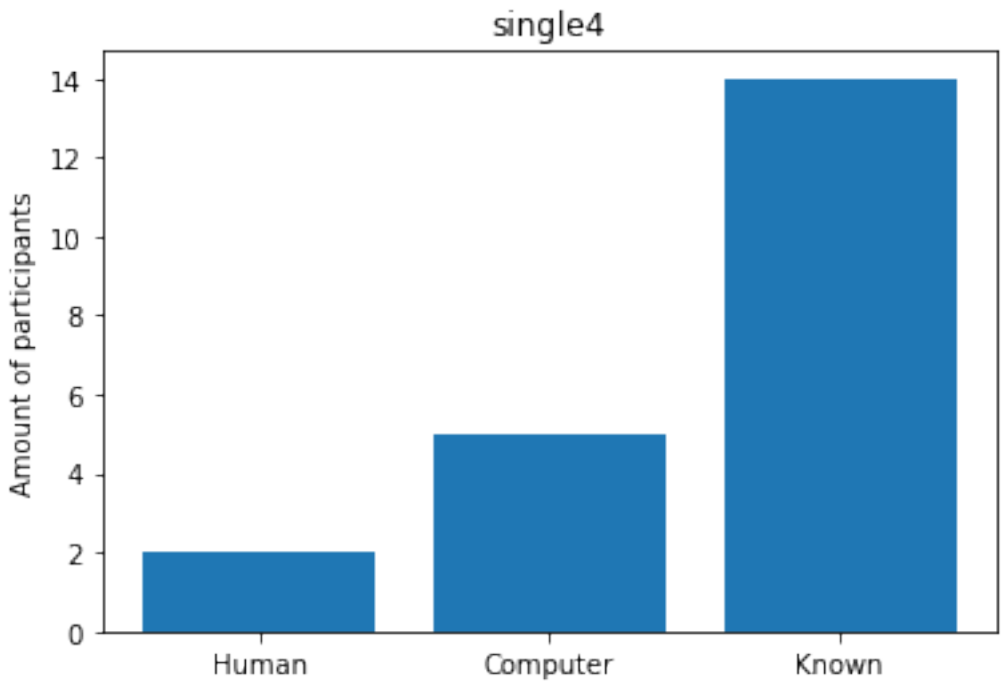

grouped_3
Amount of participants
0
2
4
6
8
10
12
14
16
Human
Computer
Known

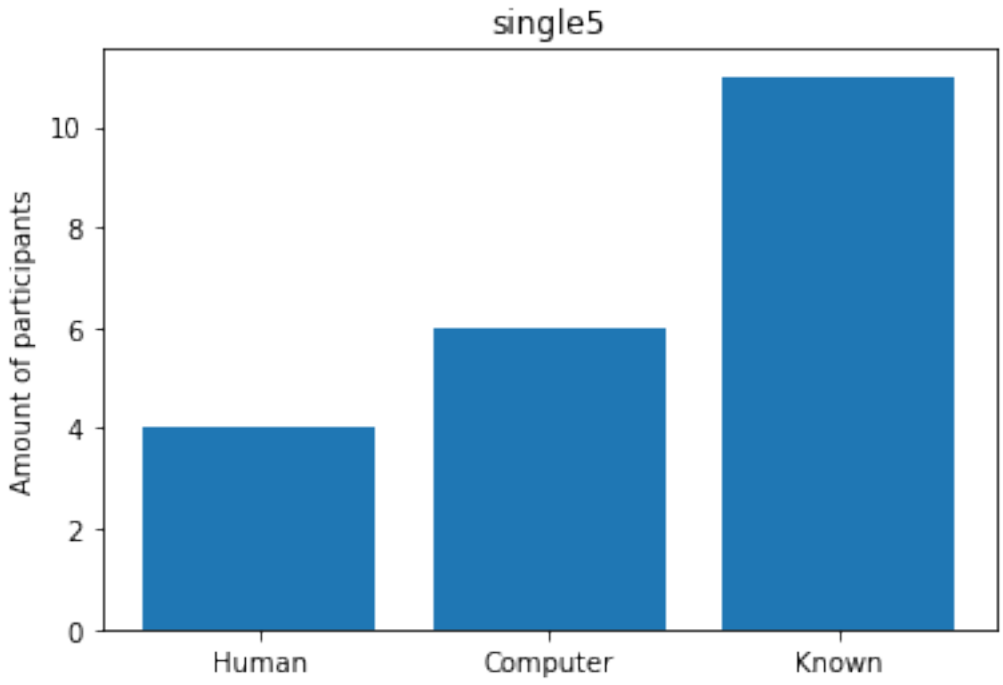

grouped_4
Amount of participants
0
2
4
6
8
Human
Computer
Known

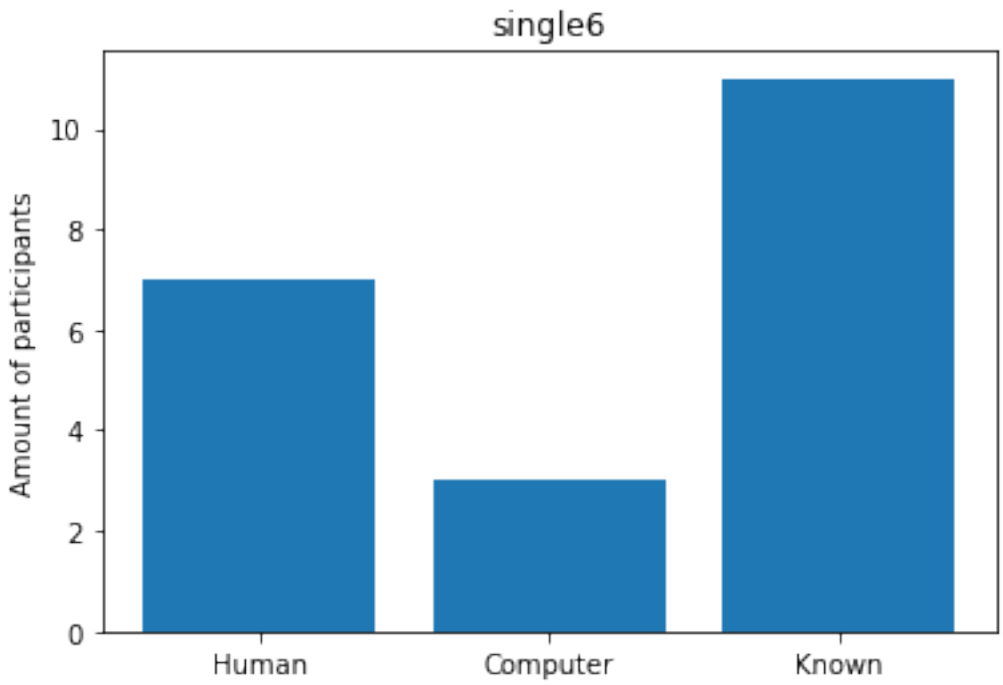

single1
Amount of participants
0
2
4
6
8
10
Human
Computer
Known

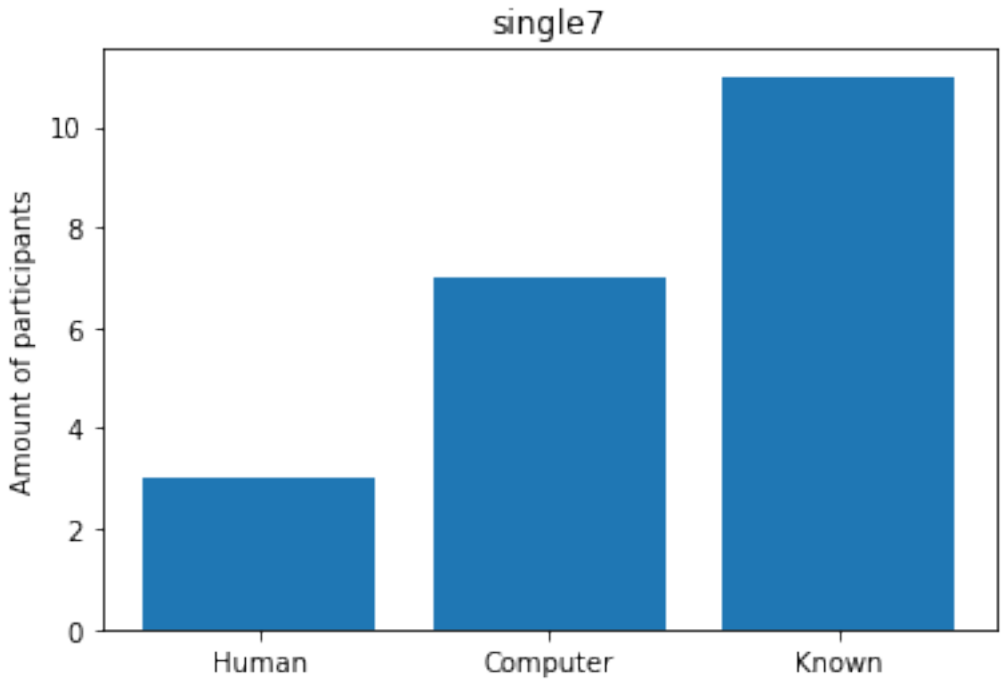

single2
Amount of participants
0
2
4
6
8
Human
Computer
Known

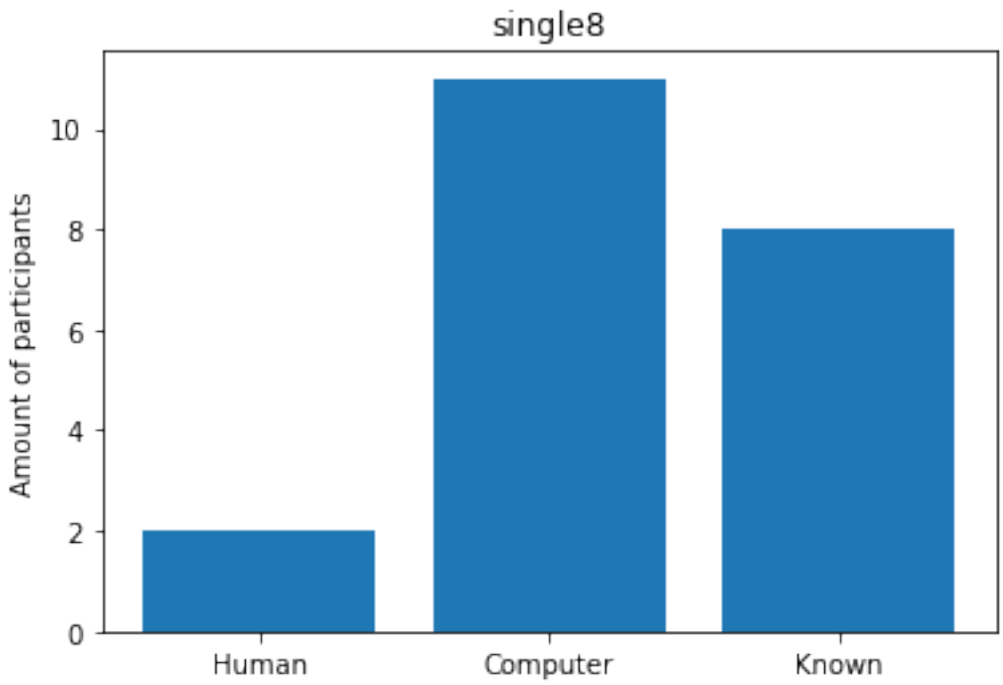

single3
Amount of participants
0
2
4
6
8
10
12
Human
Computer
Known


single4
Amount of participants
0
2
4
6
8
10
12
14
Human
Computer
Known


single5
Amount of participants
0
2
4
6
8
10
Human
Computer
Known


single6
Amount of participants
0
2
4
6
8
10
Human
Computer
Known


single7
Amount of participants
0
2
4
6
8
10
Human
Computer
Known



These results are interesting as it becomes apparant more shocking designs are often assigned to computer whilst more realistic designs are often assigned to human made. It is also interesting that the more minor edits such as rim design and car color were assigned to computer-generated whilst those doing major edits were assigned to human. Since it has become clear that minor edits without much coloteral damage are much harder to do computer-generated.

1.7 Bias analysis

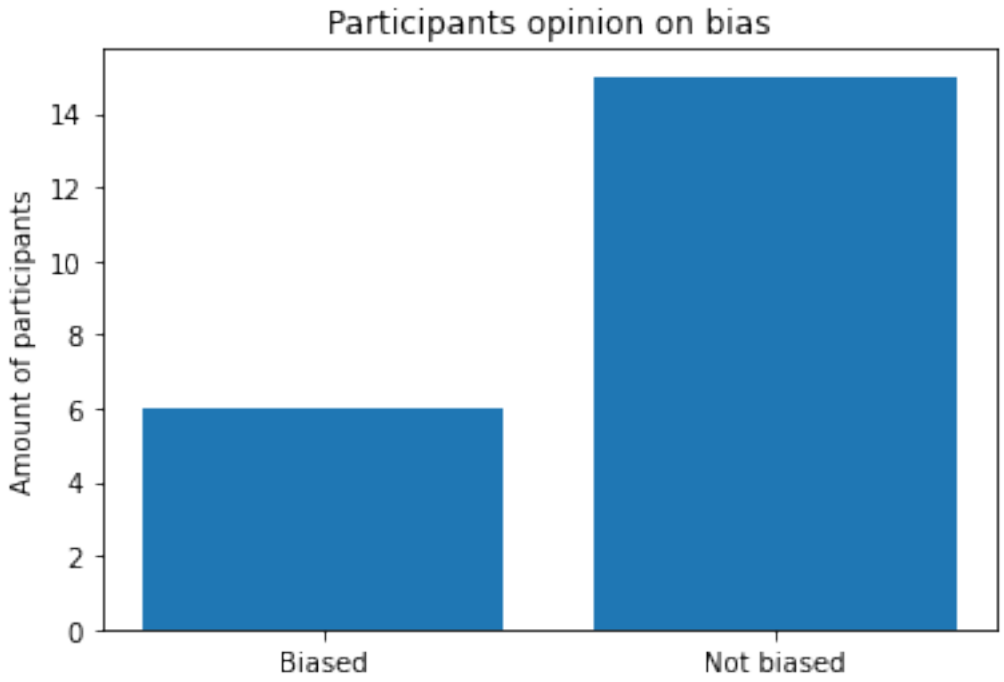
The following code blocks will give some analytics for the biases.

1.7.1 Bias distribution

Here we'll just check the distribution of the bias by making a simple bar plot.

[8]:
```
# Calculate values
values = [bias[(bias.biased == "biased")].shape[0], bias[(bias.biased ==␣
 ↪"nonbiased")].shape[0]]
x_ticks = ["Biased", "Not biased"]

x = np.arange(len(values))
y = values

# Show plot
plt.title("Participants opinion on bias")
plt.ylabel('Amount of participants')
```

```
plt.xticks(x, x_ticks)

plt.bar(x, y, align='center', capsize=5)

plt.savefig("Graphs/Bias/bias_distribution", dpi=300, bbox_inches='tight')
plt.show()
```



1.7.2 Bias notes

This code block will print all notes that were left for the bias question.

[9]:
```
bias_notes = bias.note.dropna()

for bias_note in bias_notes:
    print(bias_note)
    print()
```

```
Got me

I think I wasnt biased, but wouldnt know really

I think i might have assigned the broken looking ones to pc

I dont think I was biased
```

```
semi-biased I would say, since everyone is biased in some way or another

I was a little biased but not by that much. I felt like every image had strange
artifacts but thought they could also just be done by a human. A few images were
shockingly good. Very impressive stuff.

Im not surprised, but I also know literally nothing about cars

Denk dat ik de slechte designs aan pc heb toegekend. Indrukwekkend dat dit
allemaal door pc is
```

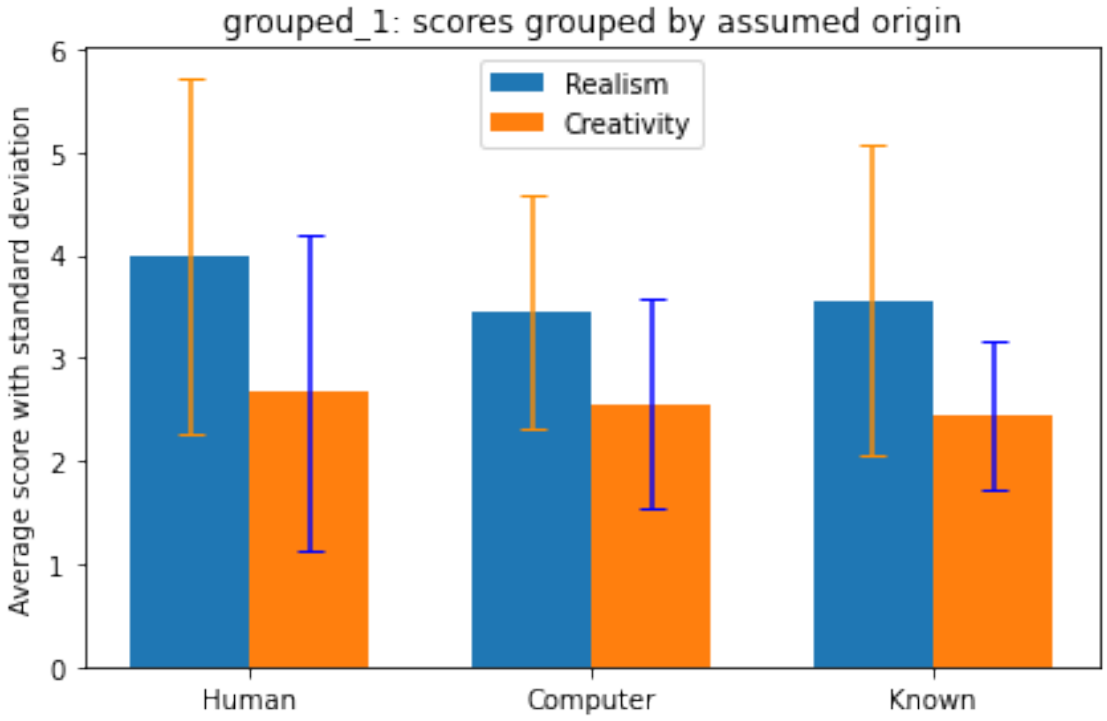
1.7.3 Bias analysis of results

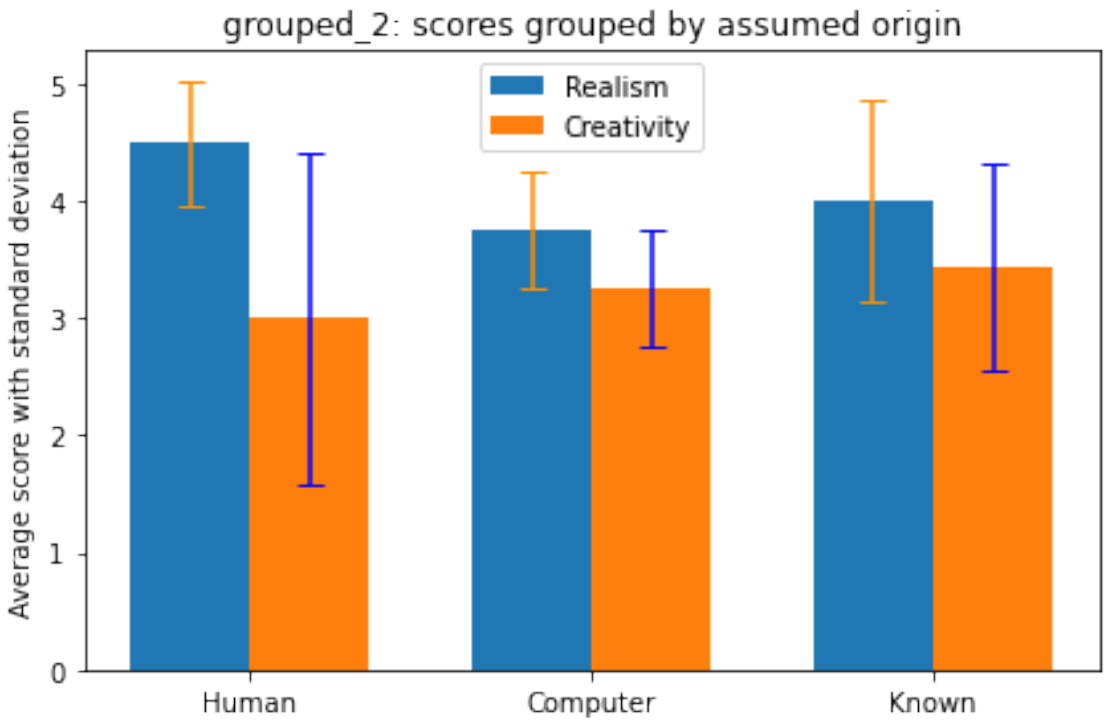
This code block will show the mean and standard deviation of assignments to realism and creativity based on whether the particpant thinks it was computer made, human made or knew it was computer made. It does this for each image.

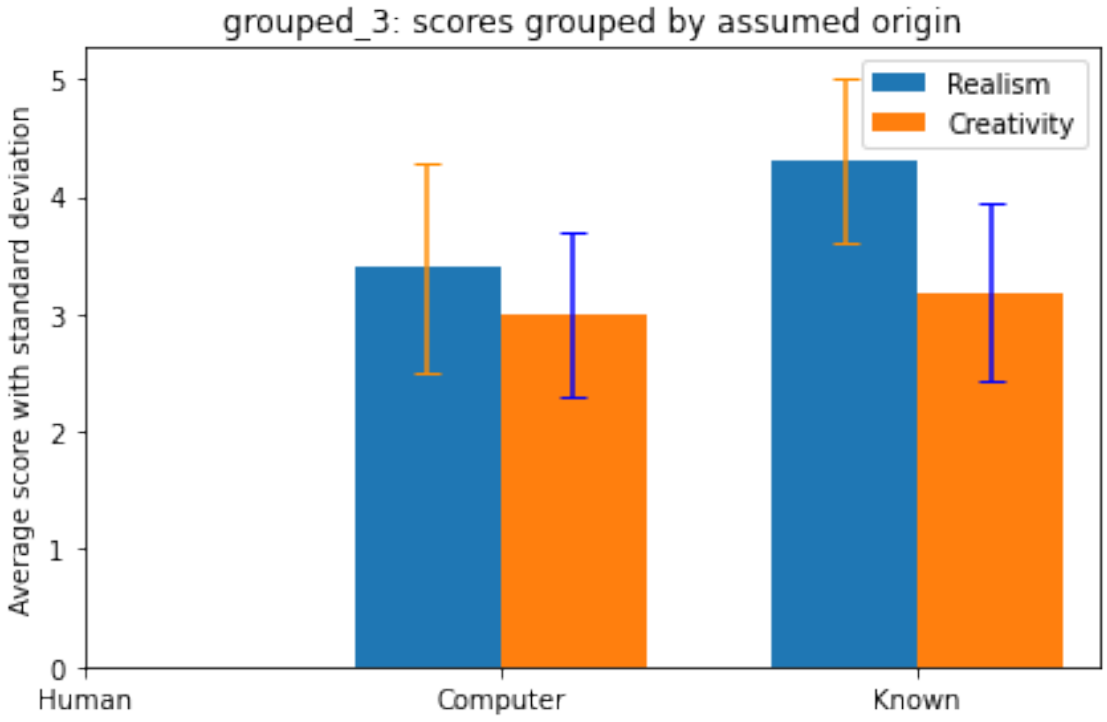
[10]:
```
# Get all images
image_ids = images.image_id

for image_id in image_ids:
    # Get image name
    image_name = images[images.image_id == image_id].filename.item()

    # Determine grouped or single image
    is_grouped = images[images.image_id == image_id].grouped.item()

    # Calculate values
    if (is_grouped):
        realism_means = [ratings_grouped[(ratings_grouped.made_by == "human") &␣
↪(ratings_grouped.image_id == image_id)].realism.mean(),
                         ratings_grouped[(ratings_grouped.made_by ==␣
↪"computer") & (ratings_grouped.image_id == image_id)].realism.mean(),
                         ratings_grouped[(ratings_grouped.made_by == "known") &␣
↪(ratings_grouped.image_id == image_id)].realism.mean()]

        creativity_means = [ratings_grouped[(ratings_grouped.made_by ==␣
↪"human") & (ratings_grouped.image_id == image_id)].creative.mean(),
                            ratings_grouped[(ratings_grouped.made_by ==␣
↪"computer") & (ratings_grouped.image_id == image_id)].creative.mean(),
                            ratings_grouped[(ratings_grouped.made_by ==␣
↪"known") & (ratings_grouped.image_id == image_id)].creative.mean()]

        realism_erros = [ratings_grouped[(ratings_grouped.made_by == "human") &␣
↪(ratings_grouped.image_id == image_id)].realism.std(),
```

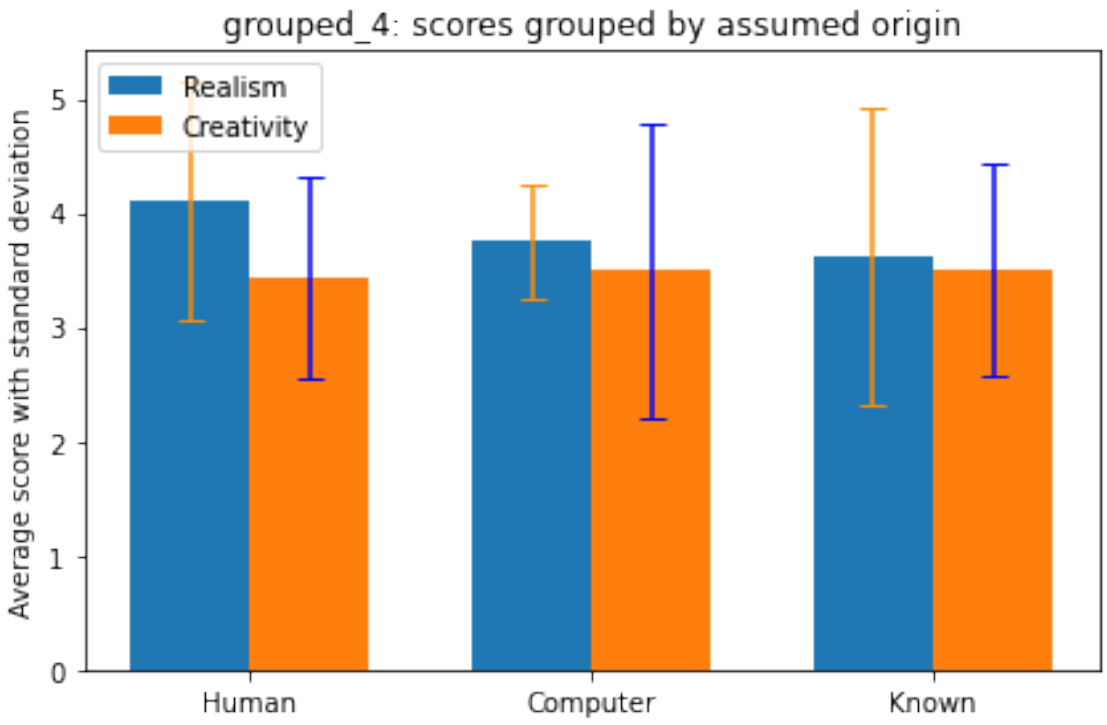
```
                        ratings_grouped[(ratings_grouped.made_by ==␣
↪"computer") & (ratings_grouped.image_id == image_id)].realism.std(),
                        ratings_grouped[(ratings_grouped.made_by == "known") &␣
↪(ratings_grouped.image_id == image_id)].realism.std()]

        creativity_errors = [ratings_grouped[(ratings_grouped.made_by ==␣
↪"human") & (ratings_grouped.image_id == image_id)].creative.std(),
                             ratings_grouped[(ratings_grouped.made_by ==␣
↪"computer") & (ratings_grouped.image_id == image_id)].creative.std(),
                             ratings_grouped[(ratings_grouped.made_by ==␣
↪"known") & (ratings_grouped.image_id == image_id)].creative.std()]
    else:
        realism_means = [ratings_single[(ratings_single.made_by == "human") &␣
↪(ratings_single.image_id == image_id)].realism.mean(),
                        ratings_single[(ratings_single.made_by == "computer")␣
↪& (ratings_single.image_id == image_id)].realism.mean(),
                        ratings_single[(ratings_single.made_by == "known") &␣
↪(ratings_single.image_id == image_id)].realism.mean()]

        creativity_means = [ratings_single[(ratings_single.made_by == "human")␣
↪& (ratings_single.image_id == image_id)].creative.mean(),
                            ratings_single[(ratings_single.made_by ==␣
↪"computer") & (ratings_single.image_id == image_id)].creative.mean(),
                            ratings_single[(ratings_single.made_by == "known")␣
↪& (ratings_single.image_id == image_id)].creative.mean()]

        realism_erros = [ratings_single[(ratings_single.made_by == "human") &␣
↪(ratings_single.image_id == image_id)].realism.std(),
                        ratings_single[(ratings_single.made_by == "computer")␣
↪& (ratings_single.image_id == image_id)].realism.std(),
                        ratings_single[(ratings_single.made_by == "known") &␣
↪(ratings_single.image_id == image_id)].realism.std()]

        creativity_errors = [ratings_single[(ratings_single.made_by == "human")␣
↪& (ratings_single.image_id == image_id)].creative.std(),
                             ratings_single[(ratings_single.made_by ==␣
↪"computer") & (ratings_single.image_id == image_id)].creative.std(),
                             ratings_single[(ratings_single.made_by == "known")␣
↪& (ratings_single.image_id == image_id)].creative.std()]

    # Plot based on: https://matplotlib.org/stable/gallery/
↪lines_bars_and_markers/barchart.html
    labels = ['Human', 'Computer', 'Known']

    x = np.arange(len(labels))  # the label locations
    width = 0.35  # the width of the bars
```

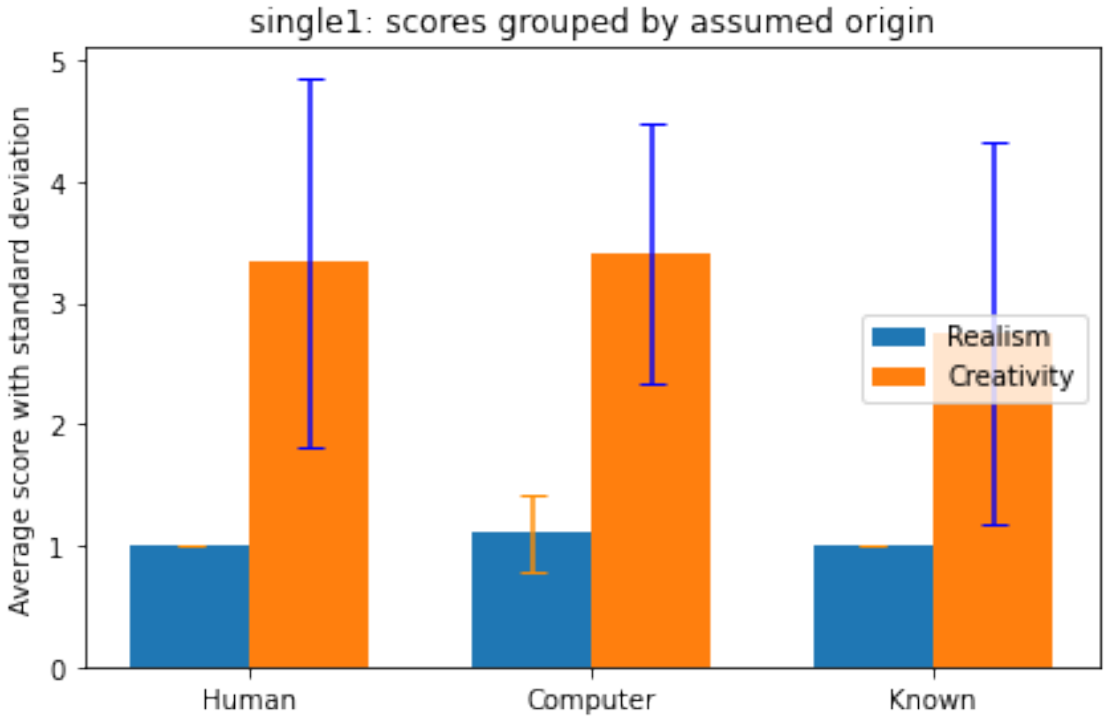
```
    fig, ax = plt.subplots()
    rects1 = ax.bar(x - width/2, realism_means, width, label='Realism',␣
↪yerr=realism_erros, align='center', ecolor='darkorange', capsize=5)
    rects2 = ax.bar(x + width/2, creativity_means, width, label='Creativity',␣
↪yerr=creativity_errors, align='center', ecolor='blue', capsize=5)

    # Add some text for labels, title and custom x-axis tick labels, etc.
    ax.set_ylabel('Average score with standard deviation')
    ax.set_title(image_name + ': scores grouped by assumed origin')
    ax.set_xticks(x)
    ax.set_xticklabels(labels)
    ax.legend()

    fig.tight_layout()

    plt.savefig("Graphs/Bias/Analysis/" + image_name, dpi=300,␣
↪bbox_inches='tight')
    plt.show()
```

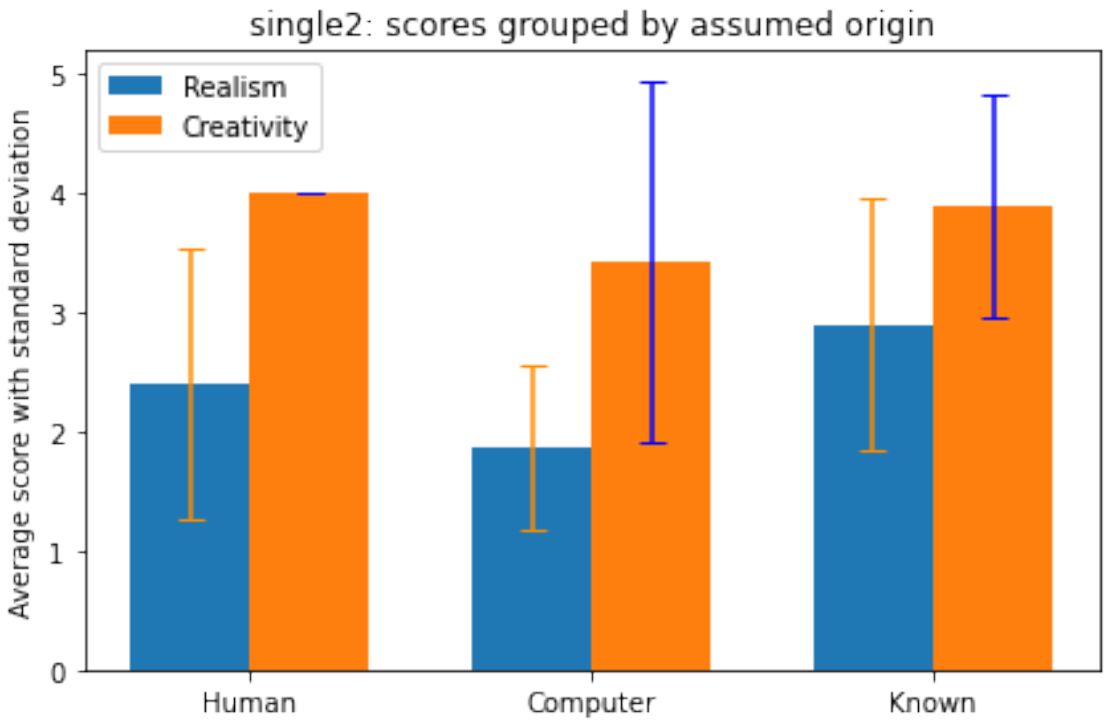


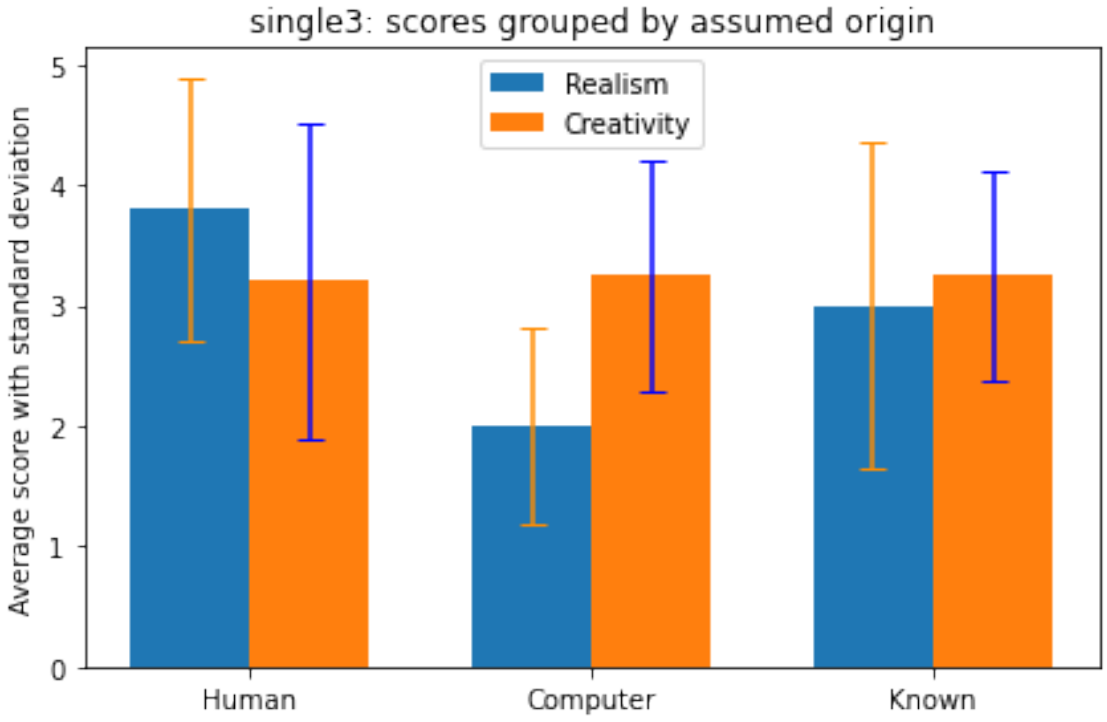

grouped_2: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

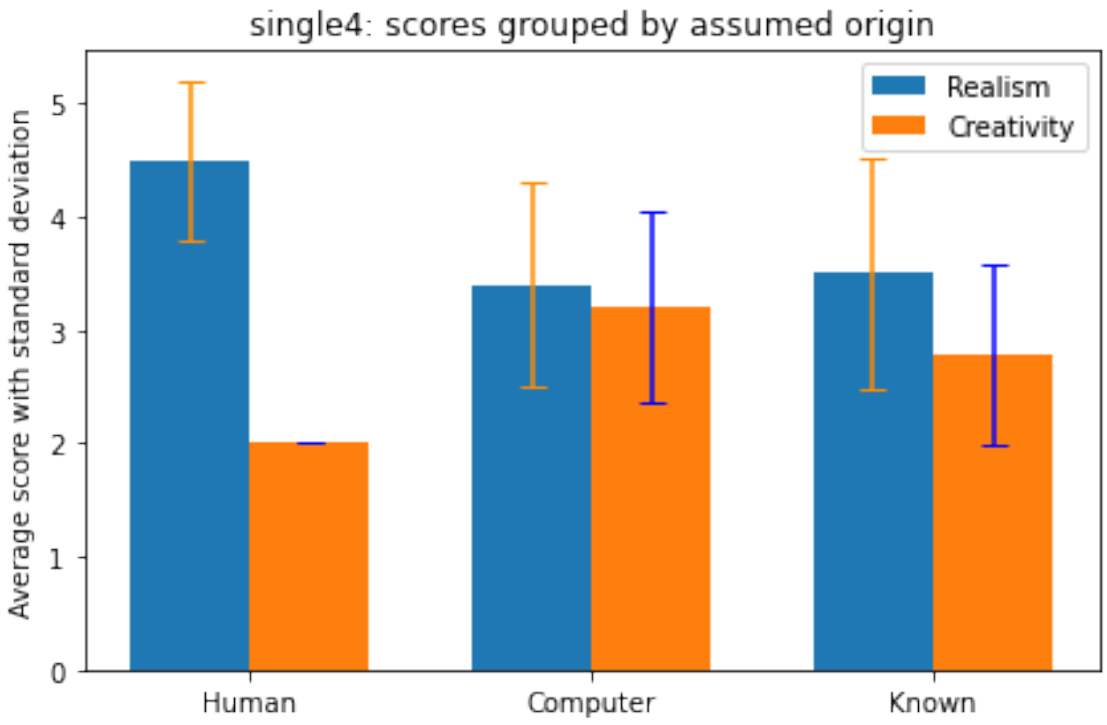

grouped_3: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

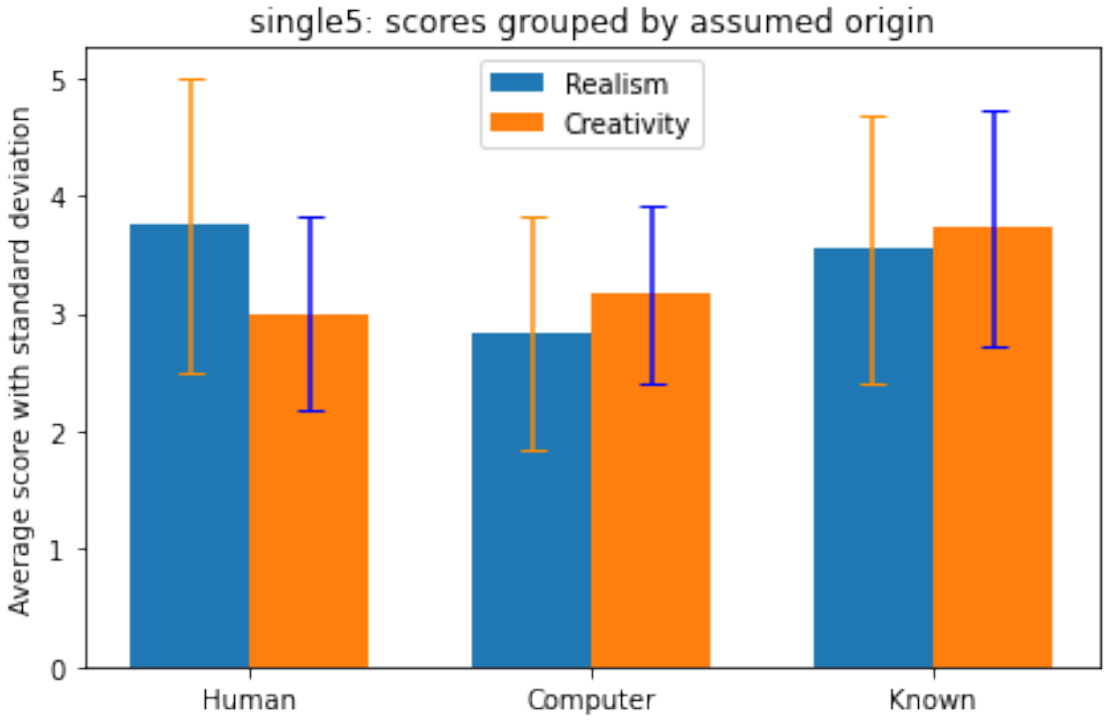

grouped_4: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

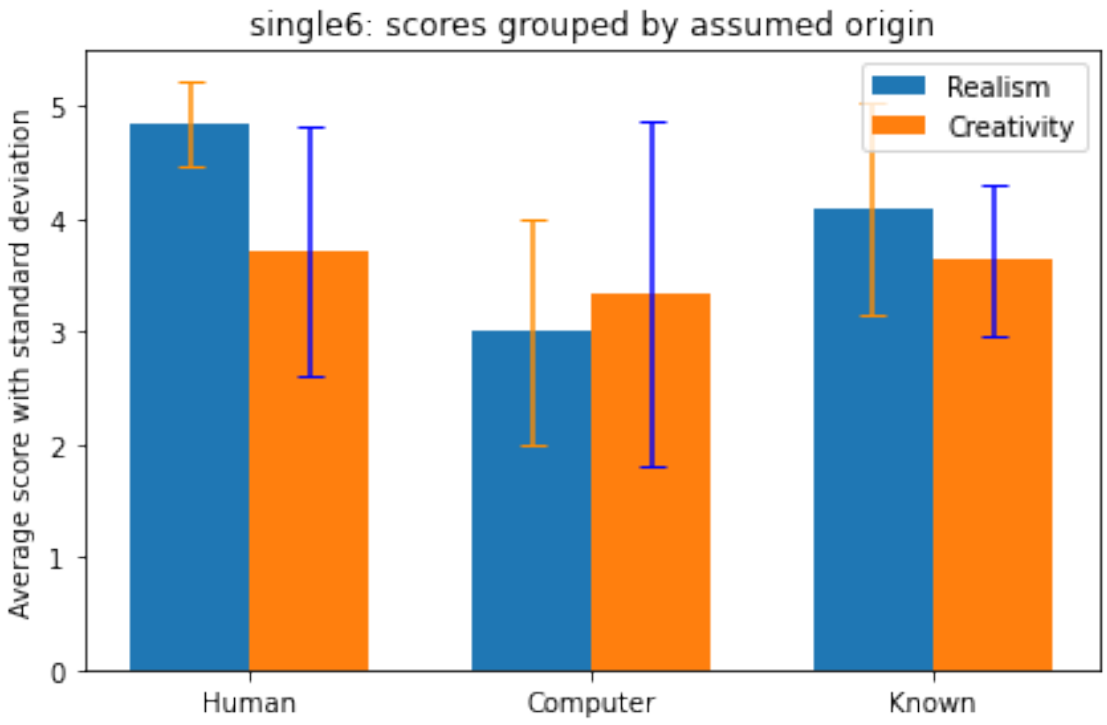

single1: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

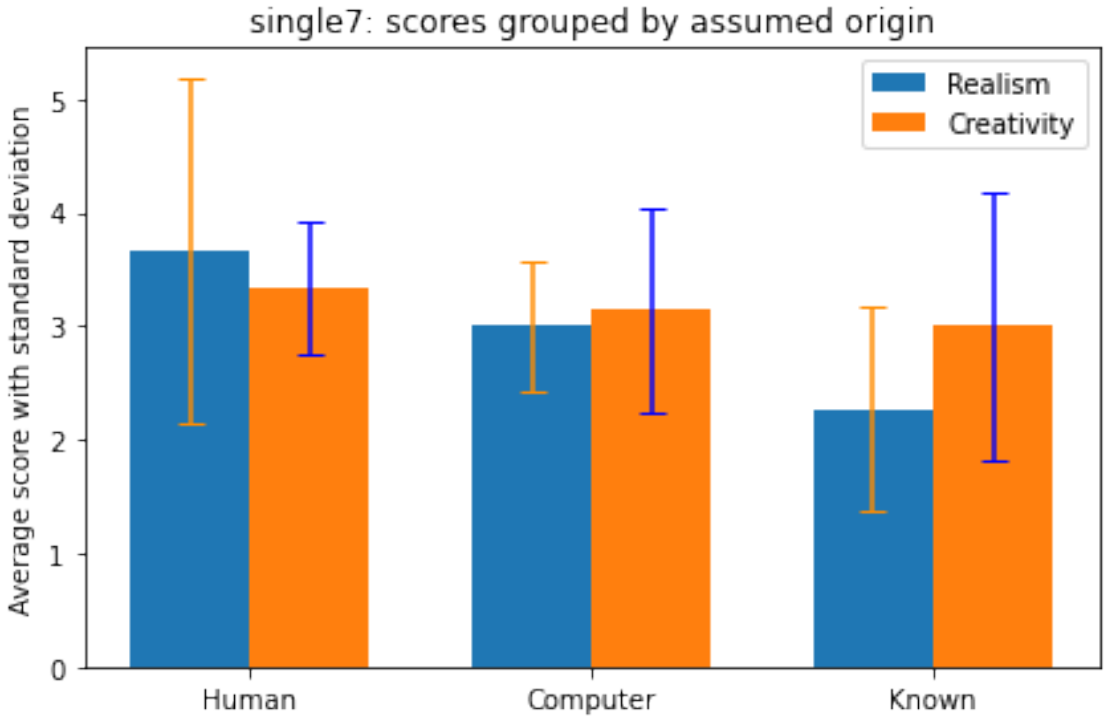

single2: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

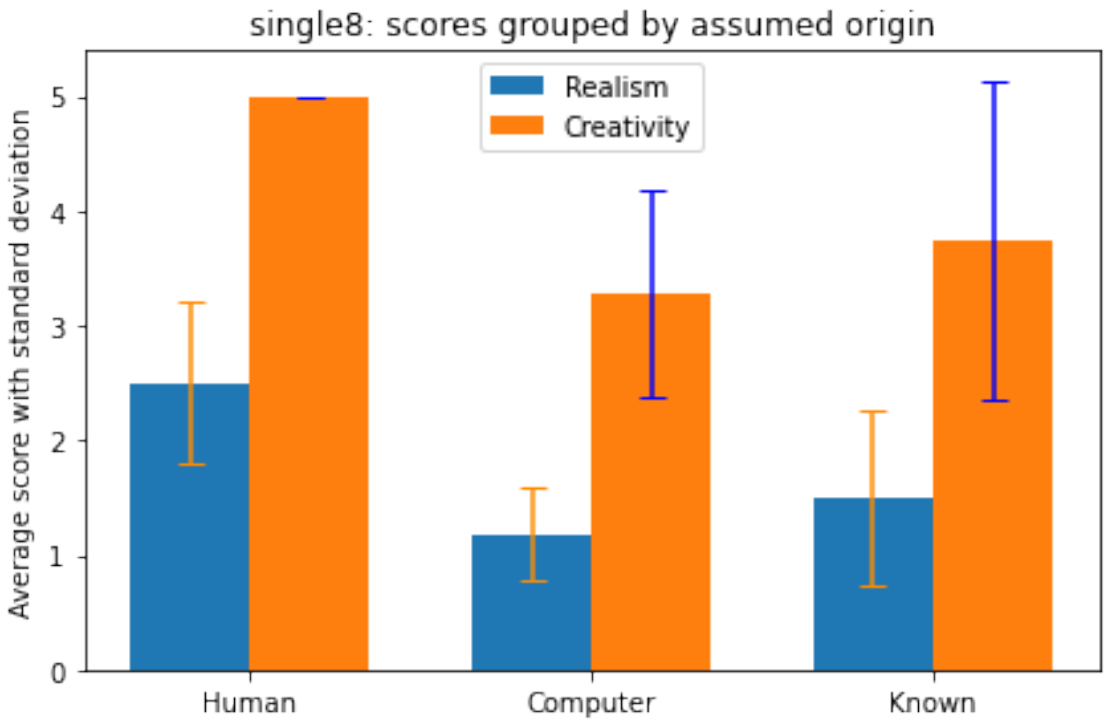

single3: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known


single4: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known


single5: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known


single6: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known

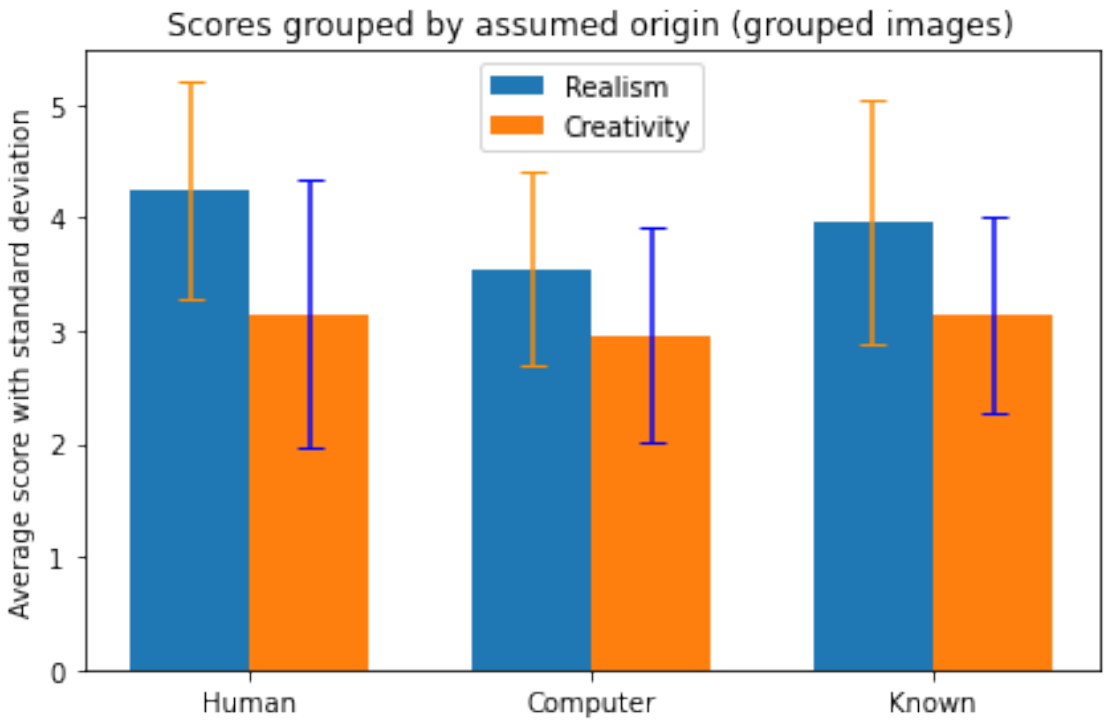

single7: scores grouped by assumed origin
Realism
Creativity
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known



We now also make a general plot for this, looking at mean and std of all grouped images.

[11]:
```
realism_means = [ratings_grouped[(ratings_grouped.made_by == "human")].realism.
 ↪mean(),
                 ratings_grouped[(ratings_grouped.made_by == "computer")].
 ↪realism.mean(),
                 ratings_grouped[(ratings_grouped.made_by == "known")].realism.
 ↪mean()]

creativity_means = [ratings_grouped[(ratings_grouped.made_by == "human")].
 ↪creative.mean(),
                    ratings_grouped[(ratings_grouped.made_by == "computer")].
 ↪creative.mean(),
                    ratings_grouped[(ratings_grouped.made_by == "known")].
 ↪creative.mean()]

realism_erros = [ratings_grouped[(ratings_grouped.made_by == "human")].realism.
 ↪std(),
                 ratings_grouped[(ratings_grouped.made_by == "computer")].
 ↪realism.std(),
                 ratings_grouped[(ratings_grouped.made_by == "known")].realism.
 ↪std()]
```

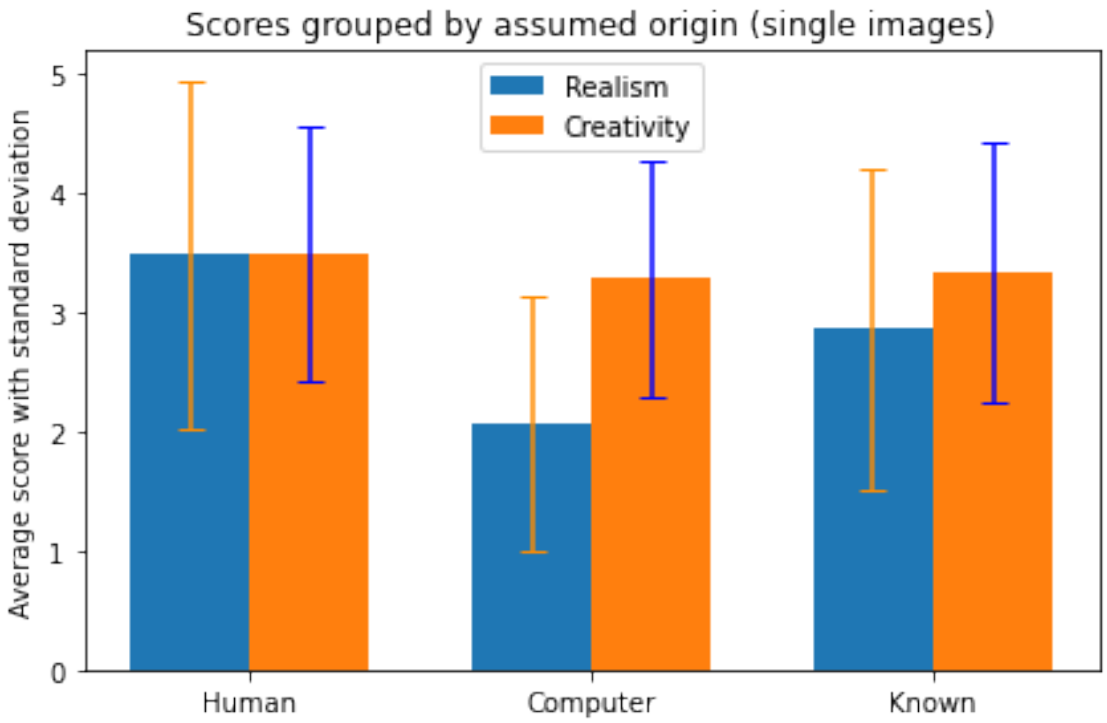
```
creativity_errors = [ratings_grouped[(ratings_grouped.made_by == "human")].
 ↪creative.std(),
                     ratings_grouped[(ratings_grouped.made_by == "computer")].
 ↪creative.std(),
                     ratings_grouped[(ratings_grouped.made_by == "known")].
 ↪creative.std()]

# Plot based on: https://matplotlib.org/stable/gallery/lines_bars_and_markers/
 ↪barchart.html
labels = ['Human', 'Computer', 'Known']

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, realism_means, width, label='Realism',␣
 ↪yerr=realism_erros, align='center', ecolor='darkorange', capsize=5)
rects2 = ax.bar(x + width/2, creativity_means, width, label='Creativity',␣
 ↪yerr=creativity_errors, align='center', ecolor='blue', capsize=5)


# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average score with standard deviation')
ax.set_title('Scores grouped by assumed origin (grouped images)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

fig.tight_layout()

plt.savefig("Graphs/Bias/grouped_images_score_bias", dpi=300,␣
 ↪bbox_inches='tight')
plt.show()
```



We now do the same for the single images

[12]:
```
realism_means = [ratings_single[(ratings_single.made_by == "human")].realism.
 ↪mean(),
                 ratings_single[(ratings_single.made_by == "computer")].realism.
 ↪mean(),
                 ratings_single[(ratings_single.made_by == "known")].realism.
 ↪mean()]

creativity_means = [ratings_single[(ratings_single.made_by == "human")].
 ↪creative.mean(),
                    ratings_single[(ratings_single.made_by == "computer")].
 ↪creative.mean(),
                    ratings_single[(ratings_single.made_by == "known")].
 ↪creative.mean()]

realism_erros = [ratings_single[(ratings_single.made_by == "human")].realism.
 ↪std(),
                 ratings_single[(ratings_single.made_by == "computer")].realism.
 ↪std(),
                 ratings_single[(ratings_single.made_by == "known")].realism.
 ↪std()]
```

```
creativity_errors = [ratings_single[(ratings_single.made_by == "human")].
 ↪creative.std(),
                     ratings_single[(ratings_single.made_by == "computer")].
 ↪creative.std(),
                     ratings_single[(ratings_single.made_by == "known")].
 ↪creative.std()]

# Plot based on: https://matplotlib.org/stable/gallery/lines_bars_and_markers/
 ↪barchart.html
labels = ['Human', 'Computer', 'Known']

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, realism_means, width, label='Realism',␣
 ↪yerr=realism_erros, align='center', ecolor='darkorange', capsize=5)
rects2 = ax.bar(x + width/2, creativity_means, width, label='Creativity',␣
 ↪yerr=creativity_errors, align='center', ecolor='blue', capsize=5)


# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Average score with standard deviation')
ax.set_title('Scores grouped by assumed origin (single images)')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

fig.tight_layout()

plt.savefig("Graphs/Bias/single_images_score_bias", dpi=300,␣
 ↪bbox_inches='tight')
plt.show()
```



1.8 Analysing notes

The following code blocks will give the notes left by participant as well as the image they belong to.

[13]:
```
# Get all images
image_ids = images.image_id

for image_id in image_ids:
    # Get image
    image = images[images.image_id == image_id]

    # Determine grouped
    is_grouped = image.grouped.item()

    if is_grouped:
        notes = ratings_grouped[(ratings_grouped.image_id == image_id)].note.
↪dropna()
    else:
        notes = ratings_single[(ratings_single.image_id == image_id)].note.
↪dropna()

    print(f" ----------Showing notes for {image.filename.item()} ----------")
```

```
    for note in notes:
        print(note)
        print()
```

```
 ----------Showing notes for grouped_1 ----------
Just turns red but cool it is the car only

Its just a car that turns red…

3 of the 4 cars have their color changed thats why I rated it lower on the
creativity scale

If it is an AI, it fooled me well.

Auto wordt rood. Goed dat het enkel de wagen is en niet de volledige foto dat
rood wordt maar dat is het ook.

 ----------Showing notes for grouped_2 ----------
Nose changes i think

Dont really see a lot of difference

looks like a car you could buy, so not very creative in my opinion

Van boxy design naar platter design; indrukwekkend hoe dit veranderd. Eerste
lijkt op nissan gtr r32, derde heeft duidelijk bmw trekken maar lichten zijn
absoluut niet bmw en eerder subaru

 ----------Showing notes for grouped_3 ----------
Nice change of wheels

Wheels changed, weird color on left one

Different wheels, weird color on left one

Here a lot more has changed so I think it has been more creative

velgen veranderen tussen foto maar kleur soms ook, laatste doet vermoeden dat
het computer is. Wel indrukwekkend als het computer is, maar niet zo creatief
aangezien het maar gewoon velgen zijn.

 ----------Showing notes for grouped_4 ----------
Looks like completely different cars to me wouldnt see correspondence if not
underneath eachother

Goes from a boring family car to a sporty one. cool stuff, especially like the
hood, interesting a computer can do this.
```

Looks more sporty, cars are verry different but color and wheels is prtty consistent

If it is an AI, it fooled me well. The cars are lowered, and changed wheels and bumpers.

de autos zijn duidelijk op elkaar gebasseerd maar worden verlaagd en sportiever gemaakt, heb het gevoel dat er soms verschillende voorlichten zijn links en rechts.

----------Showing notes for single1 ----------
Unique but not a car

Looks nothing like a car but I quite like it. It looks like a albums cover of some horror music.

lol looks like a horror movie or something

Dont really know what happened here

looks like an album cover but not like a car

It is detailed and creative but doesnt look like a car. It looks like an abstract art piece.

Im incredibly confused as to what portion of the picture is supposed to be a car

Poster voor een horror film Met een beetje fantasie zie ik in een auto in de witte vlek…

----------Showing notes for single2 ----------
Hmmm not sure if that is two fronts haha

Audi looks. Lots of detail for a car that has two fronts lol

2 fronts

it looks out of proportion, I find it ugly, but unexpected in a way so creative

Ziet er uit als een creatie dat door top gear of dergelijke gemaakt is. Lijkt 2 voorkanten te hebben door naar de spiegels te kijken echter de bumper en lichten doen een achterkant vermoeden. Best interessant.

----------Showing notes for single3 ----------
Lol, an old ford pick up truck cabriolet thing. Weird. like it tho something odd with the doors

some weird stuff in door, guess from stretching

Looks like the car is missing some parts

Wow.

Een pickup cabrio, grappig concept. Front heeft een rare bumper guard langs 1 zeide maar niet de andere zeide, deuren en lade van truck lijken slecht aan elkaar geplakt en er is een rare plek op de achterkant. Vermoedelijk pc maar wel intreressant idee.

----------Showing notes for single4 ----------
Something weird with the mirrors, looks like a mercedes SUV front with classic BMW headlights

Think it is just a car with a mercedes badge and bmw lights

The car is impressive for both human and computer

Mercedes SUV met BMW angle eyes

----------Showing notes for single5 ----------
cool text, meh car

The car is not creative but the text is cool. not easy to determine front and one of the doors is actually inside the body

the text () intrugues me

Looks like the cars front and back are both the back of a car

Heeft iets weg van de Bentley Flying Spur. Voorkant lijkt terug op achterkant als je lichten bestudeerd.

----------Showing notes for single6 ----------
Quite like this image, feels like a car brochure, something odd with the reflection and no clue what the fog in the background is but overall really like this

Looks like an audi with bentley merge haha

Amazing design. Hard to believe this could be an AI.

Leuk design, Iets in mij zegt Audi maar geeen idee welk model het juist zou zijn. Passa koplamp is raar

----------Showing notes for single7 ----------
Looks realistic, but also not at all (if that makes sense). Reminds me of those Alpinas

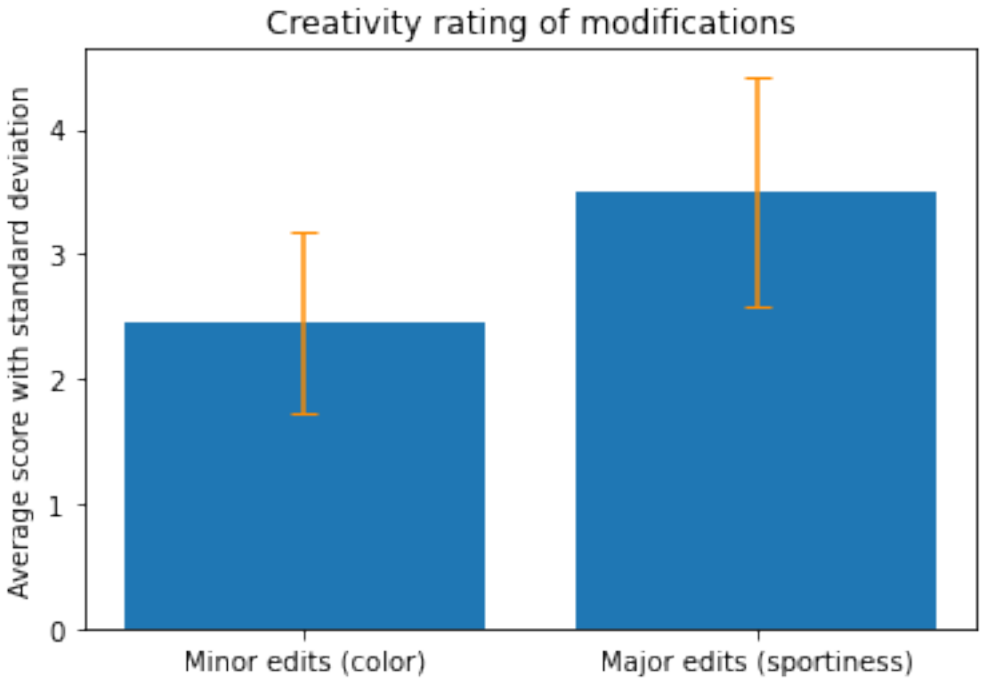
```
Aston looks pretty bad

Definitely human level design. Well done if this is an AI.

Lijkt op een aston martin met alpine lichten op geplakt. Vrij slechte kwalieteit
maar wel interesant, rare scope in motorkap.

 ----------Showing notes for single8 ----------
this one is quite cool and reminds me of a mini

I actually quite like this image. If you look at it quickly it doesnt look as if
anything is wrong yet so much is. Its really quite interesting.

Think it is some hyundai perhaps, looks funny

It (human or computer) kind of screwed up the car. I would say its creative but
it isnt really a car.

it looks like another car drove into one side of another car

This is funny. Quite obviously a car but it seems confused about angles. I would
guess this is an AI.

Grappig, op een snelle glans realistisch maar dat is het uiteraard niet. Heeft
iets weg van Kia Rio
```

1.9 Creativity and minor changes

The following code block will compare the recieved creativity score for a small and major edits.

[14]:
```
# Get image ids
image_id_minor = images[images.filename == "grouped_1"].image_id.item()
image_id_major = images[images.filename == "grouped_4"].image_id.item()

# Calculate values
means = [ratings_grouped[(ratings_grouped.made_by == "known") &␣
 ↪(ratings_grouped.image_id == image_id_minor)].creative.mean(),
         ratings_grouped[(ratings_grouped.made_by == "known") &␣
 ↪(ratings_grouped.image_id == image_id_major)].creative.mean()]

errors = [ratings_grouped[(ratings_grouped.made_by == "known") &␣
 ↪(ratings_grouped.image_id == image_id_minor)].creative.std(),
          ratings_grouped[(ratings_grouped.made_by == "known") &␣
 ↪(ratings_grouped.image_id == image_id_major)].creative.std()]
```

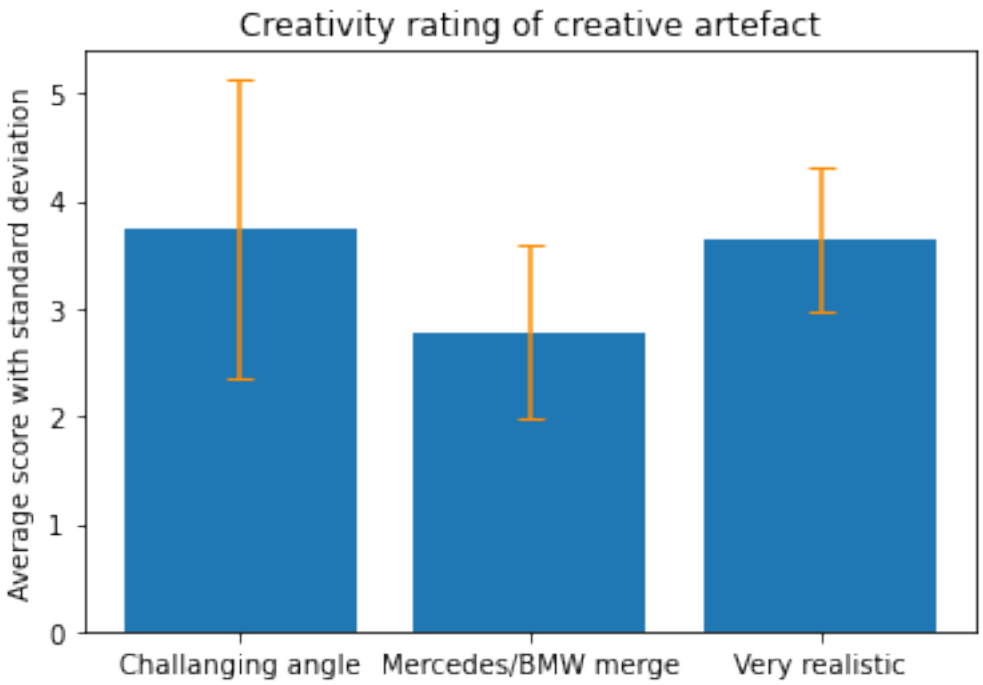
```
x_ticks = ["Minor edits (color)", "Major edits (sportiness)"]

# Replace NaN with 0 - occurs when certain rating doesn't occur
means = pd.Series(means).fillna(0).tolist()
errors = pd.Series(errors).fillna(0).tolist()

x = np.arange(len(means))
y = means

# plot a histogram with standard deviation
plt.title("Creativity rating of modifications")
plt.ylabel('Average score with standard deviation')
plt.xticks(x, x_ticks)

plt.bar(x, y, yerr=errors, align='center', ecolor='darkorange', capsize=5)

plt.savefig("Graphs/Creativity/creativity_minor_changes", dpi=300,␣
 ↪bbox_inches='tight')
plt.show()
```



1.10 Creativity and single images

The following code block will compare the recieved creativity score for an artefact, basic design and hyper realistic design.

[15]:
```
# Get image id
image_artefact_id = images[images.filename == "single8"].image_id.item()
image_merge_id = images[images.filename == "single4"].image_id.item()
image_hyperrealistic_id = images[images.filename == "single6"].image_id.item()

# Calculate values
means = [ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_artefact_id)].creative.mean(),
         ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_merge_id)].creative.mean(),
         ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_hyperrealistic_id)].creative.mean()]

errors = [ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_artefact_id)].creative.std(),
          ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_merge_id)].creative.std(),
          ratings_single[(ratings_single.made_by == "known") & (ratings_single.
 ↪image_id == image_hyperrealistic_id)].creative.std()]

x_ticks = ["Challanging angle", "Mercedes/BMW merge", "Very realistic"]

# Replace NaN with 0 - occurs when certain rating doesn't occur
means = pd.Series(means).fillna(0).tolist()
errors = pd.Series(errors).fillna(0).tolist()

x = np.arange(len(means))
y = means

# plot a histogram with standard deviation
plt.title("Creativity rating of creative artefact")
plt.ylabel('Average score with standard deviation')
plt.xticks(x, x_ticks)

plt.bar(x, y, yerr=errors, align='center', ecolor='darkorange', capsize=5)

plt.savefig("Graphs/Creativity/single_images_creativity", dpi=300,␣
 ↪bbox_inches='tight')
plt.show()
```

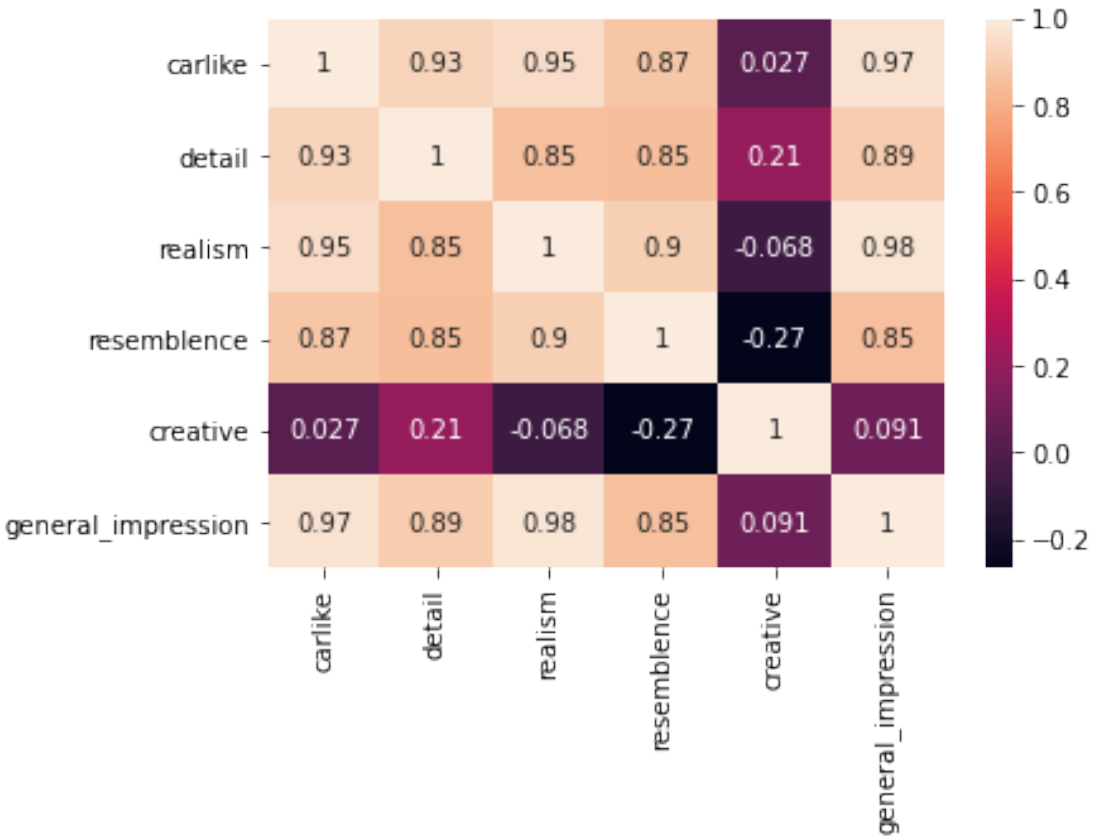


1.11 Correlation between criteria

1.11.1 Single images

The following code block will anayse the correlation between criteria of the single images

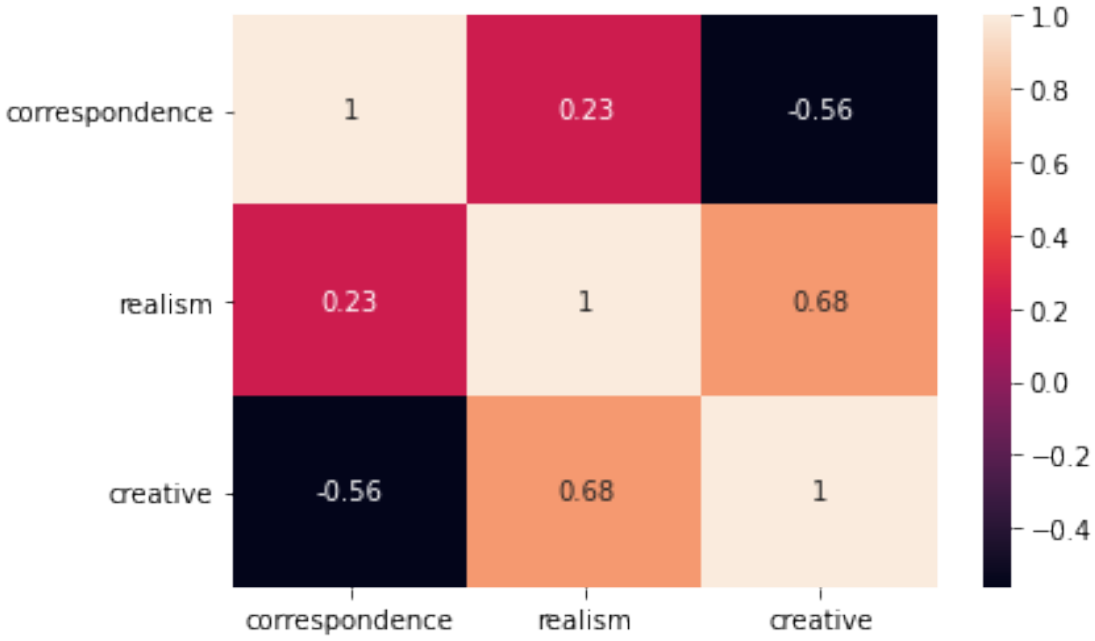
[16]:
```
ratings_single_averaged = ratings_single.drop(['participant_id', 'note'],␣
↪axis=1)
ratings_single_averaged = ratings_single_averaged.groupby(['image_id']).mean()

sn.heatmap(ratings_single_averaged.corr(), annot=True)
plt.savefig("Graphs/Correlation/single_images", dpi=300, bbox_inches='tight')
plt.show()
```



1.11.2 Grouped images

The following code block will do the same for correlation between criteria of grouped images

[17]:
```
ratings_grouped_averaged = ratings_grouped.drop(['participant_id', 'note'],␣
↪axis=1)
ratings_grouped_averaged = ratings_grouped_averaged.groupby(['image_id']).mean()

sn.heatmap(ratings_grouped_averaged.corr(), annot=True)
plt.savefig("Graphs/Correlation/grouped_images", dpi=300, bbox_inches='tight')
plt.show()
```



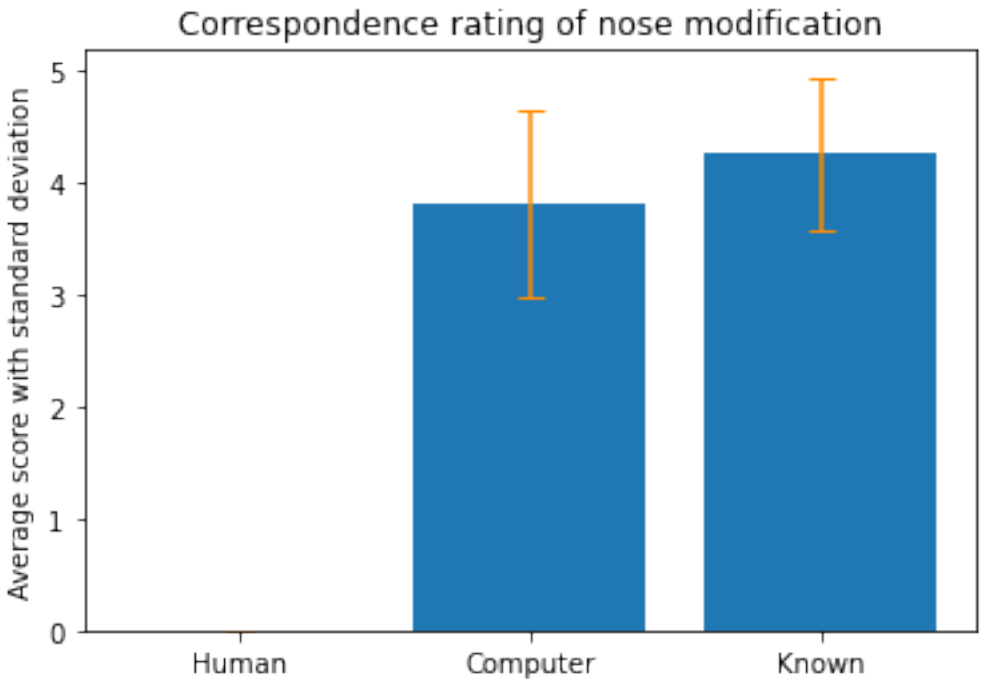
1.12 Grouped ratings analysis

The following code blocks will give some analytics for the grouped ratings.

1.12.1 Correspondence rating of rim design modification

[18]:
```
# Get image id
image_id = images[images.filename == "grouped_3"].image_id.item()

# Calculate values
means = [ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "human")].correspondence.mean(),
         ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "computer")].correspondence.mean(),
         ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "known")].correspondence.mean()]

errors = [ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "human")].correspondence.std(),
          ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "computer")].correspondence.std(),
          ratings_grouped[(ratings_grouped.image_id == image_id) &␣
 ↪(ratings_grouped.made_by == "known")].correspondence.std()]

x_ticks = ["Human", "Computer", "Known"]
```

```
# Replace NaN with 0 - occurs when certain rating doesn't occur
means = pd.Series(means).fillna(0).tolist()
errors = pd.Series(errors).fillna(0).tolist()

x = np.arange(len(means))
y = means

# plot a histogram with standard deviation
plt.title("Correspondence rating of nose modification")
plt.ylabel('Average score with standard deviation')
plt.xticks(x, x_ticks)

plt.bar(x, y, yerr=errors, align='center', ecolor='darkorange', capsize=5)

plt.savefig("Graphs/Grouped/rim_design_correspondence", dpi=300,␣
↪bbox_inches='tight')
plt.show()
```

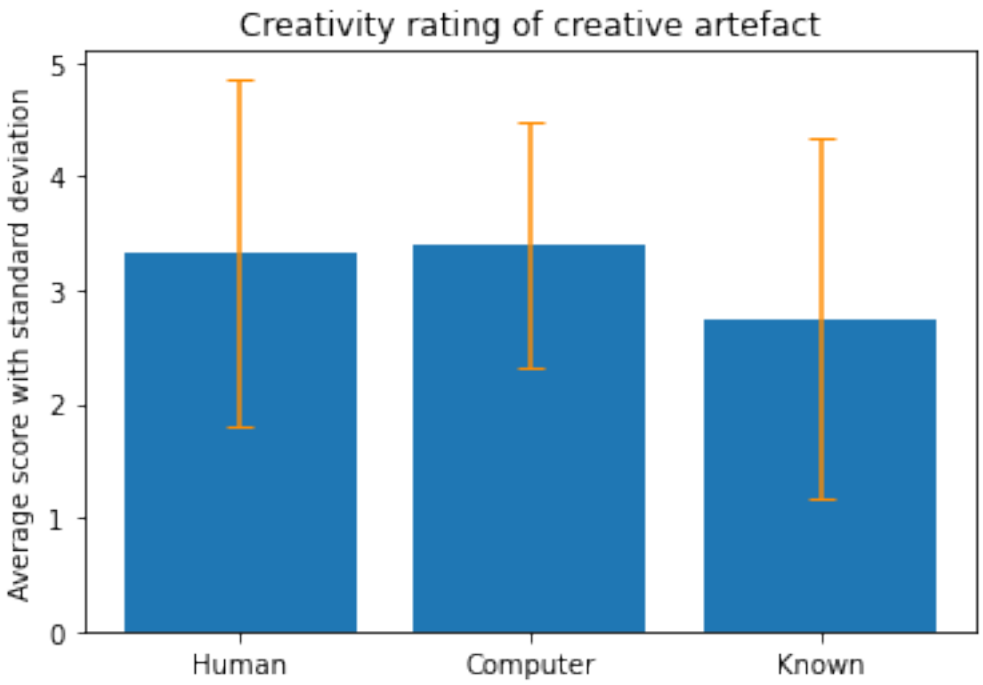


1.13 Single ratings analysis

The following code blocks will give some analytics for the single ratings.

1.13.1 Creativity of album art cover

[19]:
```
# Get image id
image_id = images[images.filename == "single1"].image_id.item()

# Calculate values
means = [ratings_single[(ratings_single.image_id == image_id) & (ratings_single.
 ↪made_by == "human")].creative.mean(),
         ratings_single[(ratings_single.image_id == image_id) & (ratings_single.
 ↪made_by == "computer")].creative.mean(),
         ratings_single[(ratings_single.image_id == image_id) & (ratings_single.
 ↪made_by == "known")].creative.mean()]

errors = [ratings_single[(ratings_single.image_id == image_id) &␣
 ↪(ratings_single.made_by == "human")].creative.std(),
          ratings_single[(ratings_single.image_id == image_id) &␣
 ↪(ratings_single.made_by == "computer")].creative.std(),
          ratings_single[(ratings_single.image_id == image_id) &␣
 ↪(ratings_single.made_by == "known")].creative.std()]

x_ticks = ["Human", "Computer", "Known"]

# Replace NaN with 0 - occurs when certain rating doesn't occur
means = pd.Series(means).fillna(0).tolist()
errors = pd.Series(errors).fillna(0).tolist()

x = np.arange(len(means))
y = means

# plot a histogram with standard deviation
plt.title("Creativity rating of creative artefact")
plt.ylabel('Average score with standard deviation')
plt.xticks(x, x_ticks)

plt.bar(x, y, yerr=errors, align='center', ecolor='darkorange', capsize=5)

plt.savefig("Graphs/Single/album_art_cover_creative", dpi=300,␣
 ↪bbox_inches='tight')
plt.show()
```


Creativity rating of creative artefact
Average score with standard deviation
0
1
2
3
4
5
Human
Computer
Known