Experiment-Analysis

May 25, 2022

1 Experiment Analysis

This notebook loads the (anonymized) human data from the behavioral experiment and analyzes it.

[1]:
```
import copy
import hashlib
import numpy as np
import pandas as pd
import json
import seaborn as sns
import matplotlib.pylab as plt

%load_ext autoreload
%autoreload 2
```

1.1 Load data and unpack conditions, etc

[2]:
```
trials = pd.read_csv("final_exp_human_trials.csv")
questions = pd.read_csv("final_human_exp_questions.csv")
```

[3]:
```
# Sanity check randomizations

if False:
    print("\nLog message counts:")
    print(questions.key.value_counts())

    print("\nFull condition messages:")
    print(questions[questions.key == "condition"].value.value_counts())

    print("\nTrial data files used:")
    print(questions[questions.key == "trialDataFile"].value.value_counts())

    print("\nFeature randomizations:")
    feature_randomizations = questions[questions.key == "featureRandomization"].
 ↪value
```

```
    feature_randomizations = feature_randomizations.apply(lambda x: json.
↪loads(x))
    feature_randomizations = pd.DataFrame(list(feature_randomizations.values))
    for feature in ["Green", "Red", "Blue", "Spotted", "Solid", "Striped"]:
        print(feature_randomizations[feature].value_counts())
```

1.2 Read in trial data

[4]:
```
# Load data, filter out debug, hash ID

# Read JSON data into Python objects
trials["data"] = trials.value.apply(lambda x: json.loads(x))

for field in ["rt", "trial_type"]:
    trials[field] = trials.data.apply(lambda x: x.get(field))

trials.trial_type.value_counts()
```

[4]:
```
survey-html-form        3780
html-button-response    1143
survey-likert            607
instructions             231
survey-multi-choice      186
survey-text              105
Name: trial_type, dtype: int64
```

[5]:
```
print("Pre-Quiz Count: {} participants.".format(trials.workerid.nunique()))
```

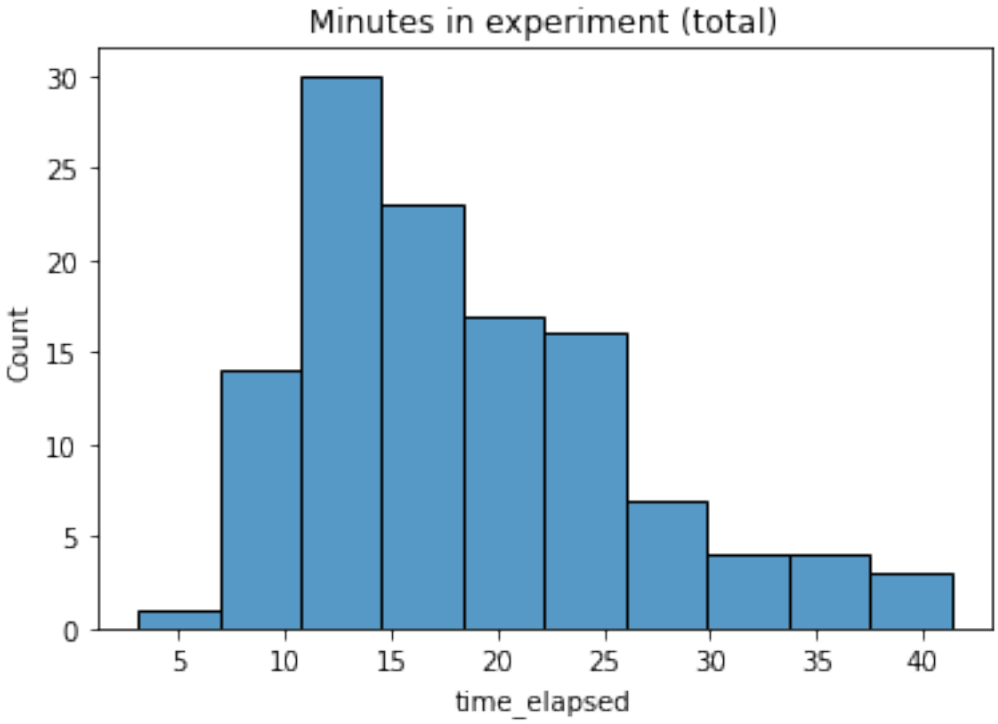
```
Pre-Quiz Count: 119 participants.
```

2 Experiment Duration

[6]:
```
trials["time_elapsed"] = trials.data.apply(lambda x: x.get("time_elapsed"))
minutes_in_experiment = trials.groupby("workerid").time_elapsed.max() / 1000 /␣
↪60

sns.histplot(minutes_in_experiment, bins=10)
plt.title("Minutes in experiment (total)")

minutes_in_experiment.describe()
```

[6]:
```
count    119.000000
mean      18.525471
std        7.624343
min        3.082400
25%       12.996092
```

```
50%        17.425250
75%        22.826625
max        41.320933
Name: time_elapsed, dtype: float64
```



2.1 Quiz

How many attempts do people take? What fraction pass? Which pages are "hardest"?

[7]:
```
quizResults = questions[questions.key.str.contains('quizPassed') | questions.
 ↪key.str.contains('quizFailed')]
quizResults["passed_quiz"] = quizResults.key.apply(lambda x: True if x ==␣
 ↪"quizPassed" else False)

quizResults = quizResults[["workerid", "passed_quiz"]].set_index("workerid")

quizResults.passed_quiz.value_counts()
```

```
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/1755126429.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quizResults["passed_quiz"] = quizResults.key.apply(lambda x: True if x ==
"quizPassed" else False)
```

[7]:
```
True     104
False     14
Name: passed_quiz, dtype: int64
```

[8]:
```
quizAttempts = questions[questions.key.str.contains('quizAttempt')]
quizAttempts = quizAttempts[quizAttempts.workerid.isin(trials.workerid.
 ↪unique())]

quizAttempts["data"] = quizAttempts.value.apply(lambda x: json.loads(x))
quizAttempts["correct"] = quizAttempts["data"].apply(lambda x: x["correct"])
quizAttempts["total"] = quizAttempts["data"].apply(lambda x: x["total"])
quizAttempts["cum_correct"] = quizAttempts["correct"] + 14 -␣
 ↪quizAttempts["total"]
quizAttempts["passed"] = quizAttempts["data"].apply(lambda x: x["total"] ==␣
 ↪x["correct"])

if False:
    quizAttempts.groupby("key").cum_correct.agg(["size", "mean", "std"])
```

[9]:
```
participant_quiz_results = quizAttempts.groupby("workerid").size()

trials = trials.set_index("workerid").merge(participant_quiz_results.
 ↪to_frame(name="quiz_attempts"), left_index=True, right_index=True,␣
 ↪how="left").reset_index()
```

[10]:
```
quiz_trials = trials[trials.data.apply(lambda x: x.get("total") is not None)]

quiz_trials["total"] = quiz_trials.data.apply(lambda x: int(x.get("total")))
quiz_trials["correct"] = quiz_trials.data.apply(lambda x: int(x.get("correct")))
quiz_trials["wrong"] = quiz_trials.total - quiz_trials.correct

quiz_trials["response"] = quiz_trials.data.apply(lambda x: x.get("response"))

quiz_trials["first_attempt"] = quiz_trials.num <= 12
wrong_answers = quiz_trials[quiz_trials.wrong >=1]
```

```
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3177066139.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  quiz_trials["total"] = quiz_trials.data.apply(lambda x: int(x.get("total")))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3177066139.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quiz_trials["correct"] = quiz_trials.data.apply(lambda x:
int(x.get("correct")))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3177066139.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quiz_trials["wrong"] = quiz_trials.total - quiz_trials.correct
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3177066139.py:7
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quiz_trials["response"] = quiz_trials.data.apply(lambda x: x.get("response"))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3177066139.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quiz_trials["first_attempt"] = quiz_trials.num <= 12
```

2.2 Exit Survey

[11]:
```
exit_survey = trials[trials.trial_type == 'survey-text']
```

[12]:
```
for question in ["experiment_objective", "participant_strategy",␣
↪"experiment_confusing", "misc_feedback"]:
    exit_survey[question] = exit_survey.data.apply(lambda x:␣
↪x["response"][question])

print("Experiment Objective:\n")
print(exit_survey.experiment_objective.values)
```

```
print("\nParticipant Strategies:\n")
print(exit_survey.participant_strategy.values)

print("\nAnything Confusing:\n")
print(exit_survey.experiment_confusing.values)

print("\nMisc Feedback:\n")
print(exit_survey.misc_feedback.values)
```

```
Experiment Objective:

['to ensure tourists chose tasty mushrooms'
 'to provide them with the knowledge of the tatsiest mushrooms'
 'To get them as many tasty mushrooms as possible'
 'For tourists to collect the tastiest mushrooms possible each time.'
 'maximise the score' 'To get them the best'
 'Get them to pick the best mushrooms'
 'TO DIRECT PEOPLE TO THE MOST ENJOYYABLE TO EAT'
 'to help them work out the tasty mushrooms'
 'to help them pick the best mushroom' 'select good tasting mushrooms'
 'To make sure that the tourists picked the highest value mushrooms.'
 'To help them pick good mushrooms.'
 'To get teh highest possible taste in mushrooms for the people'
 'To get the tourists to pick the tastiest mushrooms'
 'Help the tourists chose the tastiest mushrooms'
 'To make help the tourist pick the tastiest mushroom'
 'to inform tourists about mushrooms'
 'give out the best flavours to attract more tourists'
 'To give the customer the best tasting mushroom'
 'To teach them to avoid the bad mushrooms more so'
 'I had to get the tourists to pick tasty mushrooms' 'to make them happy'
 'Help them find the best mushrooms'
 'inform the visitors how to pick the tastiest mushrooms'
 'To get them to pick the tastiest mushrooms.'
 'My objective was to help the tourists pick'
 'To help them pick good mushrooms.'
 'To help people get the tastiest mushrooms'
 'Maximise their tastiness score.' 'help them pick good mushrooms'
 'Help them pick the best tasting mushrooms'
 'to try and teach as much as possible'
 'To help them pick the tastiest mushroom'
 'To help them get the most points/tastiest mushrooms'
 'To make sure tourist pick tasty mushrooms.'
 'To help the tourist avoid the least tasty mushrooms'
 'helping the tourists pick the tastiest mushrooms'
 'to help pick tasty mushrooms' 'maximise tastiness'
 'to help them to pick the most tasty mushroom'
```

```
 'Help tourists to pick tasty mushrooms'
 'To help the tourists pick the tastiest mushrooms.'
 'to give them the tastiest mushrooms'
 'To help tourists pick the best mushrooms' 'asda'
 'to help make them pick the tasty mushrooms'
 'To help the tourists pick good mushrooms. '
 'get them to pick the tastiest mushrooms'
 'To educate tourists on  how to identofy and pick the tastiest mushrooms.'
 'For the people to find the tastiest mushrooms'
 'To guide the tourists to pick the tastiest mushrooms'
 'To ensure customers took the tastiest mushrooms'
 'to help guide the tourist pick the best mushrooms'
 'To help the tourists pick the tastiest mushrooms.'
 'choose the best value' 'Make sure the tourists gets the best mushrooms.'
 'To help the people pick the tastiest mushrooms'
 'To help the tourist find the tastiest mushrooms'
 'To help visitors pick tasty mushrooms'
 'To help visitors pick the best mushrooms'
 'To help tourists find tasty mushrooms in the patches'
 'To get them to learn to pick the best mushrooms'
 'to help them pick good mushrooms'
 'To maximise the number of blue mushrooms the tourist would choose.'
 'To help them gain as many tasty mushrooms as possible'
 'To help them avoid the worst tastes and try to guide them towards tasty
mushrooms'
 'To get the tastiest mushrooms' 'help people pick tasty mushrooms'
 'To guide tourists to picking good mushrooms'
 'to advise the tastiest fungi to eat' 'help visitors pick good mushrooms'
 'To help the customers pick the best mushrooms'
 'To help the tourists choose the most tasty mushroom in the most easiest way
possible'
 'Help tourists pick the good (highest value) mushrooms'
 'To ensure the tourist get good mushroom'
 'To guide them to make the best choices.'
 'to help them pick the tastiest mushrooms' 'add the most value'
 'guide tourists to the tasty mushrooms'
 'to help them pick good mushrooms throughout their visit'
 'To help the tourists pick the mushrooms with the highest points'
 'To make sure they picked the tastiest mushrooms.'
 'to help tourists pick the tasty mushrooms'
 'To help pick the best mushroom '
 'To help the customers decide which mushrooms are tastiest.'
 'To be consistent' 'to get max points' 'to instruct'
 'help them pick the tastiest mushrooms' 'maximize taste = points'
 'To help the tourists to identify tasty mushrooms.'
 'To guide tourists to the best tastiest mushrooms'
 'To help them pick good mushrooms'
 'to help them choose the best mushrooms' 'get the most points'
```

```
 'To help them pick the tastiest mushrooms'
 'I did not give them the chance to really choose by themselves.'
 'to help them pick the tastiest mushrooms'
 'To help them pick the best mushrooms possible'
 'To make them choose good ones'
 'to help people pick or learn about the best mushrooms'
 'To ensure that my guests took the mushrooms with the highest points and
therefore would be the tastiest'
 'To do my best to help the tourists select the tastiest mushroom'
 'To correctly guide them or teach them to pick the tasty mushrooms from each
patch.']

Participant Strategies:

['depending on which option could give them more info on the mushrooms'
 'based on the number of patches they would visit'
 'I looked at what mushrooms they had and gave the best advice '
 'If there was more than one patch they visited - the principle could be applied
to more than one patch.'
 'always tried to teach them, if possible' 'Just chose randomly'
 'Number of patches and if there was the best possible combination there'
 'BASED ON WHAT WOULD TASTE THE BEST'
 'i prefer to teach, it makes it a learning experience'
 'i looked at the 3 visible mushrooms'
 'depending on how many pathes and if they had a good choice on the visable
patch'
 'Depending on the number of potential picks the tourist would have. If there
were many patches, I would choose to teach them.'
 'If there was one patch I instructed and if there was more than one patch I
taught.'
 'If they were going to choose their own - I taught them - if it was a simple
choice of 3 then I instructed to get the highest score'
 'If it was just one patch then I woujld instruct but if there was more then I
tended to teach as the other patches are unknow.'
 'If they were visiting more than one patch I chose to teach them'
 'If a tourist had multiple patches then i tried to teach them'
 'just varied for a change of instructin to be honest'
 'according to taste score i wanted to give most tasty ones'
 'Varied depending on what was dislayed'
 'i instructed based on if the texture and colour gave the best mushroom, if
there was more bad mushrooms i would teach and the solid texture.'
 'I tried to give them the instructions to have better chances to pick tasty
mushrooms'
 'which one I could remember'
 'If they are only visiting one patch, and there is a clear best option, I
struct them which to pick'
 'figured the best bet would be to teach them what is worth the most and what is
worth the least, so they can make informed decisions'
```

'If there was more than one patch I gave them the highest rated info.'
'I chose depending on the given scenario'
'Instruct if only visiting one patch otherwise teach.'
"Basically on the current mushrooms and the amount of patches. If there was a green or two and no blues, I'd teach against green. If one patch and blue, I'd instruct. "
'Depended on whether they were going to more where teaching would aid them further, and whether in teaching you could guide their first choice too.'
'guessed'
'Chose instruct if they were only visiting one patch; more than i decided to teach'
'depending what the mushrooms were'
'I chose to instruct if there was only one patch they were visiting and teach if there were more patches to visit'
'Depending on how many patches they would be going to'
'I instructed tourists because I felt it was easierfor the tourists. I taught some tourists that reappeared so they could know what to pick without me.'
'Teach if more than one patch visited'
'depending on how many patches they had to visit'
'by which i thought would get the tastiet mushrrom'
'the value of the visibel mushrooms versus potential value of not visible mushrooms'
'see how many patches they are visiting' 'Teaching is more polite'
'I instructed the tourists when there was a clear tastiest mushroom on offer. I taught the tourists when the selection was a little ambiguous.'
'if there were unseen patches chose to teach them'
'If they had one or more colour/pattern I would teach them what those meant for their rating but if they had no high scoring mushrooms I would instruct them to pick the highest rated one'
'asda' 'i advised them which had the highest points'
'in case of multiple patches, if blue-spotted exists in the first one, then I instruct them to choose it, otherwise I will teach them that blue is the most important feature so they focus on it in the other patches.'
'tried to share what would allow them to deduce other features'
'Based on the mushrooms present. More teaching was given as the mushrooms of any kind could spawn in other patches. Guidence was given to ensure no tourist recieved a bad tasting mushroom.'
'If there was only one patch, then it made sense to instruct them, if more than one patch then teach them to always go for blue'
'By the lot of the mushrooms'
'If there was a blue solid mushroom, always instruct.'
'Based on the mushrooms that were already shown'
'I considered if they were visiting other patches of mushrooms. '
'What seemed best'
'If there is patch they need to go on their own, then I chose to teach them. But just instruct them when I accompany them.'
'If they were visiting more than one patch'
'Instruct when only one patch, teach when more than one'

'I was trying to deduct what one piece of information would be most helpful to them to make sure they pick tasty mushrooms overall'
'If they were going to visit more patches i would teach. If they were only visiting one i would instruct'
'I liked pointing them to a specific mushroom that I knew was the tastiest.'
'If there was a big enough difference, I would teach to go for green and stripe if present'
'If it was blue stripe i instructed them'
'I looked at what was available, and tried to consider what a reasonable person might infer from the instructions. It is possible that the tourist could infer that the value of stripes, for example, was high (when directing a tourist to a blue striped mushroom) so I seldom chose direct instruction.'
'Entirely by the number of patches they would visit'
'I tried to think of information that would help them steer clear or gravitate towards things most usefully and rule out many choices'
'it depended how may patches they were visiting. I could only go to one so I had to teach them how to pick the best mushrooms'
'teach'
'When I knew the mushrooms available I instructed but when I wasn;t sure what was in the other patches I taught them instead.'
'Based on available choices to them'
'If more than one patch and visible patch had sample selection '
'If they had multiple patches, teach them so they can apply logic to those. If there is only one patch, instruct them to make sure they get the best outcome'
'I chose the easier option by telling them which mushroom to choose'
'If there was one patch, I would instruct. If there were more than one patch, I would teach.'
'Depend on how many patch they visit and option available for first patch'
'The highest bonus and how informative it would be to their decisions'
'I thought the information ws betterso tey could know for later'
'depending on how many. patches'
'depending on how many patches they were visiting, if visiting only 1 then instruct them.'
'Depends if they are choosing from multiple patches or a single patch of mushroms'
'If they had a mushroom worth 3 I instructed. If not I taught'
'If they would be visiting more than one patch I taught them. '
'Just did what i though made sense' 'Based on colour '
'I almost always chose to instruct tourists because it seemed like the easiest option.'
'random if only negatives on view' 'to guide them more' 'random'
'depended on how many patches' 'depended on mix in a patch'
'If they were going to be picking from further patches, I chose to teach them to give them a better chance of finding good mushrooms. And also if teaching would guarantee they picked the best mushroom from just one patch.'
'Preferred teach choices'
'I chose teach if there was more than 1 patch to choose from so they could work it out for themselves'

```
 'If they were going to multiple fields or not'
 'just went with what felt most appropriate'
 'Depending on how many patches they visited'
 'I was instructing themall the time as I knew the colours and numbers.'
 'unsure' 'Instruct if just one patch, teach if visiting more than one'
 'Depending upon the easiness for them to understand while picking.'
 'because of the number of patches they were going to'
 'I went for the best possible points'
 'If there was more than one mushroom patch I would teach rather than instruct'
 'I instructed them when they visited only one patch and teach them if they
visited more than one.']

Anything Confusing:

['no' 'no'
 "The third question of the first set of questions to which the answer was 'all
of the above' I don't remember reading that information"
 'Yes, I got initially mixed up with the colours by themselves and solids.'
 'n/a' 'No' 'N/A'
 'INSTRUCTIONS REGARDING VALUES COULD HAVE BEEN CLEARER, BUT WELL PRESENTED
OVERALL'
 'no' 'no' 'NO' 'No' 'no' 'no - it was fun' 'No' 'No' 'no'
 'not sure i think i got it right!' 'none' 'no' 'no, it was clear.' 'no'
 'nope'
 "I wasn't totally sure if the visible mushroom patch was an indication of how
common certain types of mushroom would be"
 'no' 'No' 'no' 'No' 'No' 'No' 'no' 'No, the instructions were excellent!'
 'no' 'no' 'No' 'No.' 'No' 'no' 'no' 'No' 'no' 'no' 'No' 'no' 'No' 'asd'
 'none' 'no' 'no' 'None. Very clear and concise.' 'No' 'NO' 'NA' 'N/A'
 'No. ' 'no' 'no' 'No' 'None' 'No' 'no' 'No' 'No' 'no'
 'No, everything was really clear.' 'No' 'Nothing' 'o' 'no ' 'No' 'no'
 'No' 'No' 'no' "I didn't understand the goal of the experiment. " 'No'
 'No' 'No' 'no' 'no' 'No' 'No' 'Nothing at all. ' 'no' 'no' 'None' 'no'
 'no' 'no' 'no' 'no' 'No' 'No I enjoyed the challenge ' 'No' 'no' 'no'
 'Sometimes unsure on whether to teach or instruct'
 'Nothing was unclear or confusing. All instructions were clear enough.'
 'no' 'N/A' 'No' 'no' 'No, it was fine' 'No' 'No']

Misc Feedback:

['no' 'no' 'None, thank you' 'Just to make the above a little clearer. '
 'n/a' 'No' 'N/A'
 'WELL PRESENTED AND GOOD USE OF PICTURES / TABULAR PRESENTATION.  MAYBE
PRACTICE WOULD BE GOOD WITH OUTCOME TO SHOW HOW IT WORKS,'
 'interesting study!' 'no' 'NO' 'No' 'n/a'
 'I liked this a great deal - thank you' 'No' 'No'
 'Thanks for the opportunity' 'no' 'none' 'was interesting'
 'no, but i did enjoy this study. It was fun.' 'no' 'nope!' 'No'
```

```
 'no further comments' 'No' 'no' 'No' 'Hope my answers are helpful ' 'No'
 'no' 'This was a very enjoyable experiment. Thank you.' 'no' 'no' 'No'
 'None.' 'No' 'none' 'no' 'good fun, thanks!' 'no' 'no thanks'
 'Fun study!' 'no' '-' 'ad' 'none' 'no' 'NO' 'A Fun experiment.'
 'Thanks for the study, it was a lot of fun.' 'Interesting task' 'NA'
 'n/a' 'No.' 'no sorry, was fun' 'no' 'None' 'It was fun, thank you' 'No'
 'no' 'No' 'None, it was fun though.' 'none'
 "I thought it was a lot of fun to work through this, although I have no idea
what information you'll get from it!"
 'It was extremely fun and i would love to participate in similar experiments
you run in the future.'
 'I enjoyed this task' 'no' 'no' 'No' 'no' 'thanks' 'No'
 'interesting study about learning'
 'I also would like to know whether my answers were correct.' 'No' 'No.'
 'No' 'no' 'no' 'Its pretty interesting :)' 'This was fun'
 'Nothing at all. ' 'no' 'no' 'No.' 'No, it was actually enjoyable' 'no'
 'no' 'none' 'none' 'No' 'Loved the test ' 'None' 'no' 'no' 'No'
 'It was an enjoyable task.' 'no' 'N/A' 'No' 'no' 'No it was all good'
 'No' 'No, thank you.']

/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/1776341517.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exit_survey[question] = exit_survey.data.apply(lambda x:
x["response"][question])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/1776341517.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exit_survey[question] = exit_survey.data.apply(lambda x:
x["response"][question])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/1776341517.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exit_survey[question] = exit_survey.data.apply(lambda x:
x["response"][question])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/1776341517.py:2
```

```
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exit_survey[question] = exit_survey.data.apply(lambda x:
x["response"][question])
```

3 Catch Trials

Do people get these right?
What is % correct by person?
By context / question? (are some harder than others?)

[13]:
```
select_trials = trials[trials.trial_type == 'survey-html-form']
catch_trials = select_trials[select_trials['data'].apply(lambda x: bool(x.
 ↪get('features', 'all') != 'all'))]

catch_trials["belief_model"] = catch_trials.data.apply(lambda x: 1 if x.
 ↪get("belief") else 0)
catch_trials["response"] = catch_trials.data.apply(lambda x:␣
 ↪bool(int(x['response']["utterance"])))
catch_trials["catch_trial_correct"] = catch_trials["belief_model"] ==␣
 ↪catch_trials["response"]

catch_trials["feature"] = catch_trials.data.apply(lambda x: x.
 ↪get("features")[0])
catch_trials["feature_value"] = catch_trials.data.apply(lambda x: x.
 ↪get("values")[0])

# Grab only the first action context, so multi-horizon participants join␣
 ↪correctly
catch_trials["action_context"] = catch_trials.data.apply(lambda x: str([x.
 ↪get("action_context")[0]]))

catch_trials["action_context"] = catch_trials.data.apply(lambda x: str(x.
 ↪get("action_context")))
# catch_trials["context_hash"] = catch_trials.action_context.apply(hash_id)
```

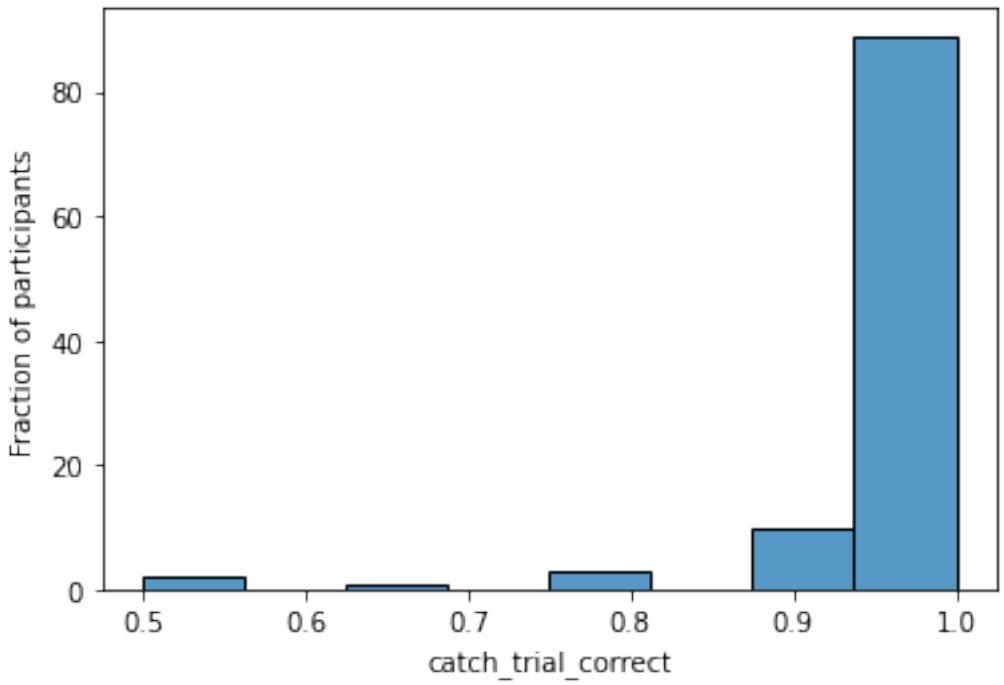
```
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  catch_trials["belief_model"] = catch_trials.data.apply(lambda x: 1 if
x.get("belief") else 0)
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["response"] = catch_trials.data.apply(lambda x:
bool(int(x['response']["utterance"])))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:6
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["catch_trial_correct"] = catch_trials["belief_model"] ==
catch_trials["response"]
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:8
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["feature"] = catch_trials.data.apply(lambda x:
x.get("features")[0])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["feature_value"] = catch_trials.data.apply(lambda x:
x.get("values")[0])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:1
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["action_context"] = catch_trials.data.apply(lambda x:
str([x.get("action_context")[0]]))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3387171192.py:1
```

```
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  catch_trials["action_context"] = catch_trials.data.apply(lambda x:
str(x.get("action_context")))
```

[14]:
```
participant_catch_results = catch_trials[["catch_trial_correct",  "workerid"]].
 ↪groupby(["workerid"]).mean().reset_index()
participant_catch_results["passed_catch_trial"] = participant_catch_results.
 ↪catch_trial_correct > .75

sns.histplot(participant_catch_results, x="catch_trial_correct",␣
 ↪multiple="dodge")
plt.ylabel("Fraction of participants")
```

[14]:
```
Text(0, 0.5, 'Fraction of participants')
```



[15]:
```
# Join catch results into main dataframe to simplify analysis
joined_trials = trials.set_index("workerid").
 ↪join(participant_catch_results[["passed_catch_trial", "catch_trial_correct",␣
 ↪"workerid"]].set_index('workerid'))
```

```
joined_trials = joined_trials.join(quizResults).reset_index()

joined_trials.passed_catch_trial = joined_trials.passed_catch_trial.
 ↪fillna(False)
joined_trials["no_catch_data"] = joined_trials.passed_catch_trial.isnull()
```

[16]:
```
print("{} unique workers before join, {} after.".format(trials.workerid.
 ↪nunique(), joined_trials.workerid.nunique()))

# joined_trials.groupby(["objective", "horizon", "passed_catch_trial"]).
 ↪workerid.nunique()
```

```
119 unique workers before join, 119 after.
```

[17]:
```
joined_trials.groupby(["passed_quiz", "passed_catch_trial"]).workerid.nunique()
```

[17]:
```
passed_quiz  passed_catch_trial
False        False                 14
True         False                  5
             True                  99
Name: workerid, dtype: int64
```

3.1 Drop participants that did not pass

[18]:
```
joined_trials = joined_trials[joined_trials.passed_catch_trial]
```

3.2 Visualize Responses

[19]:
```
select_trials = joined_trials[joined_trials.trial_type == 'survey-html-form']

exp1_trials = select_trials[select_trials["data"].apply(lambda x: x.
 ↪get("features", "all") == "all")]

len(exp1_trials)
```

[19]:
```
2772
```

[20]:
```
exp1_trials.data.iloc[1]["response"]
```

[20]:
```
{'feature': 'Green', 'feature_value': '2', 'texture': '', 'color': ''}
```

[21]:
```
exp1_trials["feature"] = exp1_trials.data.apply(lambda x:␣
 ↪x["response"]["feature"])
exp1_trials["feature_value"] = exp1_trials.data.apply(lambda x:␣
 ↪x["response"]["feature_value"])
```

```
exp1_trials["instruction_texture"] = exp1_trials.data.apply(lambda x:␣
↪x["response"]["texture"])
exp1_trials["instruction_color"] = exp1_trials.data.apply(lambda x:␣
↪x["response"]["color"])
exp1_trials["horizon"] = exp1_trials.data.apply(lambda x: int(x["horizon"]))

exp1_trials["utterance"] = exp1_trials.data.apply(lambda x: x["response"])

exp1_trials["instruction"] = exp1_trials.data.apply(lambda x:␣
↪(x["response"]["feature"]) == "")

exp1_trials["feature_value"] = exp1_trials.apply(lambda x:␣
↪int(x["feature_value"]) if not x["instruction"] else None, axis=1)

# Sanity check chosen messages
exp1_trials.feature.value_counts()
```

```
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["feature"] = exp1_trials.data.apply(lambda x:
x["response"]["feature"])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["feature_value"] = exp1_trials.data.apply(lambda x:
x["response"]["feature_value"])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["instruction_texture"] = exp1_trials.data.apply(lambda x:
x["response"]["texture"])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["instruction_color"] = exp1_trials.data.apply(lambda x:
x["response"]["color"])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["horizon"] = exp1_trials.data.apply(lambda x: int(x["horizon"]))
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:7
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["utterance"] = exp1_trials.data.apply(lambda x: x["response"])
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["instruction"] = exp1_trials.data.apply(lambda x:
(x["response"]["feature"]) == "")
/var/folders/gv/42lb0z1j4dxf3wsk74nrxwx80000gn/T/ipykernel_60670/3512373998.py:1
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exp1_trials["feature_value"] = exp1_trials.apply(lambda x:
int(x["feature_value"]) if not x["instruction"] else None, axis=1)
```

[21]:
```
           1203
Green       860
Blue        389
Spotted     213
Striped     107
Name: feature, dtype: int64
```

[22]:
```
def align_instructions(row):

    if not row["instruction"]:
        return row

    # Is this row aligned already? If so, return
    if row["instruction_texture"] in ["Spotted", "Solid", "Striped"]:
        return row

    # Otherwise, switch 'em
    row_color = row["instruction_texture"]
    row["instruction_texture"] = row["instruction_color"]
    row["instruction_color"] = row_color

    return row

exp1_trials = exp1_trials.apply(align_instructions, axis=1)## Visualize␣
↪Responses
```

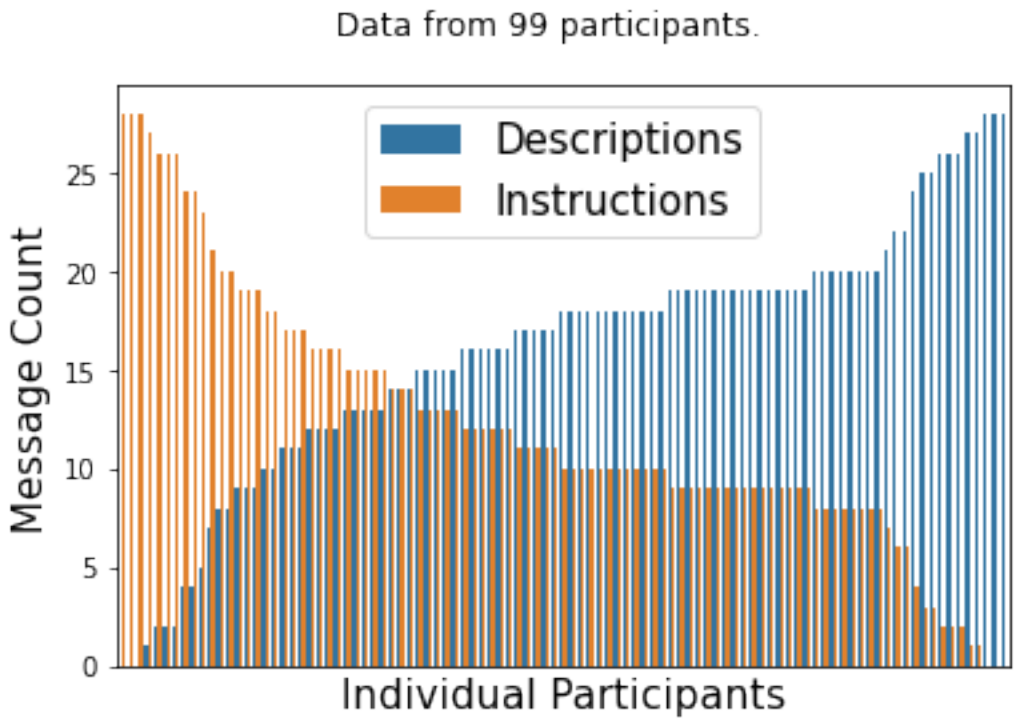
[23]:
```
exp1_trials["message_type"] = exp1_trials.instruction.apply(lambda x:␣
↪"Instructions" if x else "Descriptions")

n_instructions = exp1_trials[exp1_trials["message_type"] == "Instructions"].
↪workerid.value_counts()
```

[24]:
```
n_instructions = exp1_trials[exp1_trials["message_type"] == "Instructions"].
↪workerid.value_counts()

exp1_joined = pd.merge(exp1_trials, n_instructions.
↪to_frame(name="n_instructions"),
                       left_on='workerid', right_index=True, how='left').
↪reset_index()

exp1_joined["n_instructions"].fillna(0, inplace=True)

exp1_trials = exp1_joined
```

[25]:
```
ordering = exp1_trials.drop_duplicates(["workerid", "n_instructions"]).
↪sort_values("n_instructions", ascending=False).workerid

sns.countplot(data=exp1_trials, x="workerid", hue="message_type",␣
↪order=ordering, hue_order=["Descriptions", "Instructions"])
plt.suptitle("Data from {} participants.".format(exp1_trials.workerid.
↪nunique()))
plt.xticks([])
plt.xlabel("Individual Participants", fontsize=15)
```

```
plt.legend(loc='best', fontsize=15)
plt.ylabel("Message Count", fontsize=15)
```

[25]:
```
Text(0, 0.5, 'Message Count')
```



[26]:
```
exp1_trials[exp1_trials.n_instructions.isin([0, 28])].workerid.nunique()
```

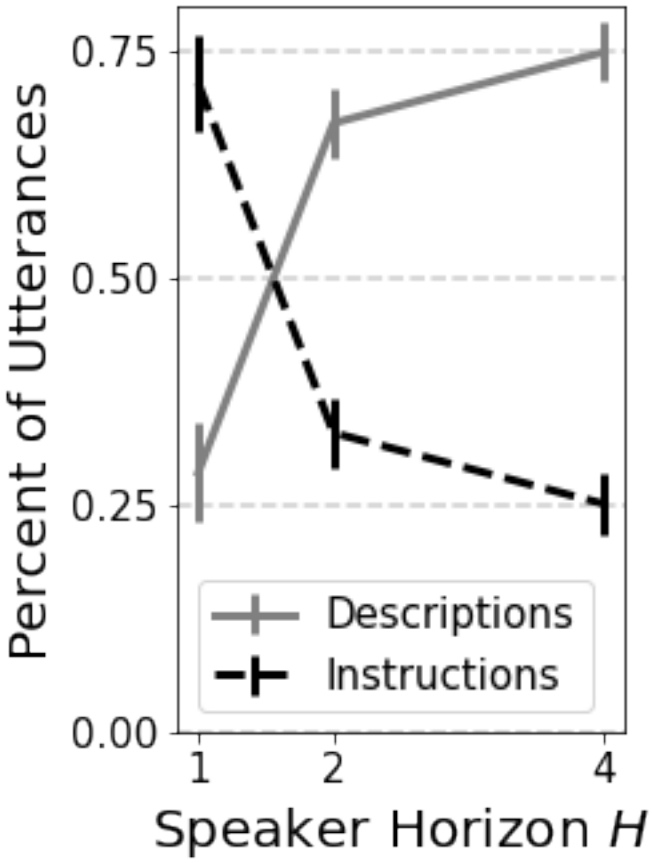
[26]:
```
6
```

[27]:
```
count_by_horizon = exp1_trials.groupby(["horizon", "message_type"]).size().
 ↪reset_index()
count_by_horizon = count_by_horizon.rename({0:"size"}, axis=1)

count_by_horizon["pct"] = count_by_horizon.groupby("horizon")["size"].
 ↪transform(lambda x: x/x.sum())
# count_by_horizon.size()#.transform(lambda x: x / x.sum())
# mean_by_horizon = exp1_trials
```

[28]:
```
import math

descriptions = count_by_horizon[count_by_horizon.message_type == "Descriptions"]
instructions = count_by_horizon[count_by_horizon.message_type != "Descriptions"]
```

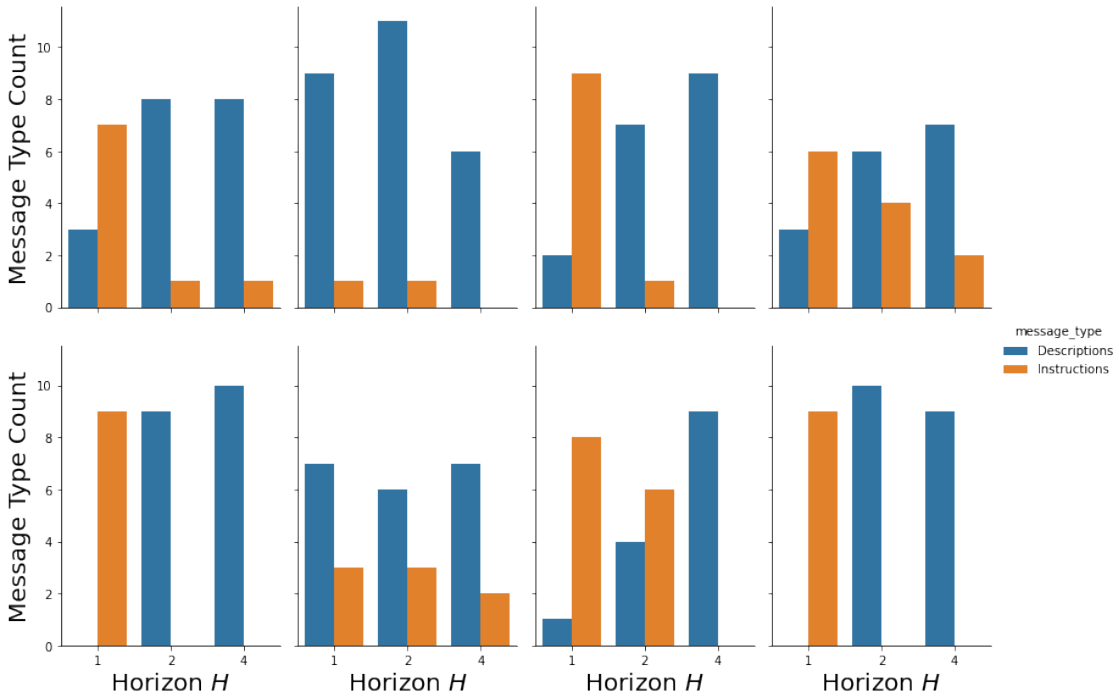
```
# plt.plot(instructions.horizon, instructions.pct, c='black', linewidth = 5)
# plt.plot(descriptions.horizon, descriptions.pct, c='black', linewidth = 5)

description_err = descriptions.apply(lambda x: 1.96 * math.
 ↪sqrt(x['pct']*(1-x['pct'])/x['size']), axis=1)
# instruction_err = instructions.apply(lambda x: 1.96 * math.
 ↪sqrt(x['pct']*(1-x['pct'])/x['size']), axis=1)

plt.figure(figsize=(3,5))
ax = plt.gca()
ax.errorbar(descriptions.horizon, descriptions.pct, description_err, ␣
 ↪color='gray', ecolor='gray',  linewidth=3, label="Descriptions")
ax.errorbar(instructions.horizon, instructions.pct, description_err, ␣
 ↪color='k', ecolor='k', linewidth=3, linestyle='--', label="Instructions")

plt.legend(loc='best', fontsize=15)
ax.set_ylim(0, .8)
plt.xticks([1, 2, 4], fontsize=15)
ys = [0, .25, .5, .75]
for y in ys:
    plt.axhline(y, alpha=.2, linestyle='--', c='k', zorder=0)
plt.yticks(ys, fontsize=15);

plt.ylabel("Percent of Utterances", fontsize=20)
plt.xlabel("Speaker Horizon $H$", fontsize=20)
```

[28]: Text(0.5, 0, 'Speaker Horizon H')



[29]:
```
random_ids = exp1_trials.workerid.sample(8)

random_sample = exp1_trials[exp1_trials.workerid.isin(random_ids)]

g = sns.catplot(x="horizon", hue="message_type", col="workerid",
                data=random_sample, kind="count", col_wrap = 4,␣
 ↪hue_order=["Descriptions", "Instructions"],
                height=4, aspect=.7);

g.set_titles("")
g.set_axis_labels(x_var="Horizon $H$", y_var="Message Type Count", fontsize=20)
```

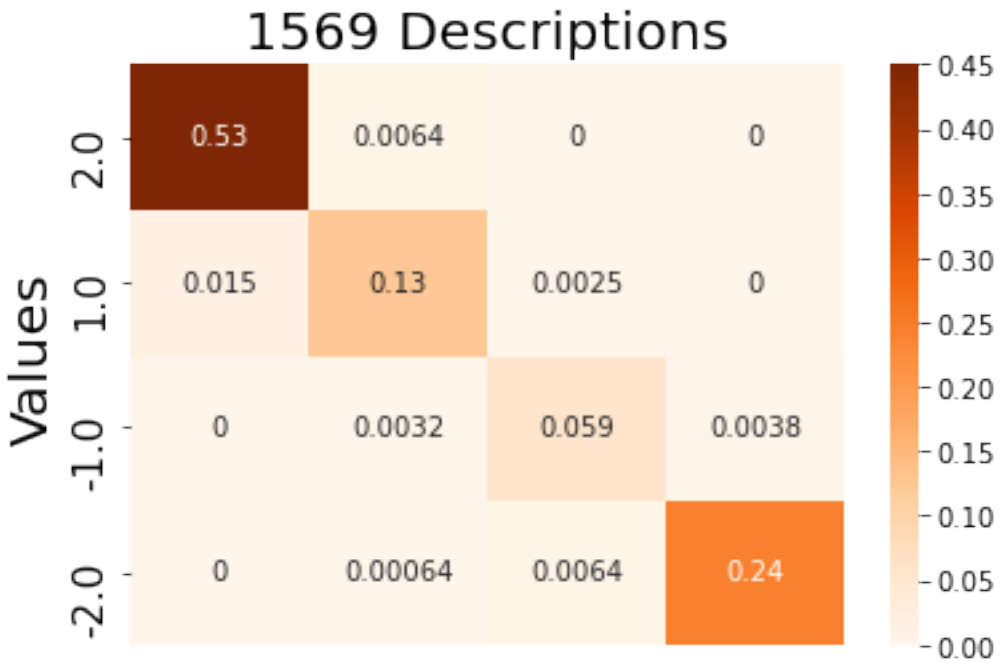
[29]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb981635cd0>
```



[30]:
```
def heatmap_descriptions(to_plot_messages, title="", vmax=None):

    descriptions = to_plot_messages[~to_plot_messages.instruction]
    if len(descriptions) == 0:
        print("{}: found 0 descriptions.".format(to_plot_messages.workerid.
↪unique()))
        return

    to_plot_messages = descriptions

    percent_of_messages = (to_plot_messages.groupby(["feature",␣
↪"feature_value"]).size() / len(to_plot_messages)).reset_index()

    unique_values = set(percent_of_messages.feature_value.unique())
    missing_vals = unique_values.symmetric_difference({-2, -1, 1, 2})

    if missing_vals:
        print("Message with val {} not found-- appending.".format(missing_vals))
        for val in missing_vals:
            percent_of_messages = percent_of_messages.append({"feature":␣
↪"Blue", "feature_value": val}, ignore_index=True)

    message_table = percent_of_messages.pivot("feature_value", "feature", 0)
```

```
    features_in_descending_order = ["Green", "Spotted", "Striped", "Blue"]
    message_table = message_table.reindex(features_in_descending_order, axis=1).
 ↪fillna(0)

    plt.figure()
    ax = sns.heatmap(message_table, cmap="Oranges", vmax=vmax, annot=True)
    ax.invert_yaxis()
    ax.set_title("{} Descriptions".format(len(to_plot_messages)), fontsize=20)
#     ax.set_xticks([])
#     ax.set_yticks([])
    ax.set_xlabel("Features")
    ax.set_ylabel("Values")

heatmap_descriptions(exp1_trials, vmax=.45)#, annot=True)
plt.yticks(fontsize=15)
plt.ylabel("Values", fontsize=20)
plt.xticks([])
plt.xlabel("")
```

[30]: Text(0.5, 15.0, '')

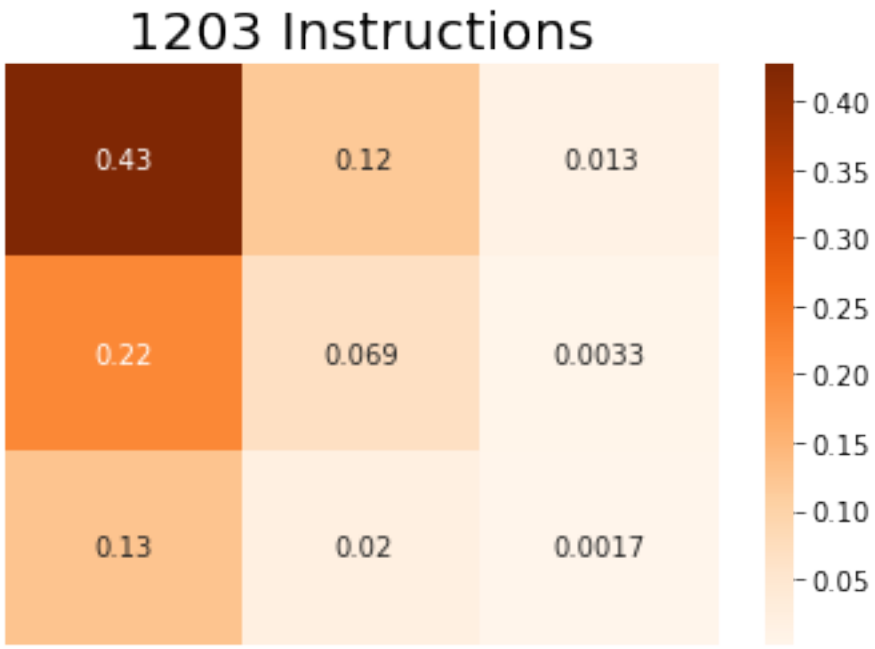


[31]:
```
instructions = exp1_trials[exp1_trials.instruction]

instructions.groupby(["instruction_color", "instruction_texture"]).size()
```

[31]:
```
instruction_color  instruction_texture
Blue               Solid                  4
                   Spotted               16
                   Striped                2
Green              Solid                266
                   Spotted              514
                   Striped              152
Red                Solid                 83
                   Spotted              142
                   Striped               24
dtype: int64
```

[32]:
```
def heatmap_instructions(to_plot_messages, title="", vmax=None):

    instructions = to_plot_messages[to_plot_messages.instruction]
    if len(instructions) == 0:
        print("{}: found 0 descriptions.".format(to_plot_messages.workerid.
↪unique()))
        return

    to_plot_messages = instructions

    percent_of_messages = (to_plot_messages.groupby(["instruction_color",␣
↪"instruction_texture"]).size() / len(to_plot_messages)).reset_index()

    message_table = percent_of_messages.pivot("instruction_texture",␣
↪"instruction_color", 0)

    color_order = ["Green", "Red",  "Blue"]
    texture_order = ["Spotted", "Solid", "Striped"]

    message_table = message_table.reindex(color_order, axis=1).fillna(0)
    message_table = message_table.reindex(texture_order, axis=0).fillna(0)

    plt.figure()
    ax = sns.heatmap(message_table, cmap="Oranges", annot=True)
    ax.set_title(title, fontsize=20)
    ax.set_xlabel("")
    ax.set_ylabel("")

title = "{} Instructions".format(len(instructions))

heatmap_instructions(instructions, title=title, vmax=.45)
plt.xticks([])
plt.yticks([])
```

[32]:
```
([], [])
```



3.3 Export for R

[33]:
```
def utterance_to_string(utt):
    if utt['texture'] != '':
        return f'{utt["texture"]} {utt["color"]}'
    else:
        return f'{utt["feature"]} {utt["feature_value"]}'

exp1_trials['utterance_for_r'] = exp1_trials.utterance.apply(lambda x:␣
↪utterance_to_string(x))
```

[34]:
```
exp1_trials[['horizon', 'utterance_for_r']].to_csv("utterance_counts_for_r.csv")
```

[35]:
```
exp1_trials.horizon.value_counts()
```

[35]:
```
1    939
2    917
4    916
Name: horizon, dtype: int64
```

[36]:
```
exp1_trials["utt"] = exp1_trials.data.apply(lambda x: x.get('response'))
```

[37]:
```
exp_to_theory_map = {"Blue": "blue", "Green": "green", "Red": "red",
                     "Spotted": "circle", "Solid": "triangle", "Striped":␣
↪"square"}
```

```
def extract_single_action_trials(trial):

    action_context = trial["data"]["action_context"][0]
    context = [{"color": exp_to_theory_map[a["color"]], "shape":␣
 ↪exp_to_theory_map[a["texture"]]} for a in action_context]

    base = {"action_context": context,
            "workerid": trial["workerid"],
            "horizon": trial["horizon"]}

    utt = {}

    if trial["instruction"]:
        utt["color"] = exp_to_theory_map[trial["instruction_color"]]
        utt["shape"] = exp_to_theory_map[trial["instruction_texture"]]
        utt["type"] = "instruction"

    elif not trial["instruction"]:
        utt["type"] = "description"
        utt["feature"] = exp_to_theory_map[trial["feature"]]
        utt["value"] = int(trial["feature_value"])
    base["utt"] = utt
    return base
```

[38]:
```
results = exp1_trials.apply(lambda x: extract_single_action_trials(x), axis=1)
```

[39]:
```
json.dump(list(results.values), open("data/exp_utterances.json", "w"))
```