

# Results

Notes:

- Some of this is already in or was based on the blogpost/interface code. Hit show to see code. I switch between R and Python
- Some of this won't make it to the paper. You can probably skip preprocessing unless you want to check certain things, example: did we make sure to remove judgments based on X condition. To find bits that I think will be in the paper search [insight]
- If you want to clarify/comment anything do so at <https://github.com/sm11197/sm11197.github.io/blob/main/analysis/debate-0923.Rmd> or message me elsewhere

## Preprocessing

### Importing, filtering, and adding columns

We have 3 sets of data from the interface:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
pd.options.mode.chained_assignment = None # default='warn'

# Load summaries that can be downloaded from the interface
debates = pd.read_csv("/Users/bila/git/for-debate/debate/save/official/summaries/debates.csv", keep_def
sessions = pd.read_csv("/Users/bila/git/for-debate/debate/save/official/summaries/sessions.csv", keep_d
turns = pd.read_csv("/Users/bila/git/for-debate/debate/save/official/summaries/turns.csv", keep_default
print(f' {debates.shape} - Debates') ;print(f'{sessions.shape} - Sessions, which has multiple rows (of p

## (632, 29) - Debates
## (1863, 46) - Sessions, which has multiple rows (of participants) for each debate
## (6220, 16) - and Turns, which has multiple rows (of participant turns) for each debate

# Only include debates within a given period
debates["Start time"] = pd.to_datetime(debates["Start time"], unit="ms")
debates["End time"] = pd.to_datetime(debates["End time"], unit="ms")
debates["Last modified time"] = pd.to_datetime(debates["Last modified time"], unit="ms")
debates = debates[
    (debates["Start time"] > pd.to_datetime("10/02/23", format="%d/%m/%y")) &
    (debates["End time"] < pd.to_datetime("01/09/23", format="%d/%m/%y"))
]
### for filtering to when we had AI debates: 16/07/23
# Filter sessions & turns to only the selected debates
sessions = sessions.merge(debates[["Room name"]], how="inner", on="Room name")
turns = turns.merge(debates[["Room name"]], how="inner", on="Room name")
print(f'We have {len(debates)} debates when filtering out the initial pilots last fall')
```

```
## We have 583 debates when filtering out the initial pilots last fall
```

```
# Secondary analysis: Question Difficulty
```

```
# Create new columns with bin labels
```

```
debates['Untimed annotator context bins'] = pd.cut(debates['Untimed annotator context'].round(), bins=[
```

```
debates['Speed annotator accuracy bins'] = pd.cut(debates['Speed annotator accuracy'], bins=[-0.999, 0.
```

```
## respectively, those speed annotator accuracies probably mean 0 right, 1 right, 2 right
```

```
debates['Final_Accuracy'] = debates['Final probability correct'] > 0.5
```

```
print(f'Average accuracy per context required by question:\n{debates.groupby("Untimed annotator context
```

```
## Average accuracy per context required by question:
```

```
##                                     Proportion_True  Total_Count
```

```
## Untimed annotator context bins
```

```
## 1                                0.781250                64
```

```
## 2                                0.711382               246
```

```
## 3                                0.702857               175
```

```
## 4                                0.632653                98
```

```
print(f'Average accuracy per difficulty based on speed annotator accuracy:\n{debates.groupby("Speed ann
```

```
## Average accuracy per difficulty based on speed annotator accuracy:
```

```
##                                     Proportion_True  Total_Count
```

```
## Speed annotator accuracy bins
```

```
## 0                                0.728682               129
```

```
## 0.2                              0.697509               281
```

```
## 0.4                              0.694118               170
```

```
## Hm, this seems less likely to be a good indicator of question difficulty
```

```
# Determine settings for each row
```

```
def setups(row):
```

```
    if 'GPT-4' in (row['Honest debater'], row['Dishonest debater']):
```

```
        if row['Is single debater']:
```

```
            return "AI Consultancy " + ("Honest" if row['Has honest debater'] else "Dishonest")
```

```
        else:
```

```
            return "AI Debate"
```

```
    else:
```

```
        if row['Is single debater']:
```

```
            return "Human Consultancy " + ("Honest" if row['Has honest debater'] else "Dishonest")
```

```
        else:
```

```
            return "Human Debate"
```

```
debates['Setting'] = debates.apply(setups, axis=1)
```

```
# Agregate settings - the 4 that we normally talk about:
```

```
debates['Final_Setting'] = debates['Setting'].str.replace(' Honest', '').str.replace(' Dishonest', '')
```

## Merging, filtering for judgments

```

# Merge sessions with debates, so we have each judge's final probability correct and the debate's metadata
source = sessions.merge(
    debates[["Room name", "Debater A", "Debater B", "Honest debater", "Dishonest debater",
            "Is single debater", 'Has honest debater',
            "Final_Setting", "Setting",
            "Question", "Article ID",
            "Speed annotator accuracy bins", "Untimed annotator context bins",
            "Speed annotator accuracy", "Untimed annotator context", "Is offline",
            'End time', 'Last modified time']],
    how="left",
    on="Room name",
)

```

```

print(f'After merging debates with sessions, we have the following participant counts for those debates:

```

```

## After merging debates with sessions, we have the following participant counts for those debates:
## Judge          549
## Debater B      487
## Debater A      458
## Offline Judge  223
## Name: Role, dtype: int64

```

```

# [source['Is over']] == True] to check for completed online/offline debates

```

```

# Filter out incomplete judgments

```

```

judgments = source[source['Final probability correct'].notnull()]
print(f'After filtering to judges that have finalized their judgment, we have the following judgments per room:

```

```

## After filtering to judges that have finalized their judgment, we have the following judgments per room:
## Judge          508
## Offline Judge   214
## Name: Role, dtype: int64
## for a total of 722 judgments.

```

```

judgments['Final_Accuracy'] = judgments['Final probability correct'] > 0.5

```

```

print(f'Of those judgments, we have this much for each setting (not consolidating honest - dishonest consultations):

```

```

## Of those judgments, we have this much for each setting (not consolidating honest - dishonest consultations):
## Human Debate          413
## AI Debate              92
## Human Consultancy Dishonest  68
## AI Consultancy Honest    56
## Human Consultancy Honest   53
## AI Consultancy Dishonest  40
## Name: Setting, dtype: int64

```

```

print(f'Of those judgments, we have this much for each setting (aggregated):\n{judgments.groupby("Final_Setting").sum()}

```

```

## Of those judgments, we have this much for each setting (aggregated):
##          Proportion_True  Total_Count

```

```
## Final_Setting
## AI Consultancy          0.802083          96
## AI Debate               0.782609          92
## Human Consultancy       0.719008         121
## Human Debate            0.876513         413
```

```
# Remove judges who see the story more than once
judgments['base_room_name'] = judgments['Room name'].str.extract('(.*?)\d+$', expand=False).fillna(judgments['Room name'])
judgments = judgments.sort_values(by=['base_room_name', 'End time']).groupby(['Participant', 'base_room_name']).first()

print(f'1. We then filter to judgments where the judge has only seen a story once, and now we have this much')

```

```
## 1. We then filter to judgments where the judge has only seen a story once, and now we have this much
##
## Proportion_True Total_Count
## Final_Setting
## AI Consultancy          0.802083          96
## AI Debate               0.782609          92
## Human Consultancy       0.719008         121
## Human Debate            0.867374         377
```

```
# Filter to online judges only
judgments_online = judgments[judgments["Role"] == "Judge"]
print(f'2. We\'ll make a copy of the online judgments only leaving us with the following judgments:\n{j}')

```

```
## 2. We\'ll make a copy of the online judgments only leaving us with the following judgments:
##
## Proportion_True Total_Count
## Final_Setting
## AI Consultancy          0.797872          94
## AI Debate               0.791209          91
## Human Consultancy       0.709091         110
## Human Debate            0.861538         195
```

```
judgments_online = judgments_online[judgments_online['Untimed annotator context bins'].isin(['2', '3', '4'])]
print(f'3. We then filter to judgments which require more than a sentence or two, and now we have this much')

```

```
## 3. We then filter to judgments which require more than a sentence or two, and now we have this much
##
## Proportion_True Total_Count
## Final_Setting
## AI Consultancy          0.806452          93
## AI Debate               0.781609          87
## Human Consultancy       0.700935         107
## Human Debate            0.838710         155
```

```
pd.set_option('display.max_columns', None)
total_counts_for_setting = judgments_online.groupby('Final_Setting').size()
result = judgments_online.groupby(["Final_Setting", "Untimed annotator context bins"]).agg(
    Proportion_True=pd.NamedAgg(column='Final_Accuracy', aggfunc=lambda x: x.mean()),
    Context_Count=pd.NamedAgg(column='Final_Accuracy', aggfunc='size'),
    Proportion_Context=pd.NamedAgg(column='Final_Setting', aggfunc=lambda x: len(x) / total_counts_for_setting)
)
print(f'Are the difficult questions equally enough distributed amongst settings?:\n{result}')
```

```

## Are the difficult questions equally enough distributed amongst settings?:
##                                     Proportion_True \
## Final_Setting      Untimed annotator context bins
## AI Consultancy      1                                     NaN
##                                     2                     0.823529
##                                     3                     0.826087
##                                     4                     0.736842
## AI Debate           1                                     NaN
##                                     2                     0.777778
##                                     3                     0.772727
##                                     4                     0.800000
## Human Consultancy   1                                     NaN
##                                     2                     0.634146
##                                     3                     0.708333
##                                     4                     0.833333
## Human Debate        1                                     NaN
##                                     2                     0.890411
##                                     3                     0.816667
##                                     4                     0.727273
##
##                                     Context_Count \
## Final_Setting      Untimed annotator context bins
## AI Consultancy      1                                     0
##                                     2                     51
##                                     3                     23
##                                     4                     19
## AI Debate           1                                     0
##                                     2                     45
##                                     3                     22
##                                     4                     20
## Human Consultancy   1                                     0
##                                     2                     41
##                                     3                     48
##                                     4                     18
## Human Debate        1                                     0
##                                     2                     73
##                                     3                     60
##                                     4                     22
##
##                                     Proportion_Context
## Final_Setting      Untimed annotator context bins
## AI Consultancy      1                                     NaN
##                                     2                     0.548387
##                                     3                     0.247312
##                                     4                     0.204301
## AI Debate           1                                     NaN
##                                     2                     0.517241
##                                     3                     0.252874
##                                     4                     0.229885
## Human Consultancy   1                                     NaN
##                                     2                     0.383178
##                                     3                     0.448598
##                                     4                     0.168224
## Human Debate        1                                     NaN

```

##	2	0.470968
##	3	0.387097
##	4	0.141935

```
pd.reset_option('display.max_columns')
```

So question difficulty isn't perfectly balanced... but consultancies have a different relationship with question difficulty anyway? **need a second opinion**

## Trying to balance the data

1. Balancing honest & dishonest consultancies
2. Question weights

### Balancing honest & dishonest consultancies

```
def balance_consultancies(df, sample_setting, random_state):
    """
    Sample distinct questions, then use common questions, ensure equal counts.
    """
    consult_df = df[df['Setting'].str.contains(sample_setting, na=False)]
    honest_df = consult_df[consult_df['Setting'].str.contains('Honest')]
    dishonest_df = consult_df[consult_df['Setting'].str.contains('Dishonest')]
    sample_column_name = f'{sample_setting} Sample'
    df[sample_column_name] = False
    # Separate into distinct and common questions
    # First, let's extract the combinations of 'Article ID' and 'Question' for both honest and dishonest
    honest_combinations = set(honest_df[['Article ID', 'Question']].itertuples(index=False, name=None))
    dishonest_combinations = set(dishonest_df[['Article ID', 'Question']].itertuples(index=False, name=None))
    # Identifying the common and distinct combinations
    common_combinations = honest_combinations.intersection(dishonest_combinations)
    distinct_honest_combinations = honest_combinations - common_combinations
    distinct_dishonest_combinations = dishonest_combinations - common_combinations
    # Filtering the original dataframes based on these combinations to get distinct and common dataframes
    common_honest_df = honest_df[honest_df.set_index(['Article ID', 'Question']).index.isin(common_combinations)]
    common_dishonest_df = dishonest_df[dishonest_df.set_index(['Article ID', 'Question']).index.isin(common_combinations)]
    distinct_honest_df = honest_df[honest_df.set_index(['Article ID', 'Question']).index.isin(distinct_honest_combinations)]
    distinct_dishonest_df = dishonest_df[dishonest_df.set_index(['Article ID', 'Question']).index.isin(distinct_dishonest_combinations)]
    def extract_correct_index(sample_df):
        if isinstance(sample_df.index, pd.MultiIndex):
            return sample_df.index.get_level_values(2)
        else:
            return sample_df.index
    # Get distinct consultancies
    sample_size = min(len(distinct_honest_df.groupby(['Question', 'Article ID'])), len(distinct_dishonest_df.groupby(['Question', 'Article ID'])))
    honest_sample = distinct_honest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(1, random_state=random_state))
    dishonest_sample = distinct_dishonest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(1, random_state=random_state))
    df.loc[extract_correct_index(honest_sample), sample_column_name] = True
    df.loc[extract_correct_index(dishonest_sample), sample_column_name] = True
    # Drop sampled questions from distinct dataframes
    honest_remove_distinct = set(honest_sample[['Article ID', 'Question']].itertuples(index=False, name=None))
```

```

dishonest_remove_distinct = set(dishonest_sample[['Article ID', 'Question']].itertuples(index=False))
distinct_honest_df = distinct_honest_df[~distinct_honest_df.set_index(['Article ID', 'Question']).index.isin(dishonest_remove_distinct)]
distinct_dishonest_df = distinct_dishonest_df[~distinct_dishonest_df.set_index(['Article ID', 'Question']).index.isin(dishonest_remove_distinct)]
honest_distinct_remaining = len(distinct_honest_df.groupby(['Question', 'Article ID']))
dishonest_distinct_remaining = len(distinct_dishonest_df.groupby(['Question', 'Article ID']))
# Sample from remaining distinct questions, using common questions for the other (bigger count) set
if honest_distinct_remaining > dishonest_distinct_remaining:
    sample_size = min(honest_distinct_remaining, len(common_dishonest_df.groupby(['Question', 'Article ID'])))
    honest_sample = distinct_honest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    dishonest_sample = common_dishonest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    df.loc[extract_correct_index(dishonest_sample), sample_column_name] = True
    df.loc[extract_correct_index(honest_sample), sample_column_name] = True
    dishonest_remove_common = set(dishonest_sample[['Article ID', 'Question']].itertuples(index=False))
    common_dishonest_df = common_dishonest_df[~common_dishonest_df.set_index(['Article ID', 'Question']).index.isin(dishonest_remove_common)]
    common_honest_df = common_honest_df[~common_honest_df.set_index(['Article ID', 'Question']).index.isin(dishonest_remove_common)]
else:
    sample_size = min(dishonest_distinct_remaining, len(common_honest_df.groupby(['Question', 'Article ID'])))
    honest_sample = common_honest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    dishonest_sample = distinct_dishonest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    df.loc[extract_correct_index(dishonest_sample), sample_column_name] = True
    df.loc[extract_correct_index(honest_sample), sample_column_name] = True
    honest_remove_common = set(honest_sample[['Article ID', 'Question']].itertuples(index=False))
    common_dishonest_df = common_dishonest_df[~common_dishonest_df.set_index(['Article ID', 'Question']).index.isin(honest_remove_common)]
    common_honest_df = common_honest_df[~common_honest_df.set_index(['Article ID', 'Question']).index.isin(honest_remove_common)]
# Remaining independent samples from common_honest_df
if len(common_honest_df) or len(common_dishonest_df) > 0:
    sample_size = min(len(common_honest_df.groupby(['Question', 'Article ID'])), len(common_dishonest_df.groupby(['Question', 'Article ID'])))
    honest_sample = common_honest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    dishonest_sample = common_dishonest_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(sample_size))
    df.loc[extract_correct_index(honest_sample), sample_column_name] = True
    df.loc[extract_correct_index(dishonest_sample), sample_column_name] = True
return df

# Run the sampling to balance the consultancies
judgments_online = balance_consultancies(judgments_online, 'Human Consultancy', random_state = 123)
judgments_online = balance_consultancies(judgments_online, 'AI Consultancy', random_state = 123)
# Create one sample column for easier indexing, create mask
sample_columns = [col for col in judgments_online.columns if 'Sample' in col]
judgments_online['Sample'] = judgments_online[sample_columns].any(axis=1)
consultancy_balanced = (~judgments_online['Setting'].str.contains('Consultancy', case=False, na=False))

#print(f'Accuracy after balancing consultancies:\n{judgments_online[consultancy_balanced].groupby(["Fin

#from statsmodels.stats.proportion import proportions_ztest

def run_experiment(judgments_online):
    judgments_online['Sample'] = False
    judgments_online = balance_consultancies(judgments_online, 'Human Consultancy')
    judgments_online = balance_consultancies(judgments_online, 'AI Consultancy')
    sample_columns = [col for col in judgments_online.columns if 'Sample' in col]
    judgments_online['Sample'] = judgments_online[sample_columns].any(axis=1)

```

```

consultancy_balanced = (~judgments_online['Setting'].str.contains('Consultancy', case=False, na=False))
result = judgments_online[consultancy_balanced].groupby(["Final_Setting"])["Final_Accuracy"].agg(Proportions)
return result

# Number of iterations
num_iterations = 1000

# Store results from each iteration
results = []
p_vals = []
# Run the experiment multiple times
for _ in range(num_iterations):
    # result = run_experiment(judgments_online.copy()) # Use a copy to ensure original data remains unchanged
    results.append(result)
    # Run the proportions test
    group_human_debate = result.loc['Human Debate']
    group_human_consultancy = result.loc['Human Consultancy']
    count = [group_human_debate.Proportion_True * group_human_debate.Total_Count, group_human_consultancy.Proportion_True * group_human_consultancy.Total_Count]
    nobs = [group_human_debate.Total_Count, group_human_consultancy.Total_Count]
    z_stat, p_val = proportions_ztest(count, nobs)
    p_vals.append(p_val)

# Calculate the average of the results
average_result = pd.concat(results).groupby(level=0).mean()

print(f'\nAverage accuracy after {num_iterations} iterations: \n{average_result}')

print(f'pval mean: {np.mean(p_vals)}')

```

## Balance debates

```

def balance_debates(df, sample_setting, random_state):
    debates_df = df[df['Setting'].str.contains(sample_setting, na=False)]
    sample_column_name = f'{sample_setting} Sample'
    df[sample_column_name] = False
    def extract_correct_index(sample_df):
        if isinstance(sample_df.index, pd.MultiIndex):
            return sample_df.index.get_level_values(2)
        else:
            return sample_df.index
    # Get distinct consultancies
    sample_size = len(debates_df.groupby(['Question', 'Article ID']))
    sample_debates = debates_df.groupby(['Question', 'Article ID']).apply(lambda x: x.sample(1, random_state=random_state))
    df.loc[extract_correct_index(sample_debates), sample_column_name] = True
    return df

# Run the sampling to balance the consultancies
judgments_online = balance_debates(judgments_online, 'Human Debate', random_state = 123)
judgments_online = balance_debates(judgments_online, 'AI Debate', random_state = 123)
# Create one sample column for easier indexing, create mask
sample_columns = [col for col in judgments_online.columns if 'Sample' in col]

```



```

consultancy_sample_columns = [col for col in judgments_online.columns if 'Consultancy Sample' in col]

judgments_online['Sample'] = judgments_online[sample_columns].any(axis=1)
judgments_online['Consultancy Sample'] = judgments_online[sample_columns].any(axis=1)

consultancy_balanced = (~judgments_online['Setting'].str.contains('Consultancy', case=False, na=False))

print(f'Accuracy after balancing consultancies:\n{judgments_online.groupby(["Final_Setting"])["Sample"]

## Accuracy after balancing consultancies:
## Final_Setting      Sample
## AI Consultancy      True      76
##                    False      17
## AI Debate            True      75
##                    False      12
## Human Consultancy   True      82
##                    False      25
## Human Debate        True     107
##                    False      48
## Name: Sample, dtype: int64

```

## Question weights

```

def question_weights(data, columns, weight_column_name, consultancy_sample=None, debate_sample=None):
    # 0. Make a copy of the original data for weight calculations
    working_data = data.copy()
    # 0.1. Custom filtering based on the 'Setting' column
    consultancy_condition = working_data['Setting'].str.contains('Consultancy', case=False, na=False)
    debate_condition = ~consultancy_condition
    if consultancy_sample is not None:
        consultancy_condition &= (working_data['Sample'] == consultancy_sample)
    if debate_sample is not None: # uncomment if we want to sample debates
        debate_condition &= (working_data['Sample'] == debate_sample)
    combined_mask = consultancy_condition | debate_condition
    working_data = working_data[combined_mask]
    # 1. Calculate the frequency of each question in the dataset
    question_frequency = working_data.groupby(columns).size()
    # 2. Invert the frequency to get the weight for each question
    question_weights = 1 / question_frequency
    # 3. Normalize the weights
    #question_weights = question_weights / question_weights.sum() * len(question_weights)
    # 4. Assign the calculated weights to the original data and fill missing values with 0
    data.loc[combined_mask, weight_column_name] = data[combined_mask].set_index(columns).index.map(question_weights)
    data[weight_column_name].fillna(0, inplace=True)
    return data

judgments_online = question_weights(
    data=judgments_online,
    columns=['Article ID', 'Question'],
    weight_column_name='initial_question_weights'
)

```

```

judgments_online = question_weights(
    data=judgments_online,
    columns=['Article ID', 'Question', 'Final_Setting'],
    weight_column_name='initial_question_weights_grouped_setting'
)

def print_weight_summary_by_setting(df, weight_column, consultancy_sample=None):
    consultancy_condition = df['Setting'].str.contains('Consultancy', case=False, na=False)
    if consultancy_sample is not None:
        consultancy_condition &= (df['Sample'] == consultancy_sample)
    for setting in df['Setting'].unique():
        total_weight = df[df['Setting'] == setting][weight_column].sum()
        print(f"Total {weight_column} for {setting}: {total_weight:.2f}")
    print("\n")

print('Unsampled (initial) weights, by group setting')

## Unsampled (initial) weights, by group setting

print_weight_summary_by_setting(judgments_online, 'initial_question_weights_grouped_setting')

## Total initial_question_weights_grouped_setting for AI Consultancy Dishonest: 32.50
## Total initial_question_weights_grouped_setting for Human Debate: 107.00
## Total initial_question_weights_grouped_setting for AI Debate: 75.00
## Total initial_question_weights_grouped_setting for Human Consultancy Dishonest: 34.67
## Total initial_question_weights_grouped_setting for Human Consultancy Honest: 26.33
## Total initial_question_weights_grouped_setting for AI Consultancy Honest: 49.50

# Recalculate weights for balanced consultancies, all debates
judgments_online = question_weights(
    data=judgments_online,
    columns=['Article ID', 'Question'],
    weight_column_name='sampled_consultancies_all_debates_weights',
    consultancy_sample=True
)
judgments_online = question_weights(
    data=judgments_online,
    columns=['Article ID', 'Question', 'Final_Setting'],
    weight_column_name='sampled_consultancies_all_debates_weights_grouped_setting',
    consultancy_sample=True
)
judgments_online = question_weights(
    data=judgments_online,
    columns=['Article ID', 'Question', 'Setting'],
    weight_column_name='sampled_consultancies_all_debates_weights_setting',
    consultancy_sample=True
)
print('Consultancy balanced weights, by no/yes group setting')

## Consultancy balanced weights, by no/yes group setting

```

```

print_weight_summary_by_setting(judgments_online[consultancy_balanced], 'sampled_consultancies_all_deba

## Total sampled_consultancies_all_debates_weights for AI Consultancy Dishonest: 28.07
## Total sampled_consultancies_all_debates_weights for Human Debate: 82.48
## Total sampled_consultancies_all_debates_weights for AI Debate: 66.52
## Total sampled_consultancies_all_debates_weights for Human Consultancy Honest: 16.52
## Total sampled_consultancies_all_debates_weights for AI Consultancy Honest: 36.42
## Total sampled_consultancies_all_debates_weights for Human Consultancy Dishonest: 16.00

print_weight_summary_by_setting(judgments_online[consultancy_balanced], 'sampled_consultancies_all_deba

## Total sampled_consultancies_all_debates_weights_grouped_setting for AI Consultancy Dishonest: 38.00
## Total sampled_consultancies_all_debates_weights_grouped_setting for Human Debate: 107.00
## Total sampled_consultancies_all_debates_weights_grouped_setting for AI Debate: 75.00
## Total sampled_consultancies_all_debates_weights_grouped_setting for Human Consultancy Honest: 30.50
## Total sampled_consultancies_all_debates_weights_grouped_setting for AI Consultancy Honest: 38.00
## Total sampled_consultancies_all_debates_weights_grouped_setting for Human Consultancy Dishonest: 30.

print_weight_summary_by_setting(judgments_online[consultancy_balanced], 'sampled_consultancies_all_deba

## Total sampled_consultancies_all_debates_weights_setting for AI Consultancy Dishonest: 38.00
## Total sampled_consultancies_all_debates_weights_setting for Human Debate: 107.00
## Total sampled_consultancies_all_debates_weights_setting for AI Debate: 75.00
## Total sampled_consultancies_all_debates_weights_setting for Human Consultancy Honest: 41.00
## Total sampled_consultancies_all_debates_weights_setting for AI Consultancy Honest: 38.00
## Total sampled_consultancies_all_debates_weights_setting for Human Consultancy Dishonest: 41.00

judgments_online = question_weights(
  data=judgments_online,
  columns=['Article ID', 'Question', 'Final_Setting'],
  weight_column_name='sampled_consultancies_debates_weights_grouped_setting',
  consultancy_sample=True,
  debate_sample=True
)
judgments_online = question_weights(
  data=judgments_online,
  columns=['Article ID', 'Question'],
  weight_column_name='sampled_consultancies_debates_weights',
  consultancy_sample=True,
  debate_sample=True
)

```

Note: we are not balancing *between settings*, and some of the counts of the debate settings are on the same questions

## Load into R environment

```

set.seed(123)
judgments <- py$judgments
judgments_online <- py$judgments_online
# Convert the Accuracy column to a factor for better plotting
judgments_online$Final_Accuracy_char <- as.logical.factor(as.character(judgments_online$Final_Accuracy))
judgments_online$Participant <- as.factor(judgments_online$Participant)
judgments_online$Setting <- as.factor(judgments_online$Setting)

subset_dishonest <- judgments_online[judgments_online$`Human Consultancy Sample` == TRUE & judgments_onlin
subset_honest <- judgments_online[judgments_online$`Human Consultancy Sample` == TRUE & judgments_onlin
table(subset_dishonest$sampled_consultancies_all_debates_weights_grouped_setting, subset_dishonest$Final_Accu

```

```

##
##      FALSE TRUE
## 0.5      11   10
## 1         7   13

```

```

table(subset_honest$sampled_consultancies_all_debates_weights_grouped_setting, subset_honest$Final_Accu

```

```

##
##      FALSE TRUE
## 0.5         5   16
## 1          1   19

```

```

table(subset_dishonest$sampled_consultancies_all_debates_weights_grouped_setting)

```

```

##
## 0.5    1
## 21    20

```

```

table(subset_honest$sampled_consultancies_all_debates_weights_grouped_setting)

```

```

##
## 0.5    1
## 21    20

```

```

subset_human_consultancies <- judgments_online[judgments_online$`Human Consultancy Sample` == TRUE & ju
table(subset_human_consultancies$sampled_consultancies_all_debates_weights_grouped_setting, subset_huma

```

```

##
##      FALSE TRUE
## 0.5      16   26
## 1         8   32

```

```

table(judgments_online$Final_Setting, judgments_online$sampled_consultancies_debates_weights_grouped_se

```

```
##
##           0 0.5  1
## AI Consultancy 17  0 76
## AI Debate      12  0 75
## Human Consultancy 25 42 40
## Human Debate   48  0 107

table(judgments_online$Final_Setting, judgments_online$sampled_consultancies_debates_weights)
```

```
##
##           0 0.2 0.25 0.3333333333333333 0.5  1
## AI Consultancy 17  1  9                4  1 61
## AI Debate      12  1  9                3  1 61
## Human Consultancy 25  2  9            32 32  7
## Human Debate   48  1  9            15 20 62
```

## Results

### Accuracy

#### Difference in proportions

```
acc_diff_test <- function(design, Setting){
  print(design)
  freq_table <- svytable(~Final_Setting+Final_Accuracy, design)
  print(freq_table)
  chisq_result <- svychisq(~Final_Setting+Final_Accuracy, design, statistic = "Chisq")
  print(chisq_result)
  pairwise_result <- pairwise.prop.test(freq_table, p.adjust.method="none", alternative="two.sided")
  print(pairwise_result)
}

print("Really raw")
```

```
## [1] "Really raw"
```

```
acc_diff_test(svydesign(ids = ~1, data = judgments))
```

```
## Warning in svydesign.default(ids = ~1, data = judgments): No weights or
## probabilities supplied, assuming equal probability
```

```
## Independent Sampling design (with replacement)
## svydesign(ids = ~1, data = judgments)
##           Final_Accuracy
## Final_Setting FALSE TRUE
## AI Consultancy 19  77
## AI Debate      20  72
## Human Consultancy 34  87
## Human Debate   50 327
```

```

##
## Pearson's X^2: Rao & Scott adjustment
##
## data: svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 15.218, df = 3, p-value = 0.001657
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data: freq_table
##
##           AI Consultancy AI Debate Human Consultancy
## AI Debate      0.88133      -      -
## Human Consultancy 0.20924      0.36922      -
## Human Debate      0.14538      0.05977      0.00026
##
## P value adjustment method: none

print("Raw")

## [1] "Raw"

acc_diff_test(svydesign(ids = ~1, data = judgments_online))

## Warning in svydesign.default(ids = ~1, data = judgments_online): No weights or
## probabilities supplied, assuming equal probability

## Independent Sampling design (with replacement)
## print(design)
##           Final_Accuracy
## Final_Setting  FALSE TRUE
## AI Consultancy      18   75
## AI Debate           19   68
## Human Consultancy    32   75
## Human Debate        25  130
##
## Pearson's X^2: Rao & Scott adjustment
##
## data: svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 7.4336, df = 3, p-value = 0.05973
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data: freq_table
##
##           AI Consultancy AI Debate Human Consultancy
## AI Debate      0.820      -      -
## Human Consultancy 0.120      0.269      -
## Human Debate      0.634      0.352      0.012
##
## P value adjustment method: none

```

```
print("Balanced consultancies")
```

```
## [1] "Balanced consultancies"
```

```
acc_diff_test(svydesign(ids = ~1, data = subset(judgments_online, `Consultancy Sample` == TRUE | !grepl
```

```
## Warning in svydesign.default(ids = ~1, data = subset(judgments_online,
## 'Consultancy Sample' == : No weights or probabilities supplied, assuming equal
## probability
```

```
## Independent Sampling design (with replacement)
```

```
## print(design)
```

```
##               Final_Accuracy
```

```
## Final_Setting  FALSE TRUE
```

```
##   AI Consultancy    15   61
```

```
##   AI Debate        19   68
```

```
##   Human Consultancy 24   58
```

```
##   Human Debate     25  130
```

```
##
```

```
## Pearson's X^2: Rao & Scott adjustment
```

```
##
```

```
## data: svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
```

```
## X-squared = 5.7412, df = 3, p-value = 0.1257
```

```
##
```

```
##
```

```
## Pairwise comparisons using Pairwise comparison of proportions
```

```
##
```

```
## data: freq_table
```

```
##
```

```
##               AI Consultancy AI Debate Human Consultancy
```

```
## AI Debate        0.892      -      -
```

```
## Human Consultancy 0.229      0.352  -
```

```
## Human Debate     0.620      0.352  0.027
```

```
##
```

```
## P value adjustment method: none
```

```
print("Balanced consultancies, question weights (grouped settings)")
```

```
## [1] "Balanced consultancies, question weights (grouped settings)"
```

```
acc_diff_test(svydesign(ids = ~1, data = subset(judgments_online, `Consultancy Sample` == TRUE | !grepl
```

```
## Independent Sampling design (with replacement)
```

```
## print(design)
```

```
##               Final_Accuracy
```

```
## Final_Setting  FALSE TRUE
```

```
##   AI Consultancy    15.0 61.0
```

```
##   AI Debate        15.5 59.5
```

```
##   Human Consultancy 16.0 45.0
```

```
##   Human Debate     16.5 90.5
```

```

##
## Pearson's X^2: Rao & Scott adjustment
##
## data: svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 3.6629, df = 3, p-value = 0.3338
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data: freq_table
##
##           AI Consultancy AI Debate Human Consultancy
## AI Debate           1.00         -          -
## Human Consultancy 0.49           0.58         -
## Human Debate       0.57           0.47        0.13
##
## P value adjustment method: none

print("Balanced consultancies sampled debates, question weights (grouped settings)")

## [1] "Balanced consultancies sampled debates, question weights (grouped settings)"

acc_diff_test(svydesign(ids = ~1, data = subset(judgments_online, `Sample` == TRUE), weights = ~sampled,

## Independent Sampling design (with replacement)
## print(design)
##           Final_Accuracy
## Final_Setting    FALSE  TRUE
## AI Consultancy      15    61
## AI Debate           15    60
## Human Consultancy   16    45
## Human Debate        16    91
##
## Pearson's X^2: Rao & Scott adjustment
##
## data: svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 3.4008, df = 3, p-value = 0.3378
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data: freq_table
##
##           AI Consultancy AI Debate Human Consultancy
## AI Debate           1.00         -          -
## Human Consultancy 0.49           0.51         -
## Human Debate       0.52           0.49        0.11
##
## P value adjustment method: none

acc_diff_test(svydesign(ids = ~1, data = judgments_online, weights = ~sampled_consultancies_debates_weig

```



```

## Independent Sampling design (with replacement)
## print(design)
##               Final_Accuracy
## Final_Setting   FALSE TRUE
##   AI Consultancy      15   61
##   AI Debate           15   60
##   Human Consultancy   16   45
##   Human Debate       16   91
##
## Pearson's X^2: Rao & Scott adjustment
##
## data:  svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 4.421, df = 3, p-value = 0.3375
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data:  freq_table
##
##               AI Consultancy AI Debate Human Consultancy
## AI Debate           1.00         -         -
## Human Consultancy  0.49         0.51         -
## Human Debate       0.52         0.49         0.11
##
## P value adjustment method: none

design = svydesign(ids = ~1, data = subset(judgments_online, `Human Consultancy Sample` == TRUE | !grep.
acc_diff_test(design)

## Independent Sampling design (with replacement)
## svydesign(ids = ~1, data = subset(judgments_online, 'Human Consultancy Sample' ==
##   TRUE | !grepl("Consultancy", Final_Setting) & !grepl("AI",
##   Final_Setting)), weights = ~sampled_consultancies_all_debates_weights_grouped_setting)
##               Final_Accuracy
## Final_Setting   FALSE TRUE
##   Human Consultancy  16.0 45.0
##   Human Debate      16.5 90.5
##
## Pearson's X^2: Rao & Scott adjustment
##
## data:  svychisq(~Final_Setting + Final_Accuracy, design, statistic = "Chisq")
## X-squared = 4.104, df = 1, p-value = 0.05155
##
##
## Pairwise comparisons using Pairwise comparison of proportions
##
## data:  freq_table
##
##               Human Consultancy
## Human Debate 0.13
##
## P value adjustment method: none

```

```
final_table <- svytable(~Final_Setting+Final_Accuracy,
                      design = svydesign(ids = ~1,
                                         data = subset(judgments_online, `Consultancy Sample` == TRUE),
                                         weights = ~sampled_consultancies_all_debates_weights_grouped),
                      final_table)
```

```
##               Final_Accuracy
## Final_Setting  FALSE TRUE
##   AI Consultancy   15.0 61.0
##   AI Debate       15.5 59.5
##   Human Consultancy 16.0 45.0
##   Human Debate    16.5 90.5
```

```
# Add accuracy
final_table <- cbind(final_table, Accuracy = (final_table[,2] / (final_table[,1]+final_table[,2]))*100)
# Calculate the difference in accuracy for each row compared to "Human Debate"
difference_with_debate <- final_table[, "Accuracy"] - final_table["Human Debate", "Accuracy"]
# Bind the difference column to the final_table
final_table <- cbind(final_table, difference_with_debate)

# Loop through each setting
ci_lowers <- c()
ci_uppers <- c()
p_values <- c()
# Loop through each setting
for (setting in rownames(final_table)) {
  # Use prop.test to compare the setting's accuracy with "Human Debate"
  results <- prop.test(c(final_table["Human Debate", "TRUE"], final_table[setting, "TRUE"]), c((final_table["Human Debate", "TRUE"] + final_table[setting, "TRUE"])))

  # Extract the confidence interval and store it as a string in the format "lower - upper"
  ci_lower <- results$conf.int[1] * 100 # Multiply by 100 to convert to percentage
  ci_upper <- results$conf.int[2] * 100 # Multiply by 100 to convert to percentage
  ci_lowers <- c(ci_lowers, ci_lower)
  ci_uppers <- c(ci_uppers, ci_upper)
  p_values <- c(p_values, results$p.value)
}
final_table <- cbind(final_table, ci_lowers, ci_uppers, p_values)
final_table
```

```
##               FALSE TRUE Accuracy difference_with_debate ci_lowers
## AI Consultancy   15.0 61.0 80.26316          -4.316281 -8.073724
## AI Debate       15.5 59.5 79.33333          -5.246106 -7.324725
## Human Consultancy 16.0 45.0 73.77049         -10.808947 -3.465654
## Human Debate    16.5 90.5 84.57944           0.000000 -9.677288
##               ci_uppers p_values
## AI Consultancy  16.706287 0.5730868
## AI Debate      17.816936 0.4731832
## Human Consultancy 25.083549 0.1329563
## Human Debate    9.677288 1.0000000
```

```
# Display the updated table using knitr::kable
knitr::kable(final_table, booktab = TRUE, digits = c(rep(1,6),3),
             col.names = c("# Incorrect (weighted)", "# Correct (weighted)", "Accuracy", "Difference", "CI Lower", "CI Upper", "P-value"))
```

	# Incorrect (weighted)	# Correct (weighted)	Accuracy	Difference	95% CI Lower Limit	95% CI Upper Limit	p- value
AI Consultancy	15.0	61.0	80.3	-4.3	-8.1	16.7	0.573
AI Debate	15.5	59.5	79.3	-5.2	-7.3	17.8	0.473
Human Consultancy	16.0	45.0	73.8	-10.8	-3.5	25.1	0.133
Human Debate	16.5	90.5	84.6	0.0	-9.7	9.7	1.000

```
human_only <- subset(judgments_online, `Human Consultancy Sample` == TRUE | !grepl("Consultancy", Final.
human_only$Setting <- droplevels(human_only$Setting)
table(human_only$Setting)
```

```
##
## Human Consultancy Dishonest Human Consultancy Honest
##                41                41
##      Human Debate
##                155
```

```
final_table <- svytable(~Setting+Final_Accuracy,
                        design = svydesign(ids = ~1,
                                           data = human_only,
                                           weights = ~sampled_consultancies_all_debates_weights_setting)
final_table
```

```
##
## Setting                Final_Accuracy
## Setting                FALSE TRUE
## Human Consultancy Dishonest 18.0 23.0
## Human Consultancy Honest    6.0 35.0
## Human Debate                16.5 90.5
```

```
# Add accuracy
final_table <- cbind(final_table, Accuracy = (final_table[,2] / (final_table[,1]+final_table[,2]))*100)
# Calculate the difference in accuracy for each row compared to "Human Debate"
difference_with_debate <- final_table[, "Accuracy"] - final_table["Human Debate", "Accuracy"]
# Bind the difference column to the final_table
final_table <- cbind(final_table, difference_with_debate)

# Loop through each setting
ci_lowers <- c()
ci_uppers <- c()
p_values <- c()
# Loop through each setting
for (setting in rownames(final_table)) {
  # Use prop.test to compare the setting's accuracy with "Human Debate"
  results <- prop.test(c(final_table["Human Debate", "TRUE"], final_table[setting, "TRUE"]), c((final_t

  # Extract the confidence interval and store it as a string in the format "lower - upper"
  ci_lower <- results$conf.int[1] * 100 # Multiply by 100 to convert to percentage
  ci_upper <- results$conf.int[2] * 100 # Multiply by 100 to convert to percentage
```

```

ci_lowers <- c(ci_lowers, ci_lower)
ci_uppers <- c(ci_uppers, ci_upper)
p_values <- c(p_values, results$p.value)
}
final_table <- cbind(final_table, ci_lowers, ci_uppers, p_values)
final_table

```

```

##               FALSE TRUE Accuracy difference_with_debate
## Human Consultancy Dishonest 18.0 23.0 56.09756          -28.4818783
## Human Consultancy Honest    6.0 35.0 85.36585           0.7864144
## Human Debate               16.5 90.5 84.57944           0.0000000
##               ci_lowers ci_uppers      p_values
## Human Consultancy Dishonest 10.134444 46.829313 0.0005598759
## Human Consultancy Honest   -14.374115 12.801286 1.0000000000
## Human Debate              -9.677288  9.677288 1.0000000000

```

```

# Display the updated table using knitr::kable
knitr::kable(final_table, booktab = TRUE, digits = c(rep(1,6),3),
              col.names = c("# Incorrect (weighted)", "# Correct (weighted)", "Accuracy", "Difference",

```

	# Incorrect (weighted)	# Correct (weighted)	Accuracy	Difference	95% CI Lower Limit	95% CI Upper Limit	p- value
Human Consultancy Dishonest	18.0	23.0	56.1	-28.5	10.1	46.8	0.001
Human Consultancy Honest	6.0	35.0	85.4	0.8	-14.4	12.8	1.000
Human Debate	16.5	90.5	84.6	0.0	-9.7	9.7	1.000

```

prop_table <- svytable(~Final_Setting+Final_Accuracy, design = svydesign(ids = ~1, data = subset(judgmen
#prop_table <- svytable(~Final_Setting+Final_Accuracy, design = svydesign(ids = ~1, data = subset(judgm
prop_table

```

```

##               Final_Accuracy
## Final_Setting      FALSE      TRUE
##   Human Consultancy 9.866667 22.650000
##   Human Debate     10.850000 71.633333

```

```

print(prop.test(c(prop_table["Human Consultancy", "TRUE"], prop_table["Human Debate", "TRUE"]), c(prop_table["Human Consultancy", "FALSE"], prop_table["Human Debate", "FALSE"])))

```

```

##
## 2-sample test for equality of proportions with continuity correction
##
## data:  c(prop_table["Human Consultancy", "TRUE"], prop_table["Human Debate", "TRUE"]) out of c(prop_table["Human Consultancy", "FALSE"], prop_table["Human Debate", "FALSE"])
## X-squared = 3.5746, df = 1, p-value = 0.05867
## alternative hypothesis: two.sided
## 95 percent confidence interval:

```

```
## -0.36737199 0.02358717
## sample estimates:
## prop 1 prop 2
## 0.6965659 0.8684583
```

## Logistic regression

```
#judgments_online$Final_Setting <- relevel(judgments_online$Final_Setting, ref = "Human Debate")
modell1 <- glm(Final_Accuracy ~ relevel(factor(Final_Setting), 'Human Debate'), family = 'binomial', data = judgments_online)
summary(modell1)
```

```
##
## Call:
## glm(formula = Final_Accuracy ~ relevel(factor(Final_Setting),
## "Human Debate"), family = "binomial", data = judgments_online)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -1.9103 0.5931 0.5931 0.7020 0.8430
##
## Coefficients:
## Estimate
## (Intercept) 1.6487
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy -0.2215
## relevel(factor(Final_Setting), "Human Debate")AI Debate -0.3736
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy -0.7969
## Std. Error
## (Intercept) 0.2184
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy 0.3414
## relevel(factor(Final_Setting), "Human Debate")AI Debate 0.3392
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy 0.3038
## z value
## (Intercept) 7.549
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy -0.649
## relevel(factor(Final_Setting), "Human Debate")AI Debate -1.102
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy -2.623
## Pr(>|z|)
## (Intercept) 0.0000000000000438
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy 0.51644
## relevel(factor(Final_Setting), "Human Debate")AI Debate 0.27067
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy 0.00871
##
## (Intercept) ***
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy
## relevel(factor(Final_Setting), "Human Debate")AI Debate
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
## Null deviance: 457.45 on 441 degrees of freedom
## Residual deviance: 450.23 on 438 degrees of freedom
## AIC: 458.23
##
## Number of Fisher Scoring iterations: 4
```

```
table(model1$fitted.values > 0.5)
```

```
##
## TRUE
## 442
```

```
table(judgments_online$Final_Accuracy)
```

```
##
## FALSE TRUE
## 94 348
```

```
model2 <- glm(Final_Accuracy ~ relevel(factor(Participant), 'Aliyaah Toussaint') + relevel(factor(Final_Setting), 'Human Debate'), family = "binomial", data = judgments_online)
```

```
summary(model2)
```

```
##
## Call:
## glm(formula = Final_Accuracy ~ relevel(factor(Participant), "Aliyaah Toussaint") +
##      relevel(factor(Final_Setting), "Human Debate"), family = "binomial",
##      data = judgments_online)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.2162   0.4597   0.5617   0.7213   1.0602
##
## Coefficients:
## (Intercept)                                Estimate
## relevel(factor(Participant), "Aliyaah Toussaint")Adelle Fernando -0.79600
## relevel(factor(Participant), "Aliyaah Toussaint")Anuj Jain        -0.89691
## relevel(factor(Participant), "Aliyaah Toussaint")David Rein        -0.43887
## relevel(factor(Participant), "Aliyaah Toussaint")Emmanuel Makinde -17.76039
## relevel(factor(Participant), "Aliyaah Toussaint")Ethan Rosen       -0.24841
## relevel(factor(Participant), "Aliyaah Toussaint")Jackson Petty     -0.55820
## relevel(factor(Participant), "Aliyaah Toussaint")Jessica Li        -0.16347
## relevel(factor(Participant), "Aliyaah Toussaint")Julian Michael    -0.08063
## relevel(factor(Participant), "Aliyaah Toussaint")Julien Dirani     13.37175
## relevel(factor(Participant), "Aliyaah Toussaint")Max Layden        13.37175
## relevel(factor(Participant), "Aliyaah Toussaint")Noor Mirza-Rashid -1.27803
## relevel(factor(Participant), "Aliyaah Toussaint")Reeya Kansra      -0.96379
## relevel(factor(Participant), "Aliyaah Toussaint")Salsabila Mahdi    -0.17942
## relevel(factor(Participant), "Aliyaah Toussaint")Sam Jin           -0.01031
## relevel(factor(Participant), "Aliyaah Toussaint")Sean Wang         0.17177
## relevel(factor(Participant), "Aliyaah Toussaint")Shlomo Kofman     -1.13135
## relevel(factor(Participant), "Aliyaah Toussaint")Shreeram Modi     -1.16733
```

```

## relevel(factor(Participant), "Aliyaah Toussaint")Vishakh Padmakumar -0.40256
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy -0.27193
## relevel(factor(Final_Setting), "Human Debate")AI Debate -0.42241
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy -0.74485
## Std. Error
## (Intercept) 0.49853
## relevel(factor(Participant), "Aliyaah Toussaint")Adelle Fernando 0.63661
## relevel(factor(Participant), "Aliyaah Toussaint")Anuj Jain 0.53893
## relevel(factor(Participant), "Aliyaah Toussaint")David Rein 0.77471
## relevel(factor(Participant), "Aliyaah Toussaint")Emmanuel Makinde 1455.39762
## relevel(factor(Participant), "Aliyaah Toussaint")Ethan Rosen 1.17957
## relevel(factor(Participant), "Aliyaah Toussaint")Jackson Petty 0.66085
## relevel(factor(Participant), "Aliyaah Toussaint")Jessica Li 0.64365
## relevel(factor(Participant), "Aliyaah Toussaint")Julian Michael 0.75783
## relevel(factor(Participant), "Aliyaah Toussaint")Julien Dirani 1029.12159
## relevel(factor(Participant), "Aliyaah Toussaint")Max Layden 1029.12159
## relevel(factor(Participant), "Aliyaah Toussaint")Noor Mirza-Rashid 0.97393
## relevel(factor(Participant), "Aliyaah Toussaint")Reeya Kansra 0.58143
## relevel(factor(Participant), "Aliyaah Toussaint")Salsabila Mahdi 0.90289
## relevel(factor(Participant), "Aliyaah Toussaint")Sam Jin 0.56587
## relevel(factor(Participant), "Aliyaah Toussaint")Sean Wang 0.67879
## relevel(factor(Participant), "Aliyaah Toussaint")Shlomo Kofman 0.50759
## relevel(factor(Participant), "Aliyaah Toussaint")Shreeram Modi 0.63420
## relevel(factor(Participant), "Aliyaah Toussaint")Vishakh Padmakumar 1.18962
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy 0.39222
## relevel(factor(Final_Setting), "Human Debate")AI Debate 0.39204
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy 0.36432
## z value
## (Intercept) 4.402
## relevel(factor(Participant), "Aliyaah Toussaint")Adelle Fernando -1.250
## relevel(factor(Participant), "Aliyaah Toussaint")Anuj Jain -1.664
## relevel(factor(Participant), "Aliyaah Toussaint")David Rein -0.566
## relevel(factor(Participant), "Aliyaah Toussaint")Emmanuel Makinde -0.012
## relevel(factor(Participant), "Aliyaah Toussaint")Ethan Rosen -0.211
## relevel(factor(Participant), "Aliyaah Toussaint")Jackson Petty -0.845
## relevel(factor(Participant), "Aliyaah Toussaint")Jessica Li -0.254
## relevel(factor(Participant), "Aliyaah Toussaint")Julian Michael -0.106
## relevel(factor(Participant), "Aliyaah Toussaint")Julien Dirani 0.013
## relevel(factor(Participant), "Aliyaah Toussaint")Max Layden 0.013
## relevel(factor(Participant), "Aliyaah Toussaint")Noor Mirza-Rashid -1.312
## relevel(factor(Participant), "Aliyaah Toussaint")Reeya Kansra -1.658
## relevel(factor(Participant), "Aliyaah Toussaint")Salsabila Mahdi -0.199
## relevel(factor(Participant), "Aliyaah Toussaint")Sam Jin -0.018
## relevel(factor(Participant), "Aliyaah Toussaint")Sean Wang 0.253
## relevel(factor(Participant), "Aliyaah Toussaint")Shlomo Kofman -2.229
## relevel(factor(Participant), "Aliyaah Toussaint")Shreeram Modi -1.841
## relevel(factor(Participant), "Aliyaah Toussaint")Vishakh Padmakumar -0.338
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy -0.693
## relevel(factor(Final_Setting), "Human Debate")AI Debate -1.077
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy -2.045
## Pr(>|z|)
## (Intercept) 0.0000107
## relevel(factor(Participant), "Aliyaah Toussaint")Adelle Fernando 0.2112
## relevel(factor(Participant), "Aliyaah Toussaint")Anuj Jain 0.0961

```

```

## relevel(factor(Participant), "Aliyaah Toussaint")David Rein 0.5711
## relevel(factor(Participant), "Aliyaah Toussaint")Emmanuel Makinde 0.9903
## relevel(factor(Participant), "Aliyaah Toussaint")Ethan Rosen 0.8332
## relevel(factor(Participant), "Aliyaah Toussaint")Jackson Petty 0.3983
## relevel(factor(Participant), "Aliyaah Toussaint")Jessica Li 0.7995
## relevel(factor(Participant), "Aliyaah Toussaint")Julian Michael 0.9153
## relevel(factor(Participant), "Aliyaah Toussaint")Julien Dirani 0.9896
## relevel(factor(Participant), "Aliyaah Toussaint")Max Layden 0.9896
## relevel(factor(Participant), "Aliyaah Toussaint")Noor Mirza-Rashid 0.1894
## relevel(factor(Participant), "Aliyaah Toussaint")Reeya Kansra 0.0974
## relevel(factor(Participant), "Aliyaah Toussaint")Salsabila Mahdi 0.8425
## relevel(factor(Participant), "Aliyaah Toussaint")Sam Jin 0.9855
## relevel(factor(Participant), "Aliyaah Toussaint")Sean Wang 0.8002
## relevel(factor(Participant), "Aliyaah Toussaint")Shlomo Kofman 0.0258
## relevel(factor(Participant), "Aliyaah Toussaint")Shreeram Modi 0.0657
## relevel(factor(Participant), "Aliyaah Toussaint")Vishakh Padmakumar 0.7351
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy 0.4881
## relevel(factor(Final_Setting), "Human Debate")AI Debate 0.2813
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy 0.0409
##
## (Intercept) ***
## relevel(factor(Participant), "Aliyaah Toussaint")Adelle Fernando
## relevel(factor(Participant), "Aliyaah Toussaint")Anuj Jain .
## relevel(factor(Participant), "Aliyaah Toussaint")David Rein
## relevel(factor(Participant), "Aliyaah Toussaint")Emmanuel Makinde
## relevel(factor(Participant), "Aliyaah Toussaint")Ethan Rosen
## relevel(factor(Participant), "Aliyaah Toussaint")Jackson Petty
## relevel(factor(Participant), "Aliyaah Toussaint")Jessica Li
## relevel(factor(Participant), "Aliyaah Toussaint")Julian Michael
## relevel(factor(Participant), "Aliyaah Toussaint")Julien Dirani
## relevel(factor(Participant), "Aliyaah Toussaint")Max Layden
## relevel(factor(Participant), "Aliyaah Toussaint")Noor Mirza-Rashid
## relevel(factor(Participant), "Aliyaah Toussaint")Reeya Kansra .
## relevel(factor(Participant), "Aliyaah Toussaint")Salsabila Mahdi
## relevel(factor(Participant), "Aliyaah Toussaint")Sam Jin
## relevel(factor(Participant), "Aliyaah Toussaint")Sean Wang
## relevel(factor(Participant), "Aliyaah Toussaint")Shlomo Kofman *
## relevel(factor(Participant), "Aliyaah Toussaint")Shreeram Modi .
## relevel(factor(Participant), "Aliyaah Toussaint")Vishakh Padmakumar
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy
## relevel(factor(Final_Setting), "Human Debate")AI Debate
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 457.45 on 441 degrees of freedom
## Residual deviance: 429.05 on 420 degrees of freedom
## AIC: 473.05
##
## Number of Fisher Scoring iterations: 14

```



## LMER

```
random.intercept.model = lmer(`Final probability correct` ~ (1|Final_Setting),
                              data = judgments, REML = TRUE)
judgments$random.intercept.preds = predict(random.intercept.model)
summary(random.intercept.model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: 'Final probability correct' ~ (1 | Final_Setting)
##      Data: judgments
##
## REML criterion at convergence: 364
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.5652 -0.2013  0.5015  0.5654  0.9255
##
## Random effects:
##   Groups             Name             Variance Std.Dev.
##   Final_Setting (Intercept) 0.00272   0.05215
##   Residual                0.09799   0.31304
## Number of obs: 686, groups:  Final_Setting, 4
##
## Fixed effects:
##              Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  0.75723    0.02948 3.33320   25.68  0.00006 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ranef(random.intercept.model)
```

```
## $Final_Setting
##              (Intercept)
## AI Consultancy    0.002319437
## AI Debate         -0.001131439
## Human Consultancy -0.056960054
## Human Debate       0.055772055
##
## with conditional variances for "Final_Setting"
```

```
ranova(random.intercept.model)
```

```
## ANOVA-like table for random-effects: Single term deletions
##
## Model:
## 'Final probability correct' ~ (1 | Final_Setting)
##              npar logLik    AIC    LRT Df Pr(>Chisq)
## <none>              3 -182.00 370.00
## (1 | Final_Setting)  2 -187.23 378.46 10.456  1  0.001222 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

random.intercept.model = lmer(`Final probability correct` ~ (1|Participant) + (1|Final_Setting),
                             data = judgments, REML = TRUE)
judgments$random.intercept.preds = predict(random.intercept.model)
summary(random.intercept.model)

```

```

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: 'Final probability correct' ~ (1 | Participant) + (1 | Final_Setting)
## Data: judgments
##
## REML criterion at convergence: 357.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.7461 -0.1555  0.4368  0.5996  1.1083
##
## Random effects:
## Groups      Name      Variance Std.Dev.
## Participant (Intercept) 0.002215 0.04707
## Final_Setting (Intercept) 0.002718 0.05213
## Residual      0.095721 0.30939
## Number of obs: 686, groups: Participant, 19; Final_Setting, 4
##
## Fixed effects:
##              Estimate Std. Error    df t value    Pr(>|t|)
## (Intercept)  0.75549    0.03211 4.44845   23.52 0.00000772 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

ranef(random.intercept.model)

```

```

## $Participant
##              (Intercept)
## Adelle Fernando  -0.0231887667
## Aliyaah Toussaint  0.0445495902
## Anuj Jain         -0.0460548530
## David Rein        0.0107246587
## Emmanuel Makinde  -0.0115704647
## Ethan Rosen       -0.0171199427
## Jackson Petty     -0.0051104119
## Jessica Li        -0.0047621455
## Julian Michael    0.0348708056
## Julien Dirani     -0.0008138972
## Max Layden        -0.0038287458
## Noor Mirza-Rashid -0.0117445230
## Reeya Kansra      -0.0261229696
## Salsabila Mahdi   0.0321800144
## Sam Jin           0.0480694982
## Sean Wang         0.0477306783
## Shlomo Kofman     -0.0519667486
## Shreeram Modi     0.0020512016
## Vishakh Padmakumar -0.0178929784

```

```
##
## $Final_Setting
##              (Intercept)
## AI Consultancy    0.0012586597
## AI Debate        -0.0009034629
## Human Consultancy -0.0564188188
## Human Debate      0.0560636219
##
## with conditional variances for "Participant" "Final_Setting"

ranova(random.intercept.model)

## ANOVA-like table for random-effects: Single term deletions
##
## Model:
## 'Final probability correct' ~ (1 | Participant) + (1 | Final_Setting)
##      npar logLik   AIC    LRT Df Pr(>Chisq)
## <none>      4 -178.95 365.9
## (1 | Participant)      3 -182.00 370.0 6.0957  1  0.013551 *
## (1 | Final_Setting)    3 -183.65 373.3 9.4004  1  0.002169 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## BRMS

```
#brm1 <- brm(data = judgments_online,
#             formula = as.numeric(Final_Accuracy) | trials(2) ~ 1 + (1 | Final_Setting),
#             family = binomial("identity"),
#             iter = 2000, warmup = 1000, chains = 4, cores = 4,
#             control = list(adapt_delta = .975, max_treedepth = 20),
#             seed = 190831)
#plot(brm1)
```

## Efficiency

Quotes %, caveats

```
characters = turns.merge(
  debates[["Room name", "Question", "Story length",
           "Untimed annotator context", "Untimed annotator context bins",
           "Setting", "Final_Setting", "Final_Accuracy",
           "Is offline"]],
  how="left",
  on="Room name",
)

# Filtering for specific roles
```

```

characters = characters[characters['Role (honest/dishonest)'].isin(['Honest debater', 'Dishonest debater'])]

# Extracting the spans
def extract_spans(span_str):
    """Extract numerical spans from the given string."""
    if pd.isna(span_str):
        return []
    spans = re.findall(r'<<(\d+)-(\d+)>>', span_str)
    return [(int(start), int(end)) for start, end in spans]

# Merging overlapping spans
def merge_overlapping_spans(span_str):
    if not isinstance(span_str, str):
        return span_str
    spans = extract_spans(span_str)
    if not spans:
        return span_str
    spans.sort(key=lambda x: x[0])
    merged = [spans[0]]
    for current in spans:
        previous = merged[-1]
        if current[0] <= previous[1]:
            upper_bound = max(previous[1], current[1])
            merged[-1] = (previous[0], upper_bound)
        else:
            merged.append(current)
    return ' '.join(f'<<{start}-{end}>>' for start, end in merged)

# Aggregating function to concatenate quote spans
def custom_join(series):
    return ' '.join(filter(lambda x: isinstance(x, str), series))

# Identify questions with more than one setting and filter out the characters dataframe
questions_with_multi_settings = characters.groupby("Question").filter(lambda x: len(x["Setting"].unique()) > 1)
filtered_characters = characters[characters["Question"].isin(questions_with_multi_settings)]

# Aggregating data
aggregates = {
    'Quote length': 'sum',
    'Story length': 'mean',
    'Num previous judging rounds': 'max',
    'Participant quote span': custom_join
}

# Grouping by 'Room name' and aggregating
characters_agg_by_room = filtered_characters.groupby('Room name').agg(aggregates).reset_index()

# Merging the aggregated results with the original data to reintroduce the desired columns
characters_agg = characters_agg_by_room.merge(
    filtered_characters[['Room name', 'Setting', 'Final_Setting', 'Question', 'Untimed annotator context']],
    on='Room name'
)

# Merge overlapping spans after the aggregation

```

```

characters_agg["merged_quote_spans"] = characters_agg["Participant quote span"].apply(merge_overlapping)

# Functions to compute and compare spans across settings
def extract_numbers_from_span(span_str):
    spans = extract_spans(span_str)
    numbers = set()
    for start, end in spans:
        numbers.update(range(int(start), int(end)+1))
    return numbers

def quote_length(span_str):
    spans = extract_spans(span_str)
    numbers = set()
    for start, end in spans:
        numbers.update(range(int(start), int(end)))
    return numbers

characters_agg["quote_length"] = characters_agg["Participant quote span"].apply(lambda row: len(quote_length(row["Participant quote span"])))
#characters_agg["merged_quote_length"] = characters_agg["Participant quote span"].apply(lambda row: len(quote_length(row["Participant quote span"])))
#print(characters_agg["merged_quote_length"][1])
#print((characters_agg["merged_quote_length"]==characters_agg["quote_length"]).value_counts())

#print((characters_agg['quote_length'].fillna(0)/characters_agg['Story length'].fillna(0)).describe())

def convert_to_span_format(numbers):
    sorted_numbers = sorted(list(numbers))
    spans = []
    if sorted_numbers:
        start = sorted_numbers[0]
        end = sorted_numbers[0]
        for num in sorted_numbers[1:]:
            if num == end + 1:
                end = num
            else:
                spans.append((start, end))
                start = end = num
        spans.append((start, end))
    return ' '.join(f'<<{start}-{end}>>' for start, end in spans)

def compute_span_differences(dataframe):
    differences = {}
    for question, group in dataframe.groupby("Question"):
        settings = group["Setting"].unique()
        if len(settings) > 1:
            for i in range(len(settings)):
                for j in range(i+1, len(settings)):
                    setting_1 = settings[i]
                    setting_2 = settings[j]
                    room_1 = group[group["Setting"] == setting_1]["Room name"].values[0]
                    room_2 = group[group["Setting"] == setting_2]["Room name"].values[0]
                    acc_1 = group[group["Setting"] == setting_1]["Final_Accuracy"].values[0]
                    acc_2 = group[group["Setting"] == setting_2]["Final_Accuracy"].values[0]

```

```

        span_str_1 = group[group["Setting"] == setting_1]["merged_quote_spans"].values[0]
        span_str_2 = group[group["Setting"] == setting_2]["merged_quote_spans"].values[0]
        numbers_1 = extract_numbers_from_span(span_str_1)
        numbers_2 = extract_numbers_from_span(span_str_2)
        diff_1 = numbers_1 - numbers_2
        diff_2 = numbers_2 - numbers_1
        key = (question, setting_1, room_1, acc_1, setting_2, room_2, acc_2)
        value = (convert_to_span_format(diff_1), convert_to_span_format(diff_2))
        differences[key] = value

    return differences

span_differences_all = compute_span_differences(characters_agg)

#print(span_differences_all.keys())
#for span in span_differences_all[('Why were Jorgenson and Ganti not put to death?', 'Human Consultancy
# print(len(quote_length(span)))

split_span_differences_with_room = []
# Iterate over the span differences
for (question, setting_1, room_1, acc_1, setting_2, room_2, acc_2), (diff_1, diff_2) in span_differences:
    split_span_differences_with_room.append((question, setting_1, room_1, acc_1, setting_2, room_2, acc_2, diff_1, diff_2))
    split_span_differences_with_room.append((question, setting_2, room_2, acc_2, setting_1, room_1, acc_1, diff_2, diff_1))

# Convert the list to a DataFrame
split_span_df = pd.DataFrame(split_span_differences_with_room, columns=['Question', 'Setting 1', 'Room 1', 'Acc 1', 'Setting 2', 'Room 2', 'Acc 2', 'Span Difference 1', 'Span Difference 2'])

split_span_df["Span Difference Count"] = split_span_df["Span Difference"].apply(lambda x: len(quote_length(x)))
split_span_df["Settings"] = split_span_df["Setting 1"] + " - " + split_span_df["Setting 2"]

# Group by the new 'Settings' column and compute aggregated counts and average of 'Span Difference Count'
grouped_data = split_span_df.groupby("Settings").agg(
    Count=('Span Difference Count', 'size'),
    Average_Span_Difference=('Span Difference Count', 'mean')
).reset_index()

grouped_data

##
## 0      AI Consultancy Dishonest - AI Consultancy Honest ...  Average_Span_Difference
## 1      AI Consultancy Dishonest - AI Debate ... 141.500000
## 2      AI Consultancy Dishonest - Human Consultancy Dishonest ... 169.833333
## 3      AI Consultancy Dishonest - Human Consultancy Honest ... 96.384615
## 4      AI Consultancy Dishonest - Human Debate ... 129.153846
## 5      AI Consultancy Honest - AI Consultancy Dishonest ... 202.916667
## 6      AI Consultancy Honest - AI Debate ... 189.750000
## 7      AI Consultancy Honest - Human Consultancy Dishonest ... 211.333333
## 8      AI Consultancy Honest - Human Consultancy Honest ... 177.416667
## 9      AI Consultancy Honest - Human Debate ... 197.833333
## 10     AI Debate - AI Consultancy Dishonest ... 85.083333
## 11     AI Debate - AI Consultancy Honest ... 65.500000
## 12     AI Debate - Human Consultancy Dishonest ... 94.500000
## 13     AI Debate - Human Consultancy Honest ... 78.000000

```

```

## 14          AI Debate - Human Debate ...      88.062500
## 15 Human Consultancy Dishonest - AI Consultancy D... ... 340.166667
## 16 Human Consultancy Dishonest - AI Consultancy H... ... 315.000000
## 17          Human Consultancy Dishonest - AI Debate ... 404.750000
## 18 Human Consultancy Dishonest - Human Consultanc... ... 334.815789
## 19          Human Consultancy Dishonest - Human Debate ... 300.847826
## 20 Human Consultancy Honest - AI Consultancy Dish... ... 280.692308
## 21 Human Consultancy Honest - AI Consultancy Honest ... 293.333333
## 22          Human Consultancy Honest - AI Debate ... 299.083333
## 23 Human Consultancy Honest - Human Consultancy D... ... 272.763158
## 24          Human Consultancy Honest - Human Debate ... 255.380952
## 25          Human Debate - AI Consultancy Dishonest ... 179.153846
## 26          Human Debate - AI Consultancy Honest ... 201.250000
## 27          Human Debate - AI Debate ... 188.625000
## 28          Human Debate - Human Consultancy Dishonest ... 163.956522
## 29          Human Debate - Human Consultancy Honest ... 147.880952
##
## [30 rows x 3 columns]

```

```

filtered_df = split_span_df[
    (split_span_df["Setting 1"] == "Human Debate") &
    ((split_span_df["Setting 2"] == "Human Consultancy Honest") | (split_span_df["Setting 2"] == "Human
]

print(filtered_df.groupby(['Setting 2', 'Acc_1', 'Acc_2'])['Span Difference Count'].describe())

```

```

##
##          count      mean ...    75%    max
## Setting 2    Acc_1 Acc_2
## Human Consultancy Dishonest False False    5.0  187.200000 ...  275.00  293.0
##                                True      8.0  149.625000 ...  236.25  275.0
##                                True False   16.0  148.687500 ...  182.00  358.0
##                                True      17.0  178.235294 ...  233.00  526.0
## Human Consultancy Honest   False False    4.0  144.750000 ...  257.25  267.0
##                                True      12.0  122.416667 ...  164.75  325.0
##                                True False    4.0  197.000000 ...  224.25  273.0
##                                True      22.0  153.409091 ...  195.00  394.0
##
## [8 rows x 8 columns]

```

```

# Calculate the IQR and bounds for each group in 'Setting 2'
grouped = filtered_df.groupby('Setting 2')['Span Difference Count']

Q1 = grouped.quantile(0.25)
Q3 = grouped.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers based on the computed bounds
filtered_no_outliers = filtered_df[
    (filtered_df['Setting 2'].map(lower_bound) <= filtered_df['Span Difference Count']) &
    (filtered_df['Setting 2'].map(upper_bound) >= filtered_df['Span Difference Count'])
]

```

```
]
```

```
filtered_no_outliers
```

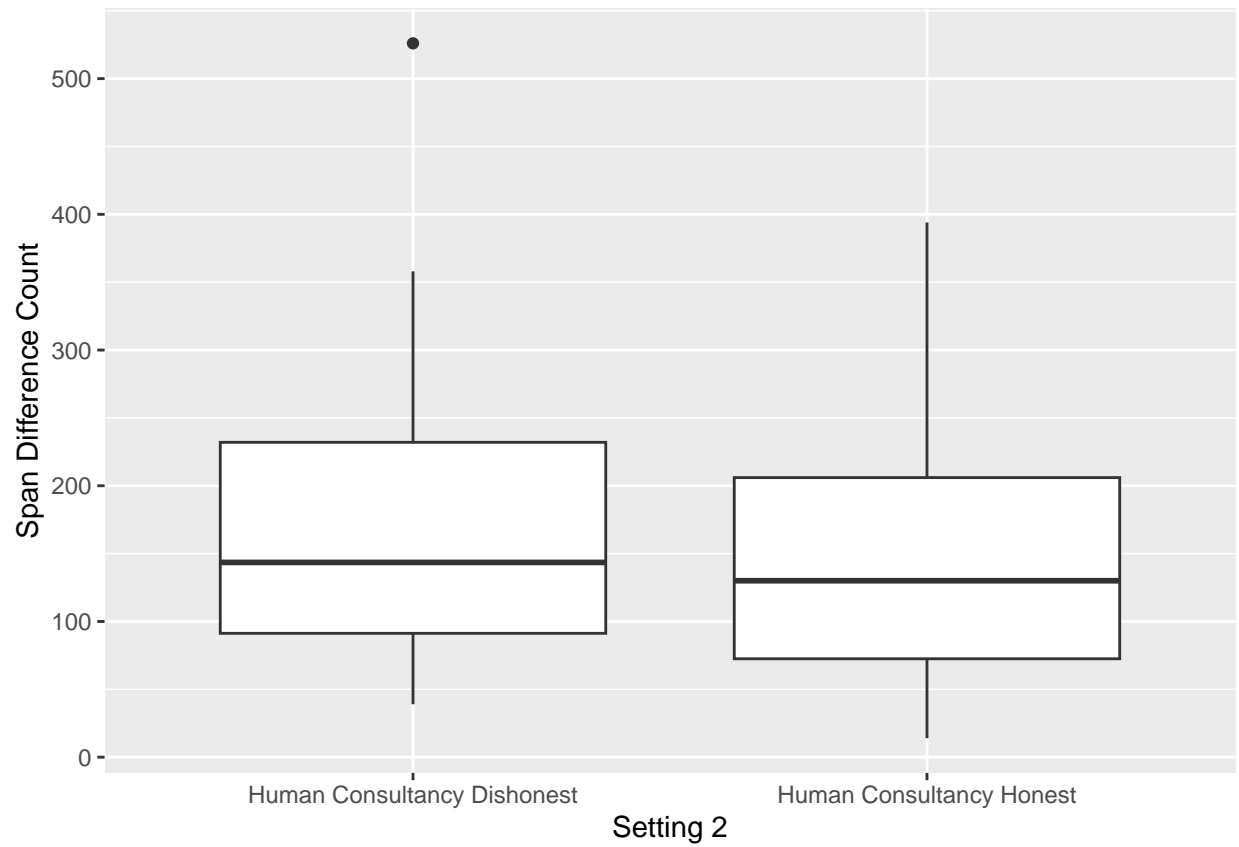
```
##                                Question ...                               Settlin
## 0    By the end of the passage. what can we underst... ...   Human Debate - Human Consultancy Hon
## 2    By the end of the passage. what can we underst... ...   Human Debate - Human Consultancy Dishon
## 30   Did the questions Tremaine needed answers to g... ...   Human Debate - Human Consultancy Hon
## 32   Did the questions Tremaine needed answers to g... ...   Human Debate - Human Consultancy Dishon
## 60   From the information the story provides, do yo... ...   Human Debate - Human Consultancy Hon
## ..                                ... ...
## 510  Why was the main character daydreaming about b... ...   Human Debate - Human Consultancy Dishon
## 514           Why was the murderer trying to kill Bo? ...   Human Debate - Human Consultancy Hon
## 516           Why was the murderer trying to kill Bo? ...   Human Debate - Human Consultancy Dishon
## 544   Why were Jorgenson and Ganti not put to death? ...   Human Debate - Human Consultancy Dishon
## 546   Why were Jorgenson and Ganti not put to death? ...   Human Debate - Human Consultancy Hon
##
## [87 rows x 10 columns]
```

```
print(filtered_no_outliers.groupby(['Setting 2','Acc_1','Acc_2'])['Span Difference Count'].describe())
```

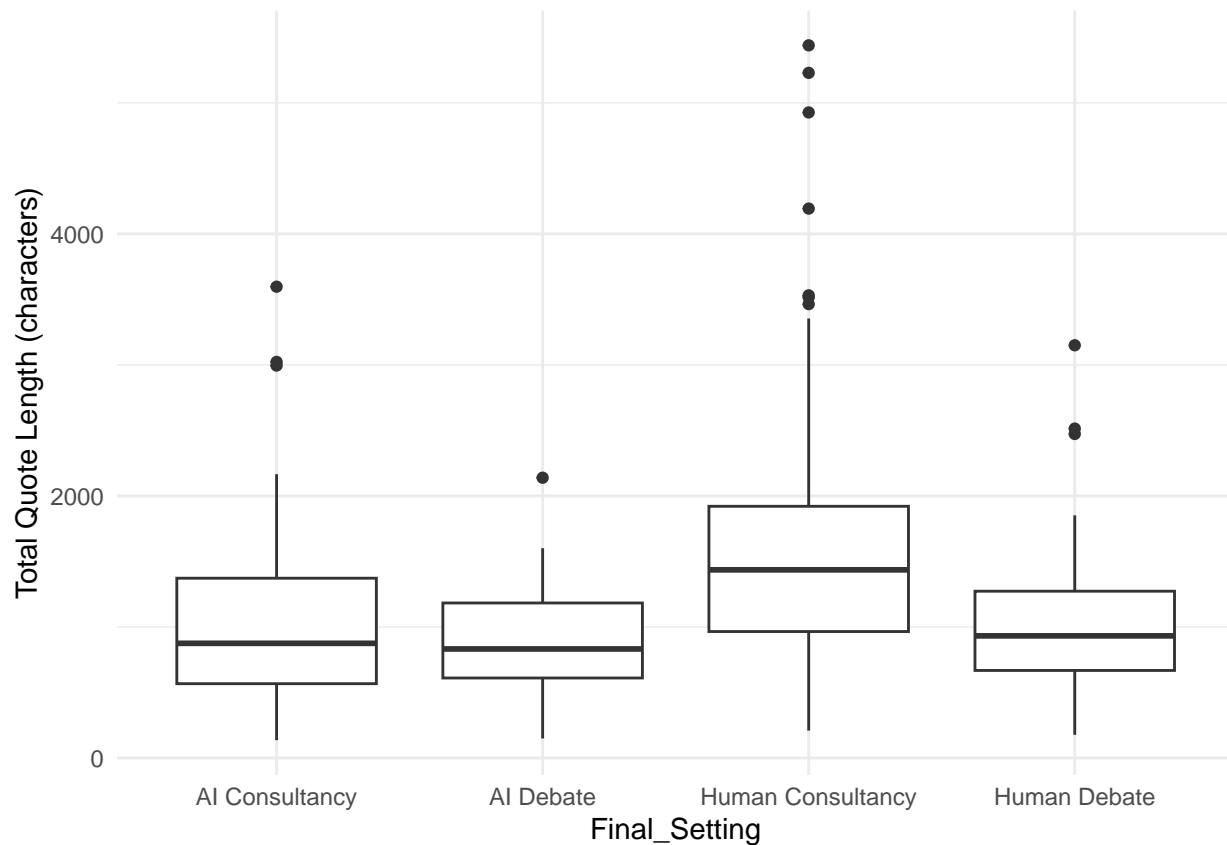
```
##                                count      mean ...    75%    max
## Setting 2                Acc_1 Acc_2
## Human Consultancy Dishonest False False    5.0  187.200000 ...  275.00  293.0
##                                True      8.0  149.625000 ...  236.25  275.0
##                                True False  16.0  148.687500 ...  182.00  358.0
##                                True      16.0  156.500000 ...  220.25  289.0
## Human Consultancy Honest   False False    4.0  144.750000 ...  257.25  267.0
##                                True      12.0  122.416667 ...  164.75  325.0
##                                True False    4.0  197.000000 ...  224.25  273.0
##                                True      22.0  153.409091 ...  195.00  394.0
##
## [8 rows x 8 columns]
```

```
characters<- py$characters_agg
span_difference_debate_consultancies<-py$filtered_df
ggplot(span_difference_debate_consultancies) +
  geom_boxplot(aes(x = `Setting 2`, y = `Span Difference Count`))
```

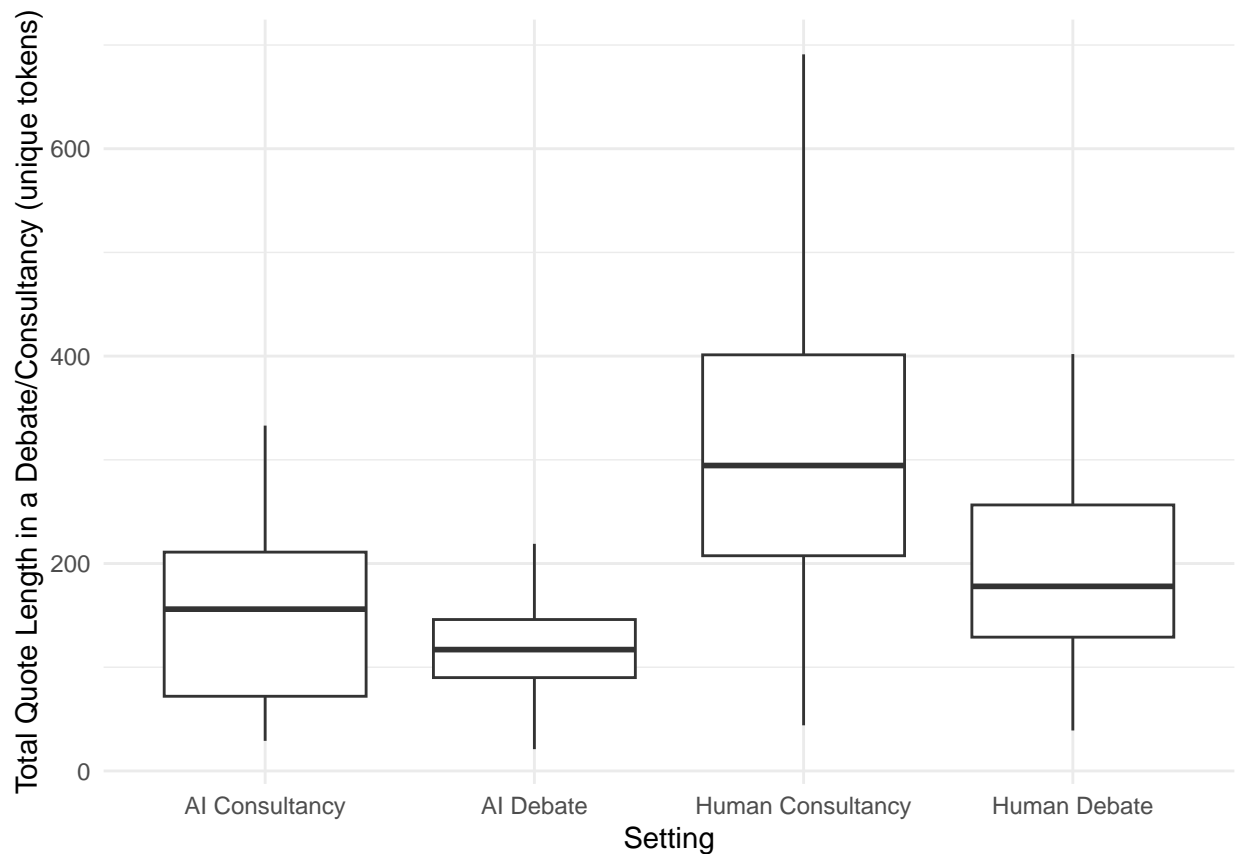




```
ggplot(characters) +  
  geom_boxplot(aes(x = Final_Setting, y = `Quote length`)) +  
  labs(y = "Total Quote Length (characters)") +  
  theme_minimal()
```



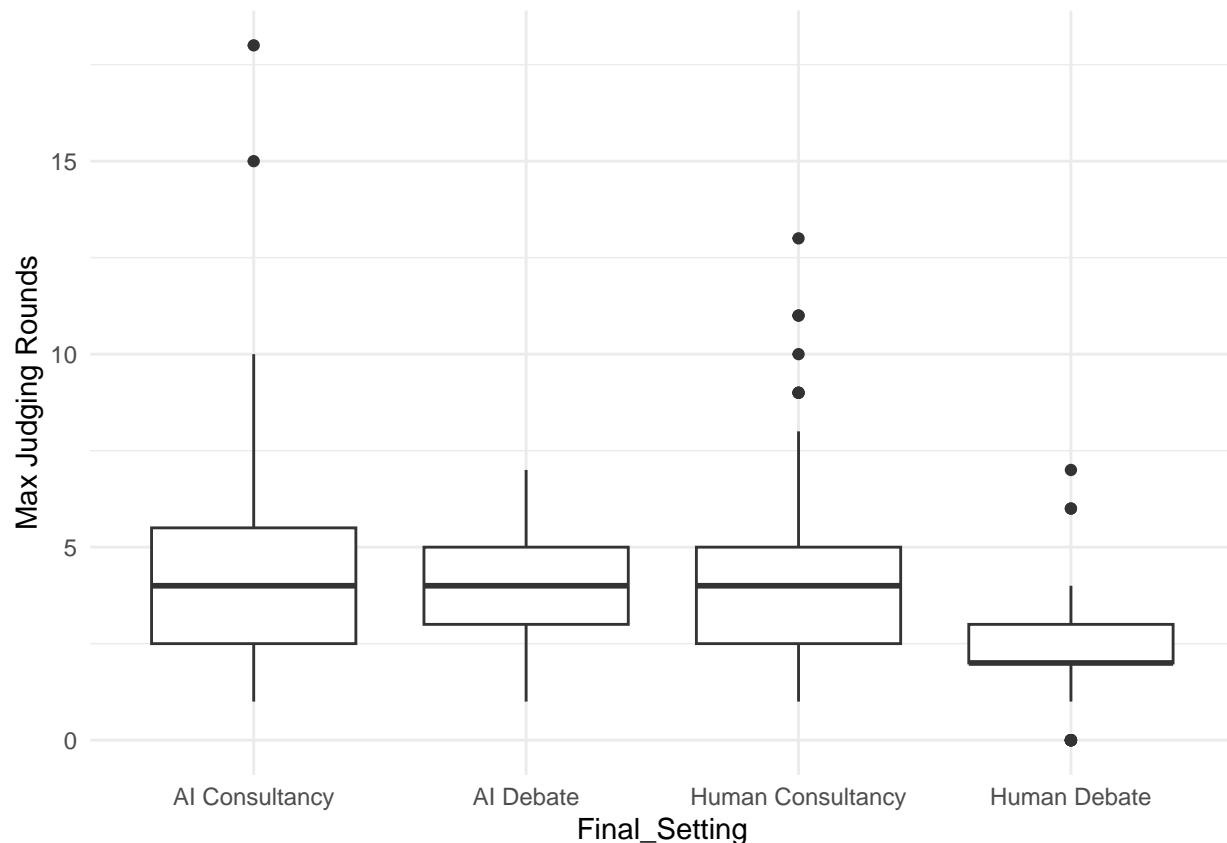
```
filtered <- characters %>%
  group_by(Final_Setting) %>%
  mutate(Q1 = quantile(quote_length, 0.25),
         Q3 = quantile(quote_length, 0.75),
         IQR = Q3 - Q1,
         lower_bound = Q1 - 1.5 * IQR,
         upper_bound = Q3 + 1.5 * IQR) %>%
  filter(quote_length >= lower_bound & quote_length <= upper_bound) %>%
  select(-Q1, -Q3, -IQR, -lower_bound, -upper_bound)
filtered %>%
  ggplot() +
  geom_boxplot(aes(x = Final_Setting, y = quote_length), outlier.shape = NA) +
  labs(y = "Total Quote Length in a Debate/Consultancy (unique tokens)", x = "Setting")+
  theme_minimal()
```



```
pairwise.t.test(filtered$quote_length, filtered$Final_Setting)
```

```
##
## Pairwise comparisons using t tests with pooled SD
##
## data: filtered$quote_length and filtered$Final_Setting
##
##           AI Consultancy  AI Debate      Human Consultancy
## AI Debate      0.1580      -              -
## Human Consultancy 0.0000005872140 0.00000000000002 -
## Human Debate      0.3784      0.0045      0.00000000000367
##
## P value adjustment method: holm
```

```
characters <- characters %>%
  group_by(`Room name`,) %>%
  mutate(`Max judge rounds by room` = max(`Num previous judging rounds`, na.rm = TRUE)) %>%
  ungroup()
ggplot(characters) +
  geom_boxplot(aes(x = Final_Setting, y = `Max judge rounds by room`)) +
  labs(y = 'Max Judging Rounds') +
  theme_minimal()
```

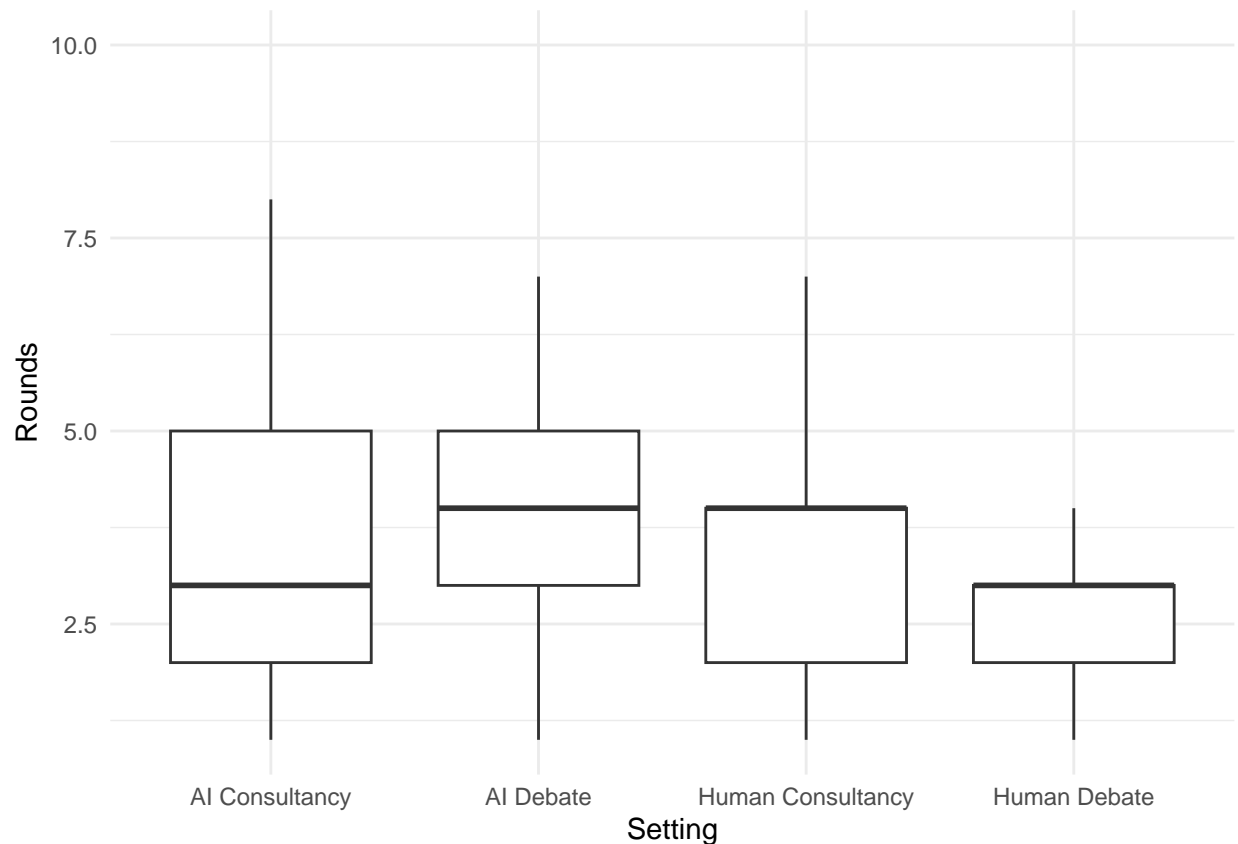


```
pairwise.t.test(characters$`Max judge rounds by room`, characters$Final_Setting)
```

```
##
## Pairwise comparisons using t tests with pooled SD
##
## data: characters$`Max judge rounds by room` and characters$Final_Setting
##
##           AI Consultancy AI Debate Human Consultancy
## AI Debate      0.137      -          -
## Human Consultancy 0.055      0.914      -
## Human Debate    0.0000003    0.002    0.0000020
##
## P value adjustment method: holm
```

```
filtered <- characters %>%
  group_by(Final_Setting) %>%
  mutate(Q1 = quantile(`Max judge rounds by room`, 0.25),
         Q3 = quantile(`Max judge rounds by room`, 0.75),
         IQR = Q3 - Q1,
         lower_bound = Q1 - 1.5 * IQR,
         upper_bound = Q3 + 1.5 * IQR) %>%
  filter(`Max judge rounds by room` >= lower_bound & `Max judge rounds by room` <= upper_bound) %>%
  select(-Q1, -Q3, -IQR, -lower_bound, -upper_bound)
filtered %>%
  ggplot() +
```

```
geom_boxplot(aes(x = Final_Setting, y = `Max judge rounds by room`), outlier.shape = NA) +
labs(y = "Rounds", x = "Setting")+
theme_minimal()
```



```
pairwise.t.test(filtered$quote_length, filtered$Final_Setting)
```

```
##
## Pairwise comparisons using t tests with pooled SD
##
## data: filtered$quote_length and filtered$Final_Setting
##
##           AI Consultancy  AI Debate  Human Consultancy
## AI Debate           0.192         -                  -
## Human Consultancy 0.00000150627713 0.000000000000097 -
## Human Debate      0.560           0.018          0.00000000003675
##
## P value adjustment method: holm
```

```
filtered %>% group_by(Final_Setting) %>% summarise(avground = mean(`Max judge rounds by room`))
```

```
## # A tibble: 4 x 2
##   Final_Setting  avground
##   <chr>         <dbl>
## 1 AI Consultancy 4.24
```

```
## 2 AI Debate          4.07
## 3 Human Consultancy  3.61
## 4 Human Debate       2.52
```

### Length of debates, stratified

```
per_turn = turns.merge(
    debates[["Room name", "Honest debater", "Dishonest debater", "Question", "Article ID",
            "Speed annotator accuracy", "Untimed annotator context", "Untimed annotator context bins"],
    how="left",
    on="Room name",
)

print(per_turn.groupby('Final_Setting')['Num previous judging rounds'].mean())
```

```
## Final_Setting
## AI Consultancy      4.173252
## AI Debate           2.986231
## Human Consultancy   2.759310
## Human Debate        1.475072
## Name: Num previous judging rounds, dtype: float64
```

```
# Calculate the IQR and bounds for each group in 'Setting 2'
grouped = per_turn.groupby('Setting')['Num previous judging rounds']

Q1 = grouped.quantile(0.25)
Q3 = grouped.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers based on the computed bounds
filtered_no_outliers = per_turn[
    (per_turn['Setting'].map(lower_bound) <= per_turn['Num previous judging rounds']) &
    (per_turn['Setting'].map(upper_bound) >= per_turn['Num previous judging rounds'])
]

filtered_no_outliers
```

```
##           Room name ... Final_Accuracy
## 0           ambition-8 ...           True
## 1           ambition-8 ...           True
## 2           ambition-8 ...           True
## 3           ambition-8 ...           True
## 4           ambition-8 ...           True
## ...           ... ...
## 6013        break-a-leg-3 ...           True
## 6014        break-a-leg-3 ...           True
## 6015 the-absurdity-of-family-love-2 ...           True
## 6016 the-absurdity-of-family-love-2 ...           True
```

```
## 6017 the-absurdity-of-family-love-2 ... True
##
## [5875 rows x 27 columns]
```

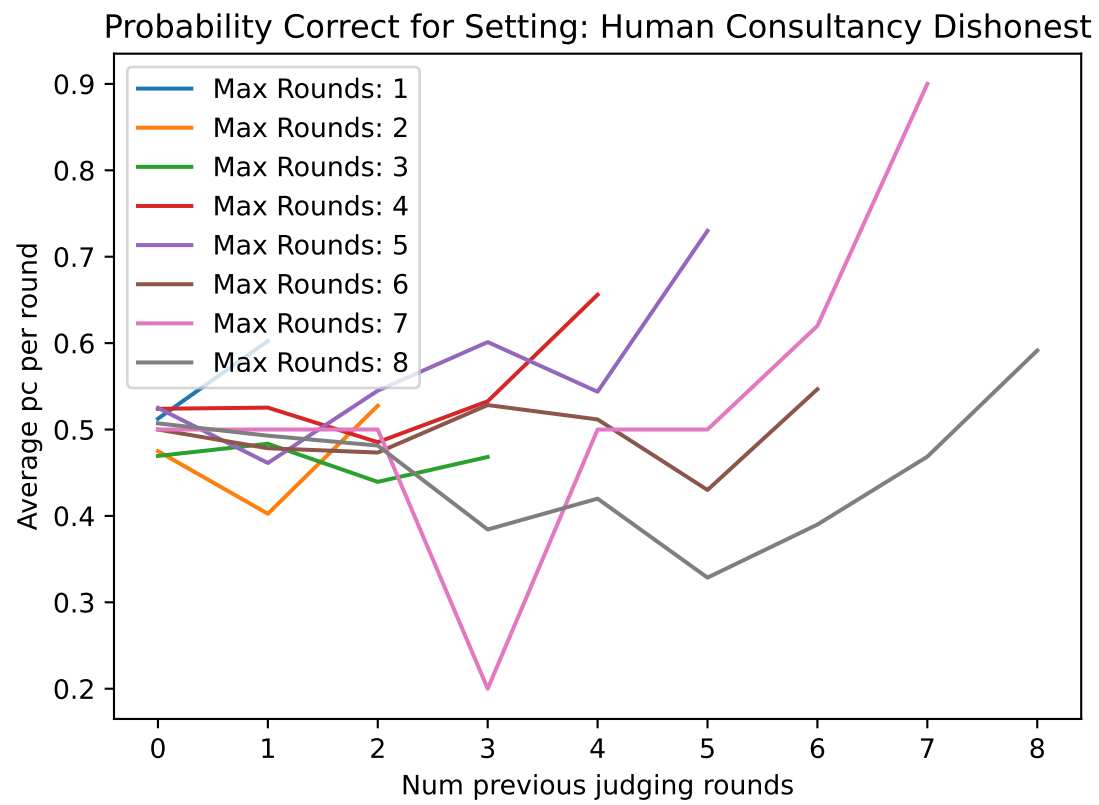
```
for setting in filtered_no_outliers['Setting'].unique():
    per_turn_setting = filtered_no_outliers[filtered_no_outliers['Setting']==setting]
    print(setting)
    # Calculate the maximum 'Num previous judging rounds' for each combination of 'Room name' and 'Participant'
    per_turn_setting['Max judge rounds by room'] = per_turn_setting.groupby(['Room name', 'Participant'])
    ## Just based on the number of rounds

    for i in range(1, per_turn_setting['Max judge rounds by room'].max() + 1):
        max_rounds = per_turn_setting[(per_turn_setting['Max judge rounds by room'] == i) & (per_turn_setting['Probability correct'] > 0.5)]
        print(len(max_rounds))
        # Group by 'Num previous judging rounds' and calculate the mean of 'Probability correct'
        average_pc_per_round = max_rounds.groupby('Num previous judging rounds')['Probability correct'].mean()

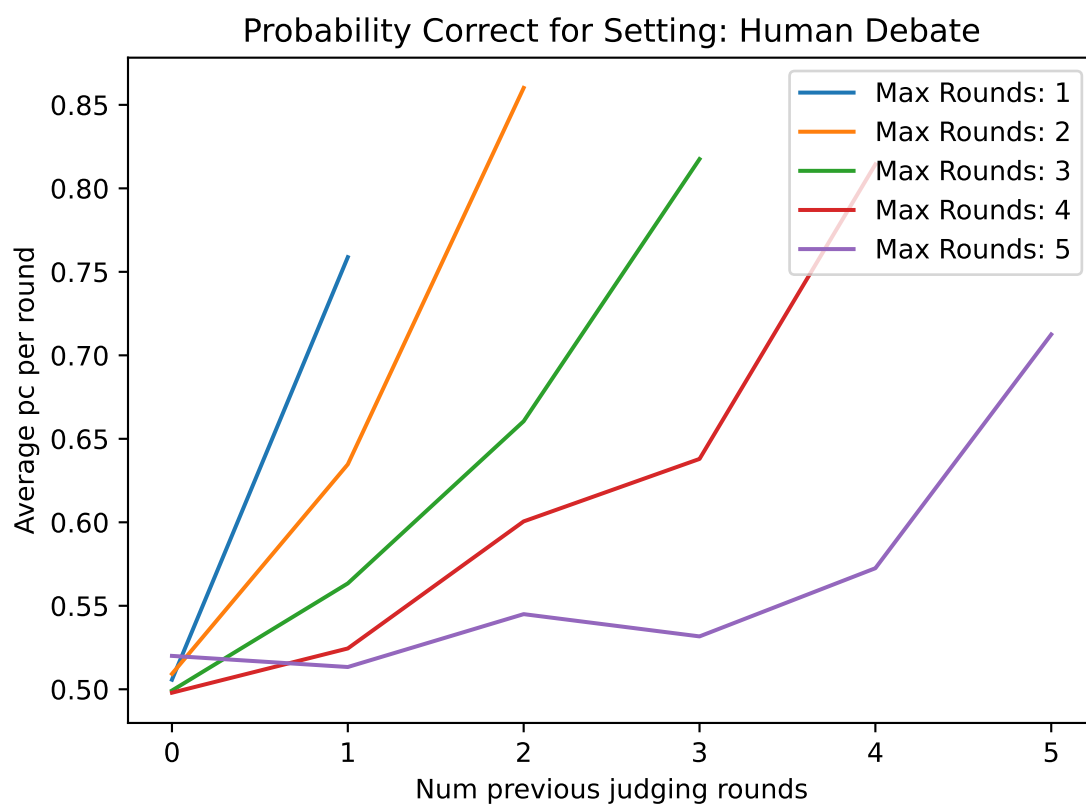
        # Create a new DataFrame with 'Num previous judging rounds' and 'Average pc per round'
        probability_correct_round = pd.DataFrame({'Num previous judging rounds': average_pc_per_round.index,
                                                  'Average pc per round': average_pc_per_round.values})

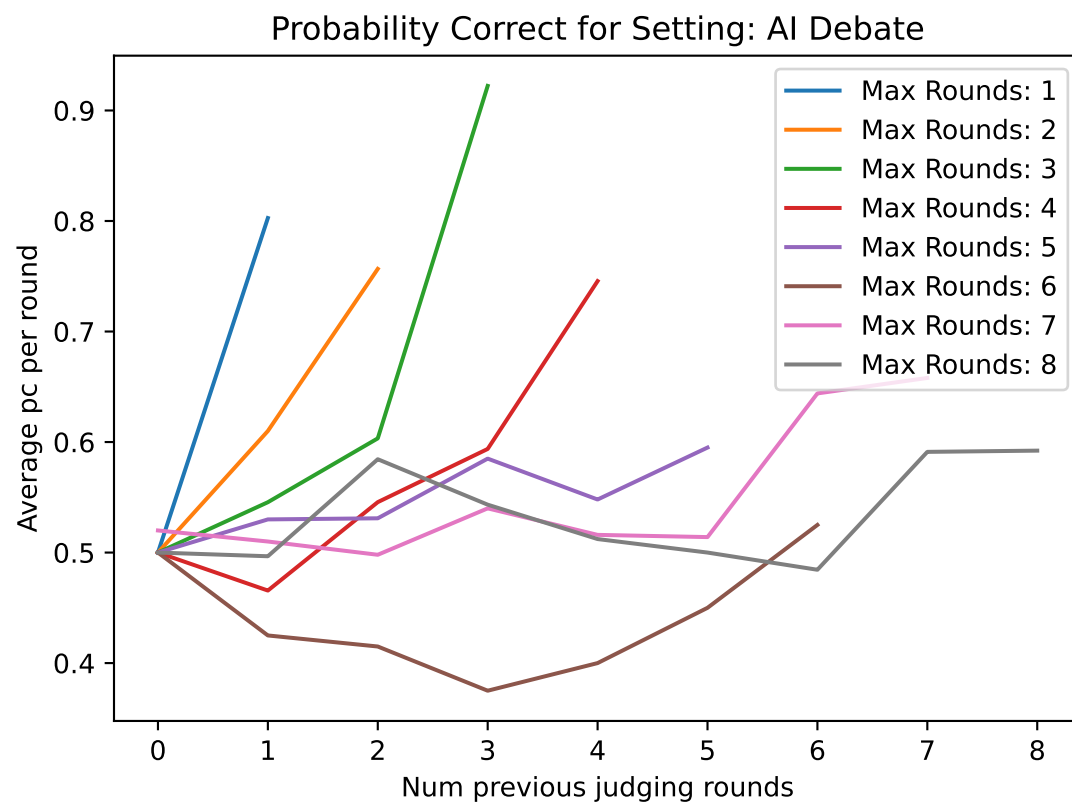
        # Plotting the data with label for the line
        plt.plot(probability_correct_round['Num previous judging rounds'], probability_correct_round['Average pc per round'], label=setting)

plt.title(f"Probability Correct for Setting: {setting}")
plt.xlabel('Num previous judging rounds')
plt.ylabel('Average pc per round')
plt.legend()
plt.show()
```

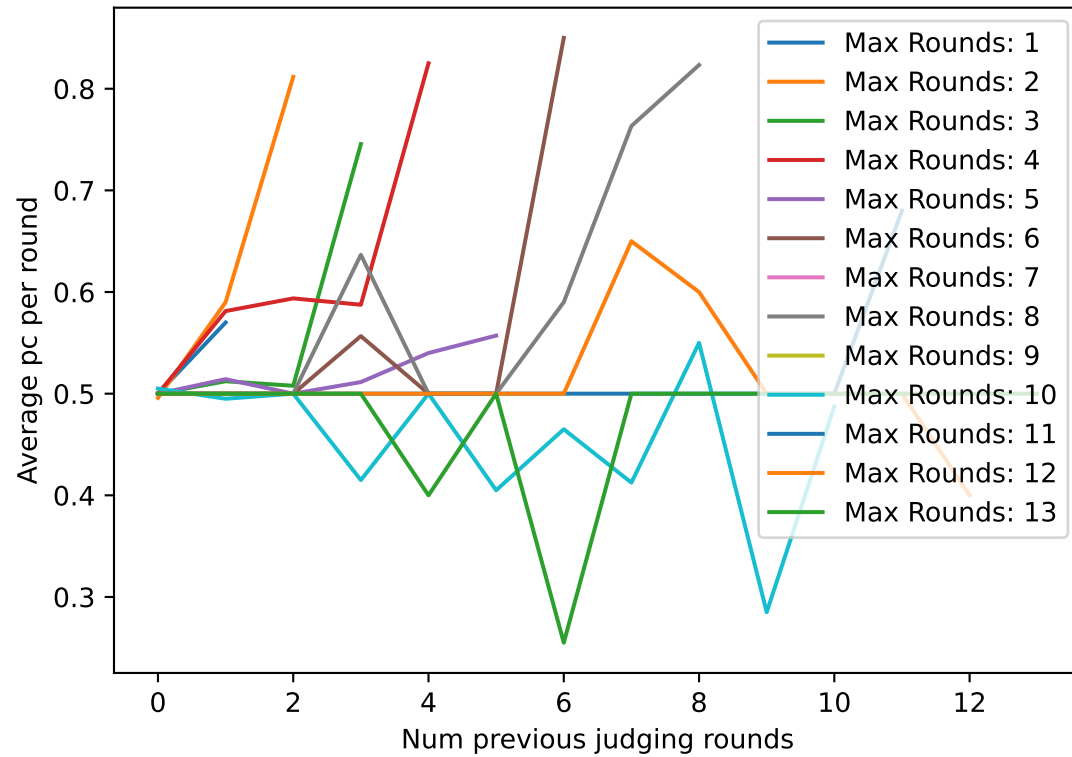


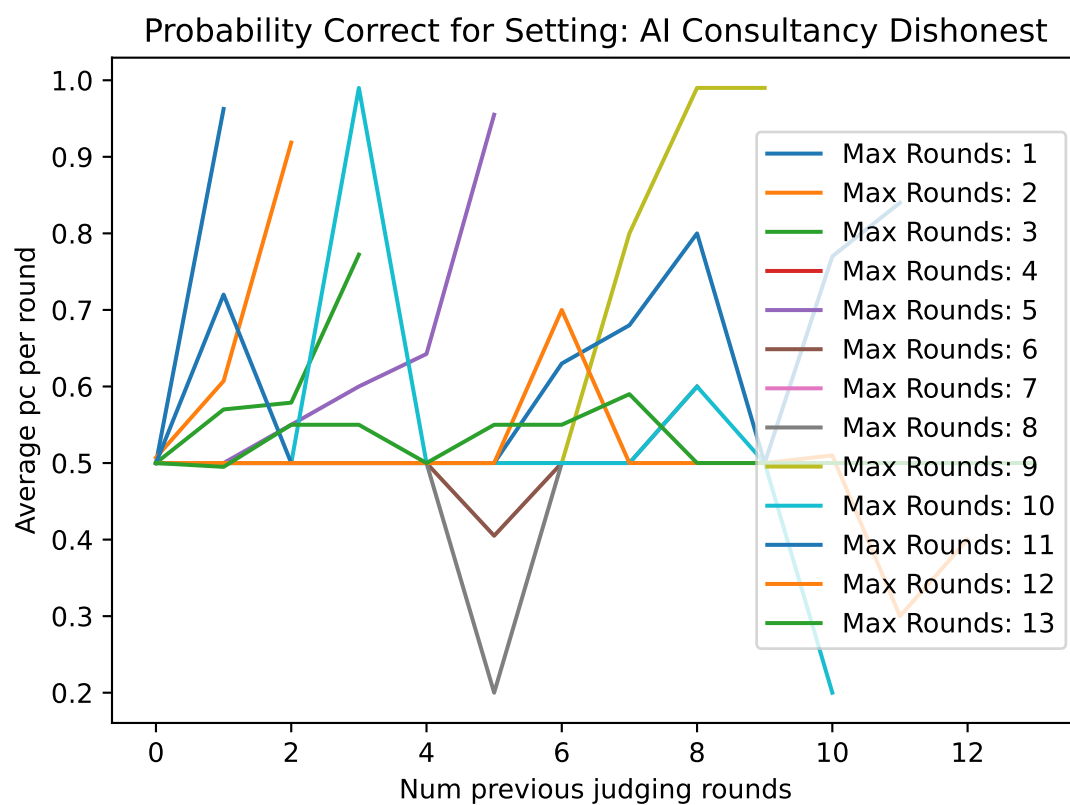




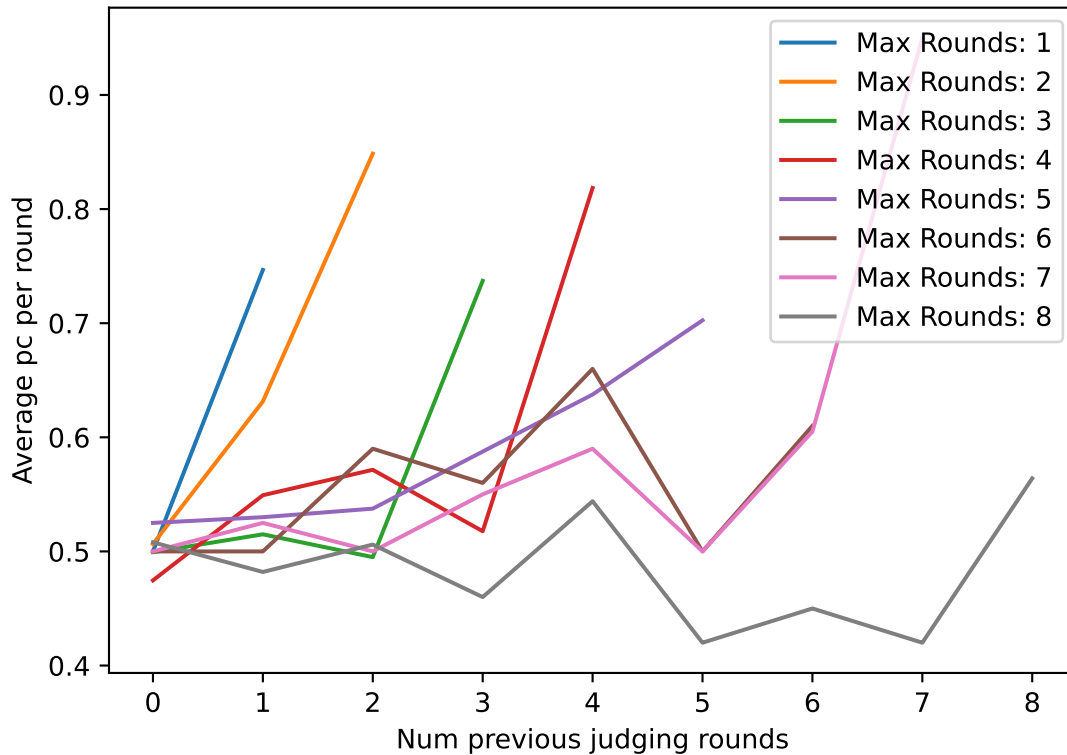


Probability Correct for Setting: AI Consultancy Honest





Probability Correct for Setting: Human Consultancy Honest



```

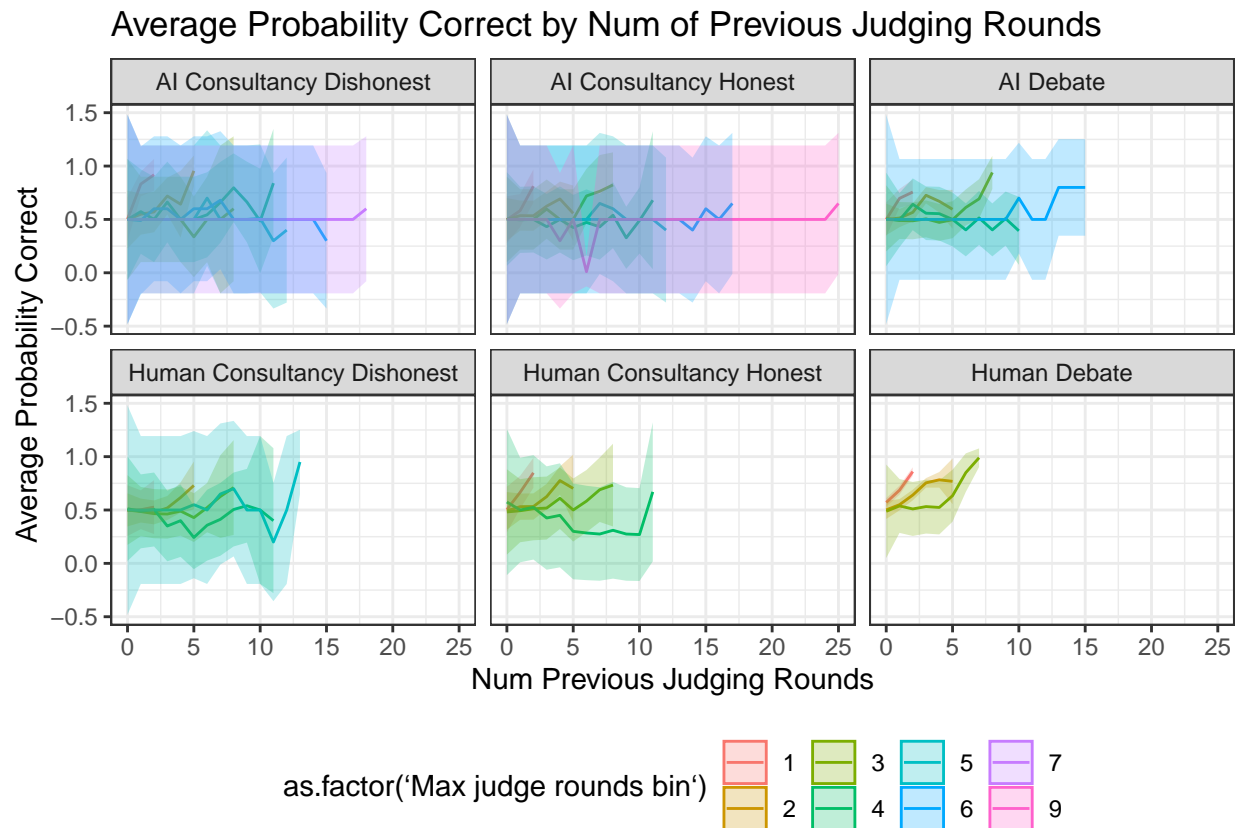
strat <- py$per_turn
strat <- strat %>%
  group_by(`Room name`, Participant) %>%
  mutate(`Max judge rounds by room` = max(`Num previous judging rounds`, na.rm = TRUE)) %>%
  ungroup()
strat <- strat %>%
  mutate(`Max judge rounds bin` = cut(`Max judge rounds by room`,
                                     breaks = seq(0, max(`Max judge rounds by room`, na.rm = TRUE) + 3,
                                     labels = FALSE,
                                     include.lowest = TRUE,
                                     right = FALSE))

# Plot using ggplot2
strat %>%
  group_by(Setting, `Num previous judging rounds`, `Max judge rounds bin`) %>%
  summarize(
    `Average Probability Correct` = mean(`Probability correct`, na.rm = TRUE),
    n = n(),
    se = sqrt(`Average Probability Correct` * (1 - `Average Probability Correct`) / n)
  ) %>%
  mutate(
    lower_ci = `Average Probability Correct` - 1.96 * se,
    upper_ci = `Average Probability Correct` + 1.96 * se
  ) %>%
  ggplot(aes(x = `Num previous judging rounds`, y = `Average Probability Correct`, col = as.factor(`Max
  geom_line() +

```

```
geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci, fill = as.factor(`Max judge rounds bin`)), group = a
labs(title = "Average Probability Correct by Num of Previous Judging Rounds",
     x = "Num Previous Judging Rounds",
     y = "Average Probability Correct") +
facet_wrap(~Setting) +
theme_bw() +
theme(legend.position = "bottom")
```

## 'summarise()' has grouped output by 'Setting', 'Num previous judging rounds'.  
## You can override using the '.groups' argument.

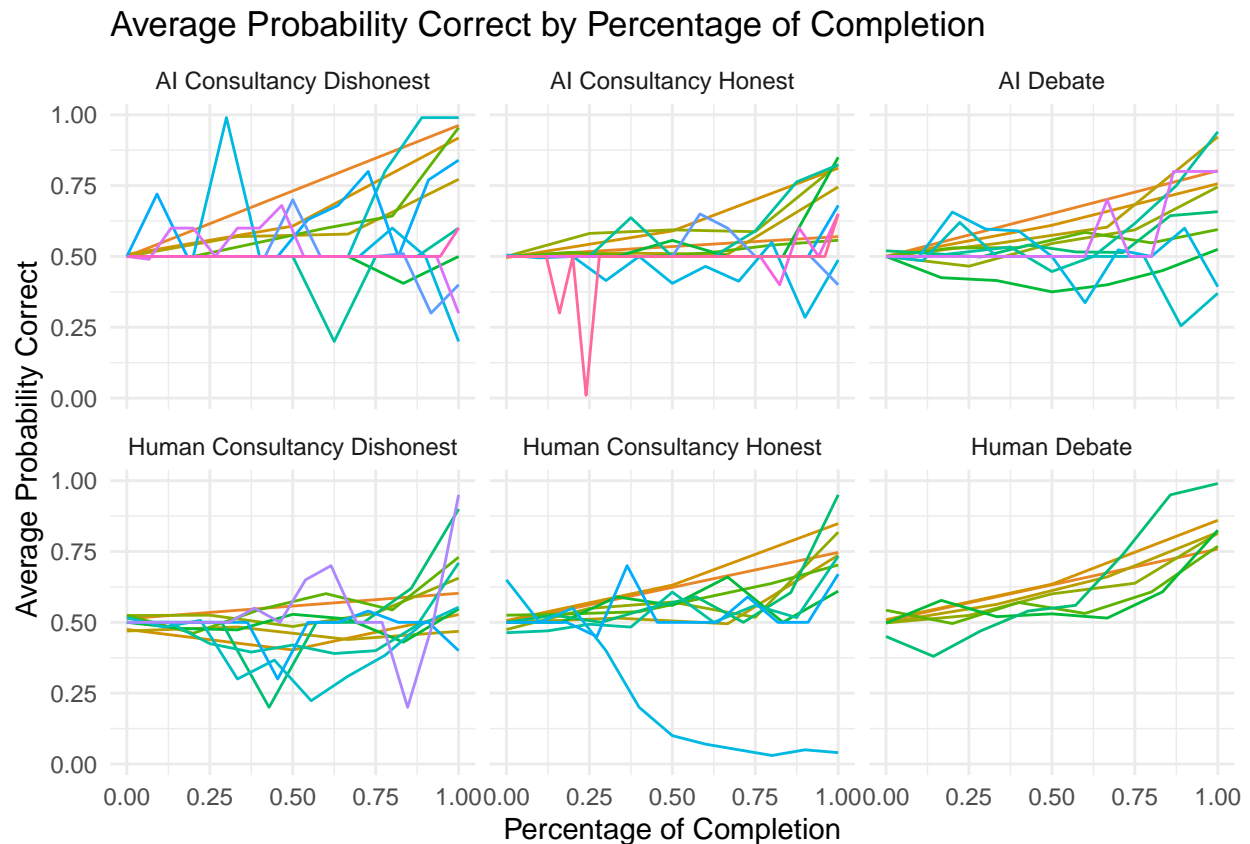


```
strat %>%
  group_by(Setting, `Num previous judging rounds`, `Max judge rounds by room`) %>%
  summarize(`Average Probability Correct` = mean(`Probability correct`, na.rm = TRUE)) %>%
  mutate(Completion = `Num previous judging rounds` / `Max judge rounds by room`) %>%
  ggplot(aes(x = Completion, y = `Average Probability Correct`, col = as.factor(`Max judge rounds by room`))) +
  geom_line() +
  labs(title = "Average Probability Correct by Percentage of Completion",
       x = "Percentage of Completion",
       y = "Average Probability Correct") +
  facet_wrap(~Setting) +
  theme_minimal() +
  theme(legend.position = "none")
```

## 'summarise()' has grouped output by 'Setting', 'Num previous judging rounds'.

```
## You can override using the '.groups' argument.
```

```
## Warning: Removed 10 rows containing missing values ('geom_line()').
```



Time (offline judging..?)

```
# Convert to datetime
judgments["Offline judging start time"] = pd.to_datetime(judgments["Offline judging start time"], unit="ms")
judgments["Offline judging end time"] = pd.to_datetime(judgments["Offline judging end time"], unit="ms")

# Calculate offline judging time in minutes
judgments["Offline judging time"] = (judgments["Offline judging end time"] - judgments["Offline judging start time"]) / 60

print(f"Number of offline judgments on consultancies:\n{judgments[judgments['Setting'].str.contains('Consultancy')]['Offline judging time'].count()}")

## Number of offline judgments on consultancies:
## count      13.000000
## mean       447.514203
## std        1236.792144
## min         1.169167
## 25%         1.836600
## 50%         5.664767
```

```
## 75%          13.967783
## max          4369.697933
## Name: Offline judging time, dtype: float64
## Only 13...
```

```
# Filter out rows with NaT values
valid_judging_time = judgments["Offline judging time"].dropna()

# Calculate summary statistics
summary_stats = valid_judging_time.describe()
print(summary_stats)
```

```
## count          203.000000
## mean           255.826710
## std            1372.208730
## min             0.667467
## 25%             2.867950
## 50%             5.176250
## 75%            10.295583
## max            14202.493917
## Name: Offline judging time, dtype: float64
```

```
# Filter judgments with offline judging time above 65 minutes
filtered_judgments = judgments[(judgments["Offline judging time"] < 65) & (judgments["Untimed annotator"] > 0)]

# Print filtered judgments
# print("Filtered judgments with offline judging time above 65 minutes:")
print(filtered_judgments['Offline judging time'].describe())
```

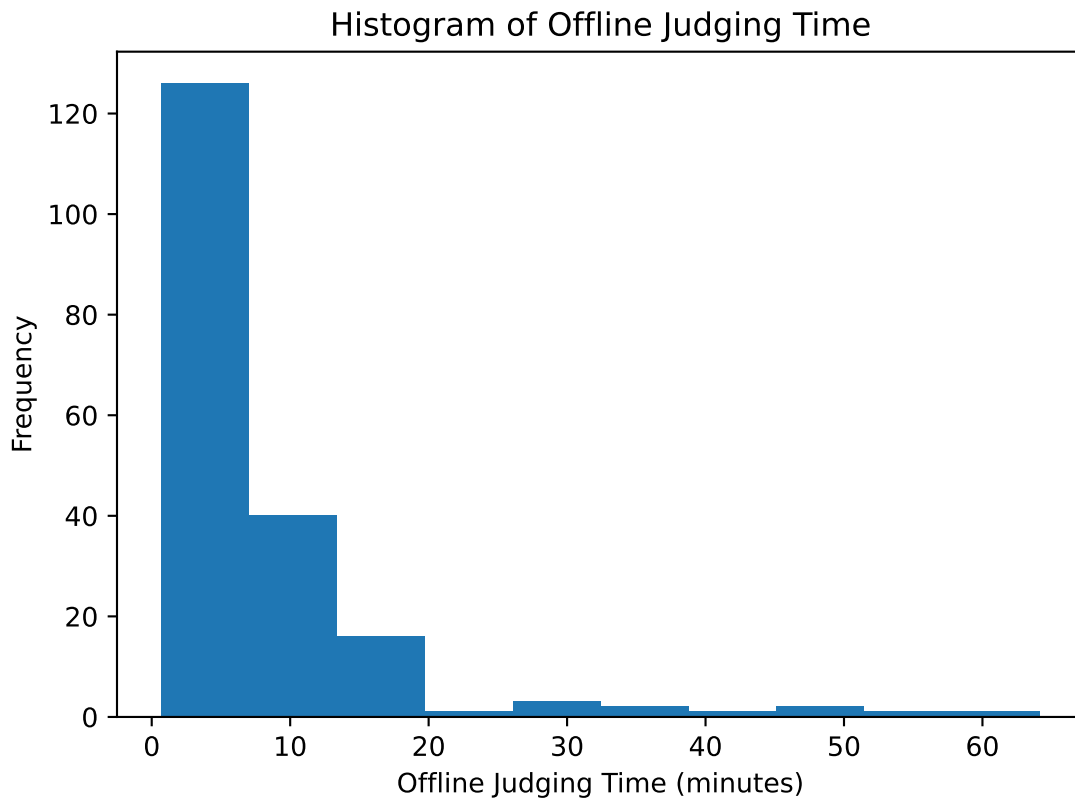
```
## count          193.000000
## mean            8.013787
## std             9.410150
## min             0.667467
## 25%             2.850450
## 50%             5.107450
## 75%             8.716300
## max             64.173267
## Name: Offline judging time, dtype: float64
```

```
# Create the histogram
plt.hist(filtered_judgments['Offline judging time'], bins=10)

# Set labels and title
plt.xlabel("Offline Judging Time (minutes)")
plt.ylabel("Frequency")
plt.title("Histogram of Offline Judging Time")

# Display the histogram
plt.show()
```





```

aggregates = {
    'Final probability correct': 'mean',
    'Untimed annotator context': 'mean'
}
filtered_judgments = filtered_judgments.groupby('Offline judging time').agg(aggregates).reset_index()

```

## Analysis

### Question Difficulty

confounder rounds, quotes

```

judgments["Number of judge continues bins"] = pd.cut(
    judgments["Number of judge continues"],
    bins=[0, 3, 6, 9, float('inf')], # bin edges
    labels=['1-3', '4-6', '7-9', '10+'], # labels for the resulting bins
    right=True # includes the right edge of the bin
)
aggregated_df = judgments.groupby(["Setting", "Number of judge continues bins"])["Final_Accuracy"].agg(
    Proportion_True=lambda x: x.mean(),
    Total_Count="size"
).reset_index()
pd.set_option('display.max_rows', None)
print(aggregated_df)

```

```
##           Setting ... Total_Count
## 0    AI Consultancy Dishonest ...      27
## 1    AI Consultancy Dishonest ...       6
## 2    AI Consultancy Dishonest ...       2
## 3    AI Consultancy Dishonest ...       5
## 4      AI Consultancy Honest ...      27
## 5      AI Consultancy Honest ...      18
## 6      AI Consultancy Honest ...       3
## 7      AI Consultancy Honest ...       8
## 8              AI Debate ...      51
## 9              AI Debate ...      27
## 10             AI Debate ...      10
## 11             AI Debate ...       4
## 12 Human Consultancy Dishonest ...      31
## 13 Human Consultancy Dishonest ...      29
## 14 Human Consultancy Dishonest ...       6
## 15 Human Consultancy Dishonest ...       2
## 16   Human Consultancy Honest ...      28
## 17   Human Consultancy Honest ...      18
## 18   Human Consultancy Honest ...       5
## 19   Human Consultancy Honest ...       2
## 20             Human Debate ...     318
## 21             Human Debate ...      57
## 22             Human Debate ...       1
## 23             Human Debate ...       0
##
## [24 rows x 4 columns]
```

```
pd.reset_option('display.max_rows')
```

```
total_counts_for_setting = judgments.groupby('Final_Setting').size()
result = judgments.groupby(["Final_Setting", "Untimed annotator context bins", "Number of judge continu
    Proportion_True=pd.NamedAgg(column='Final_Accuracy', aggfunc=lambda x: x.mean()),
    Context_Count=pd.NamedAgg(column='Final_Accuracy', aggfunc='size'),
    Proportion_Context=pd.NamedAgg(column='Final_Setting', aggfunc=lambda x: len(x) / total_counts_for_s
).reset_index()
print(f'Is it number of rounds (meaning more evidence) that confounds the consultancy accuracy?:\n{result}')
```

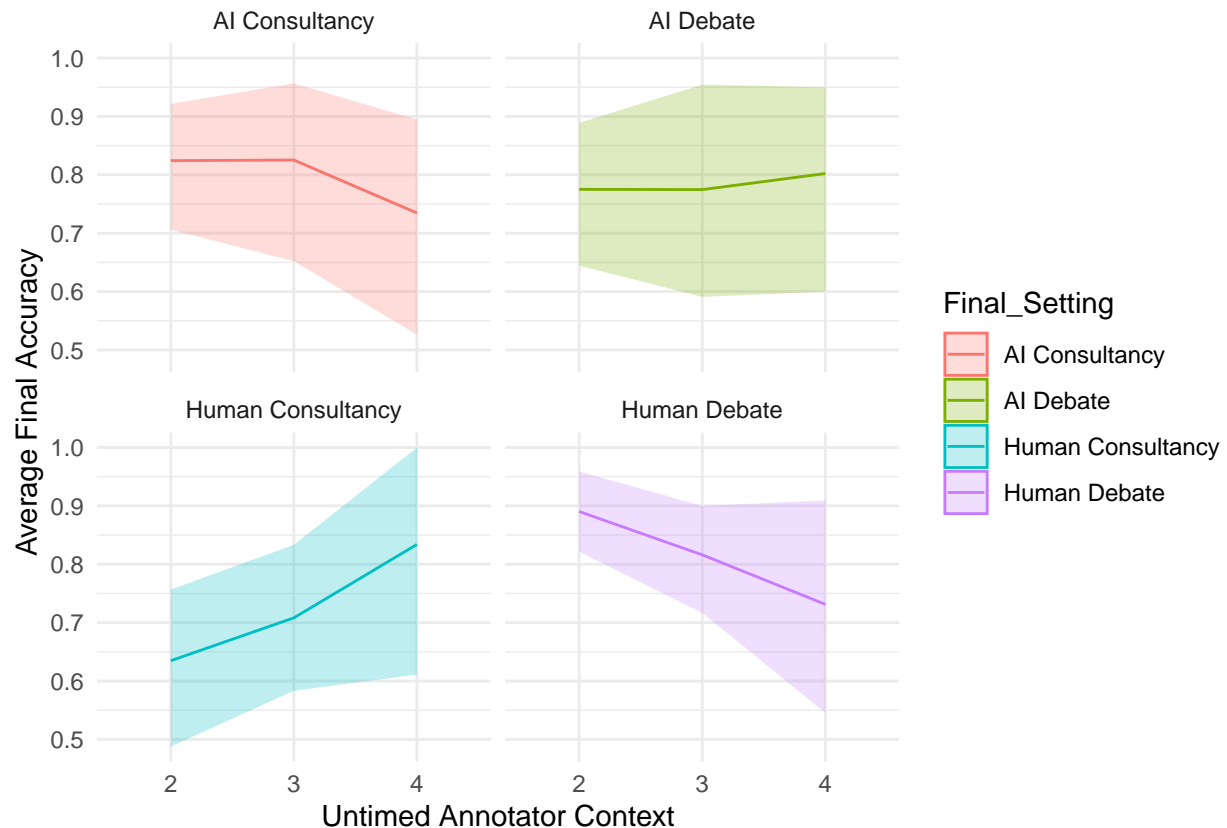
```
## Is it number of rounds (meaning more evidence) that confounds the consultancy accuracy?:
##      Final_Setting ... Proportion_Context
## 0    AI Consultancy ...              NaN
## 1    AI Consultancy ...      0.010417
## 2    AI Consultancy ...              NaN
## 3    AI Consultancy ...              NaN
## 4    AI Consultancy ...      0.291667
## ..           ... ..
## 59   Human Debate ...              NaN
## 60   Human Debate ...      0.076923
## 61   Human Debate ...      0.018568
## 62   Human Debate ...              NaN
## 63   Human Debate ...              NaN
```

```
##
## [64 rows x 6 columns]

judgments$`Untimed annotator context bins` <- as.factor(judgments$`Untimed annotator context bins`)

bootstrap_mean <- function(data, indices) {
  return(mean(data[indices], na.rm = TRUE))
}

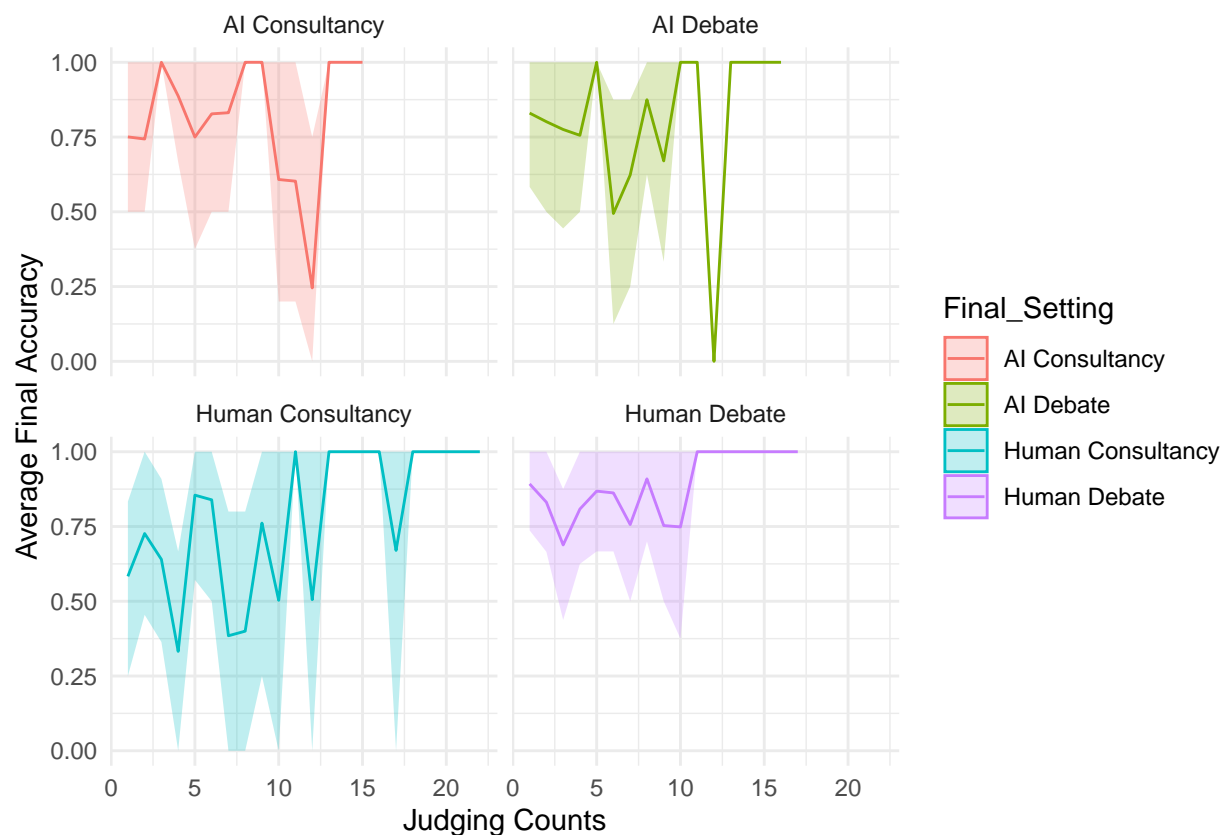
judgments_online %>%
  group_by(`Untimed annotator context bins`, Final_Setting) %>%
  do({
    boot_result <- boot(data = .$Final_Accuracy, statistic = bootstrap_mean, R = 1000)
    data.frame(
      mean_accuracy = mean(boot_result$t, na.rm = TRUE),
      lower_ci = quantile(boot_result$t, 0.025),
      upper_ci = quantile(boot_result$t, 0.975)
    )
  }) %>%
  ggplot(aes(x = `Untimed annotator context bins`, y = mean_accuracy, color = Final_Setting, group = Final_Setting)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci, fill = Final_Setting, color = NULL), alpha = 0.25) +
  labs(y = "Average Final Accuracy", x = "Untimed Annotator Context") +
  theme_minimal() +
  facet_wrap(~ Final_Setting)
```



## Judge Skill

### Judge “Experience”

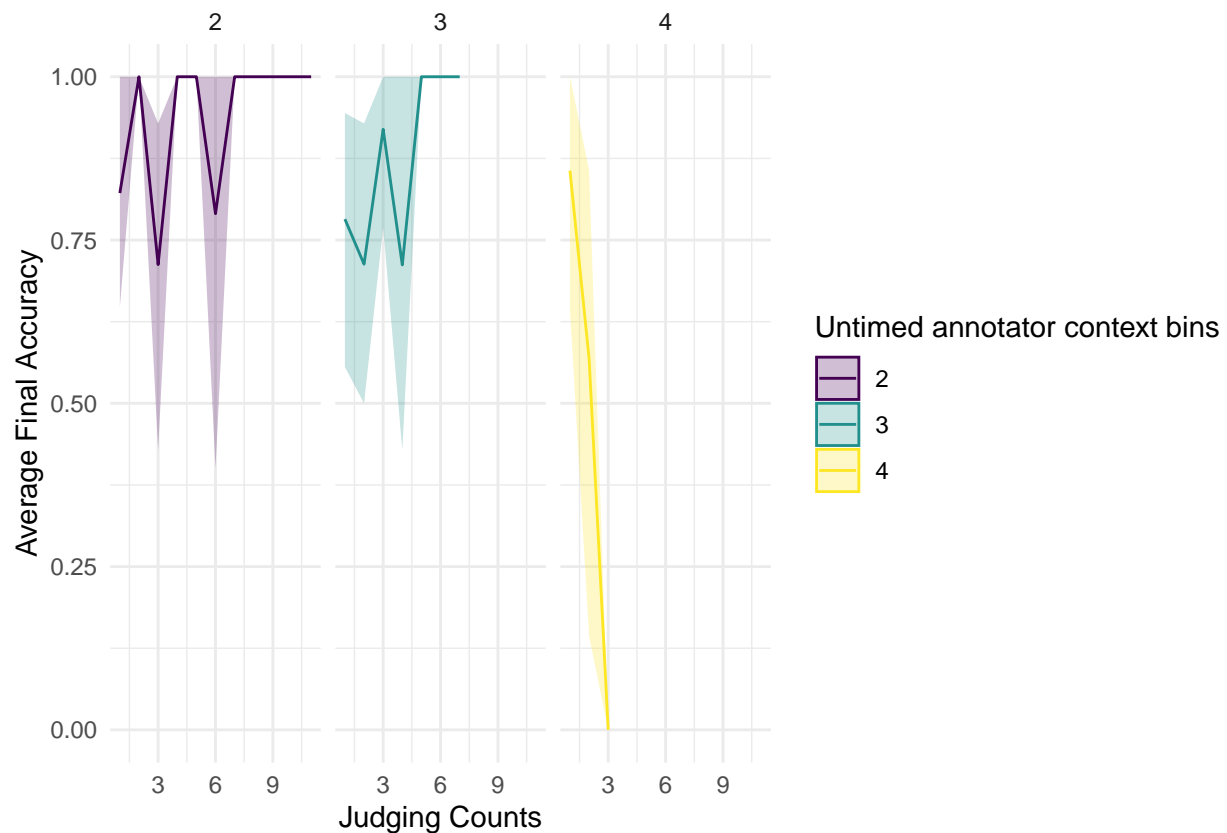
```
judgments_online %>%
  group_by(Final_Setting, Participant) %>%
  arrange(`End time`) %>%
  mutate(count=row_number()) %>%
  group_by(Final_Setting, count) %>%
  do({
    boot_result <- boot(data = .$Final_Accuracy, statistic = bootstrap_mean, R = 1000)
    data.frame(
      mean_accuracy = mean(boot_result$t, na.rm = TRUE),
      lower_ci = quantile(boot_result$t, 0.025, na.rm = TRUE),
      upper_ci = quantile(boot_result$t, 0.975, na.rm = TRUE)
    )
  }) %>%
  ggplot(aes(x = count, y = mean_accuracy, color = Final_Setting, group = Final_Setting)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci, fill = Final_Setting, color = NULL), alpha = 0.25) +
  labs(y = "Average Final Accuracy", x = "Judging Counts") +
  theme_minimal() +
  facet_wrap(~ Final_Setting)
```



```

subset(judgments_online, judgments_online['Setting'] == 'Human Debate') %>%
  group_by(`Untimed annotator context bins`, Participant) %>%
  arrange(`End time`) %>%
  mutate(count=row_number()) %>%
  group_by(`Untimed annotator context bins`, count) %>%
  do({
    boot_result <- boot(data = .$Final_Accuracy, statistic = bootstrap_mean, R = 1000)
    data.frame(
      mean_accuracy = mean(boot_result$t, na.rm = TRUE),
      lower_ci = quantile(boot_result$t, 0.025, na.rm = TRUE),
      upper_ci = quantile(boot_result$t, 0.975, na.rm = TRUE)
    )
  }) %>%
  ggplot(aes(x = count, y = mean_accuracy, color = `Untimed annotator context bins`, group = `Untimed a
  geom_line() +
  geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci, fill = `Untimed annotator context bins`, color = NU
  labs(y = "Average Final Accuracy", x = "Judging Counts") +
  theme_minimal() +
  facet_wrap(~ `Untimed annotator context bins`)

```



## Calibration

S: (1) debaters didnt learn calibration -> calibration over time? S: (2) dishonest debater tricks

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.calibration import calibration_curve

def calibration_plot(df, setting_name, ax=None):
    df['outcome'] = pd.Series(df['Final probability correct'] > 0.5, dtype=int)
    df['confidence'] = df['Final probability correct'].apply(lambda x: x if x > 0.5 else 1 - x)
    df['bins'] = pd.cut(df['confidence'], [0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99])
    # Group by bins and calculate the mean outcome
    df_grouped = df.groupby('bins')['outcome'].mean().reset_index()
    # Compute standard error in each bin
    std_error = df.groupby('bins')['outcome'].apply(lambda x: x.std() / np.sqrt(len(x)) if len(x) > 1 else 0)
    df_grouped['std_error'] = df['bins'].cat.categories.map(std_error)
    if ax is None:
        plt.rcParams.update({'font.size': 16})
        fig, ax = plt.subplots(figsize=(8, 6))
    # Plot the calibration curve with error bars
    ax.plot(df_grouped['bins'].apply(lambda x: x.mid), df_grouped['outcome'], marker='o', linewidth=2, color='blue')
    ax.errorbar(df_grouped['bins'].apply(lambda x: x.mid), df_grouped['outcome'], yerr=df_grouped['std_error'], color='red')
    ax.set_xlabel('Final judge probability')
    ax.set_ylabel('Accuracy')
    ax.set_title(f'Judge calibration for {setting_name}')
    ax.plot([0.5, 1], [0.5, 1], linestyle='--', color='gray', label='Perfect Calibration')
    ax.grid(True)
    ax.legend()
    # Calculate ECE
    actual_labels = df['outcome'].values
    predicted_probs = df['Final probability correct'].values
    prob_true, prob_pred = calibration_curve(actual_labels, predicted_probs, n_bins=10)
    ece = np.mean(np.abs(prob_pred - prob_true) * (prob_true.size / len(actual_labels)))
    # Print ECE
    print(f"Expected Calibration Error (ECE) for {setting_name}: {ece:.4f}")
    plt.show()
    plt.rcParams.update({'font.size': plt.rcParamsDefault['font.size']})

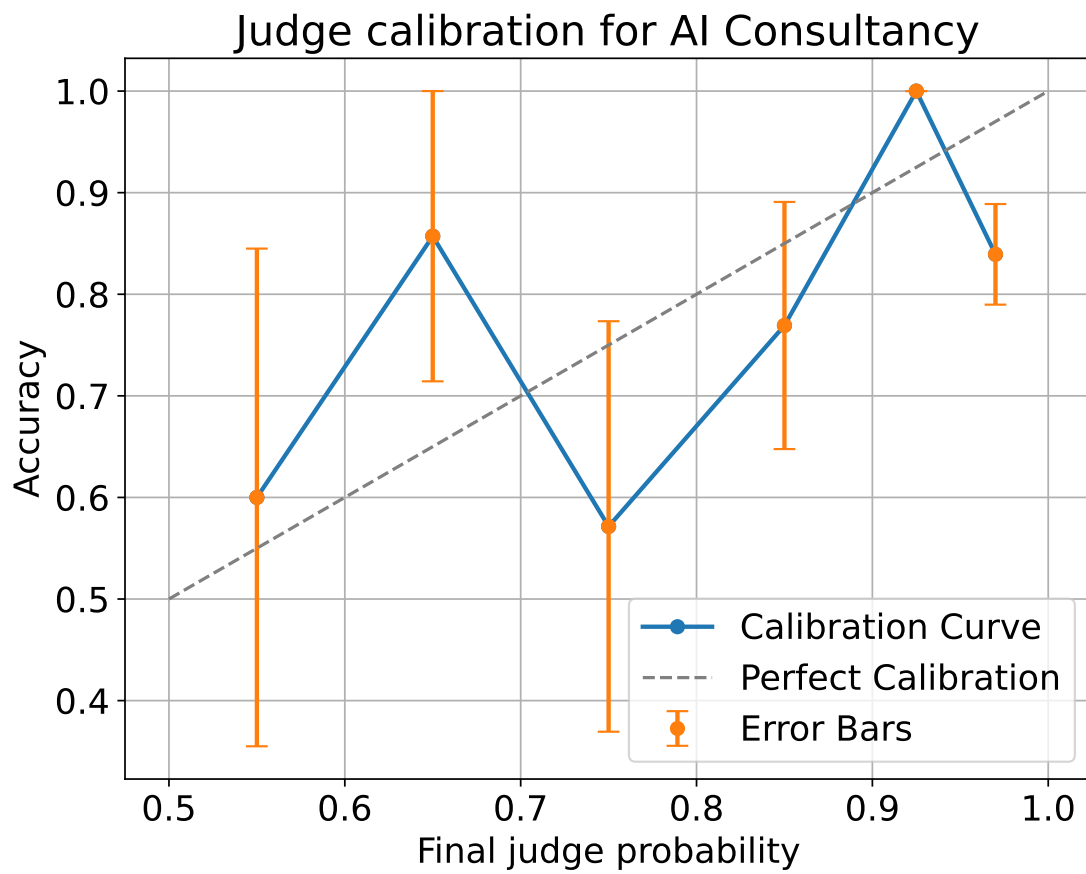
# Loop through each unique setting and create a calibration plot
for setting in judgments_online['Final_Setting'].unique():
    setting_df = judgments_online[judgments['Final_Setting'] == setting].copy()
    calibration_plot(setting_df, setting)

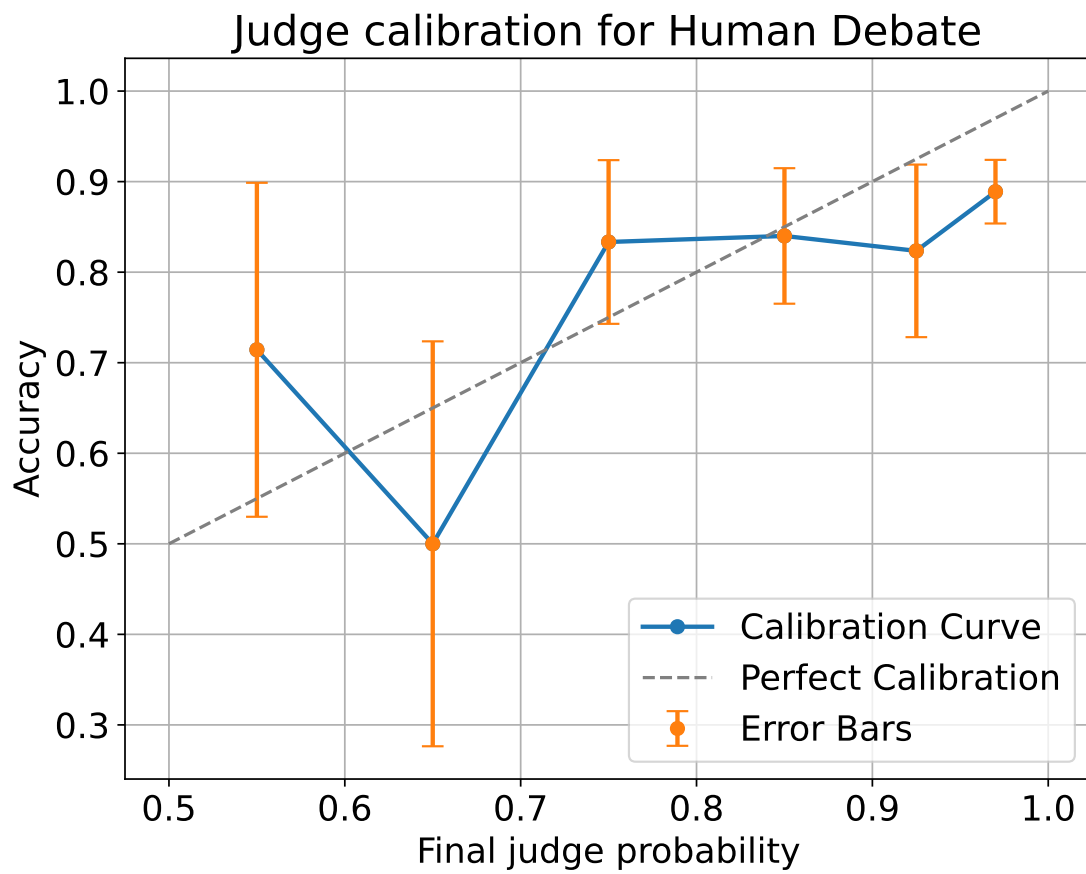
```

```

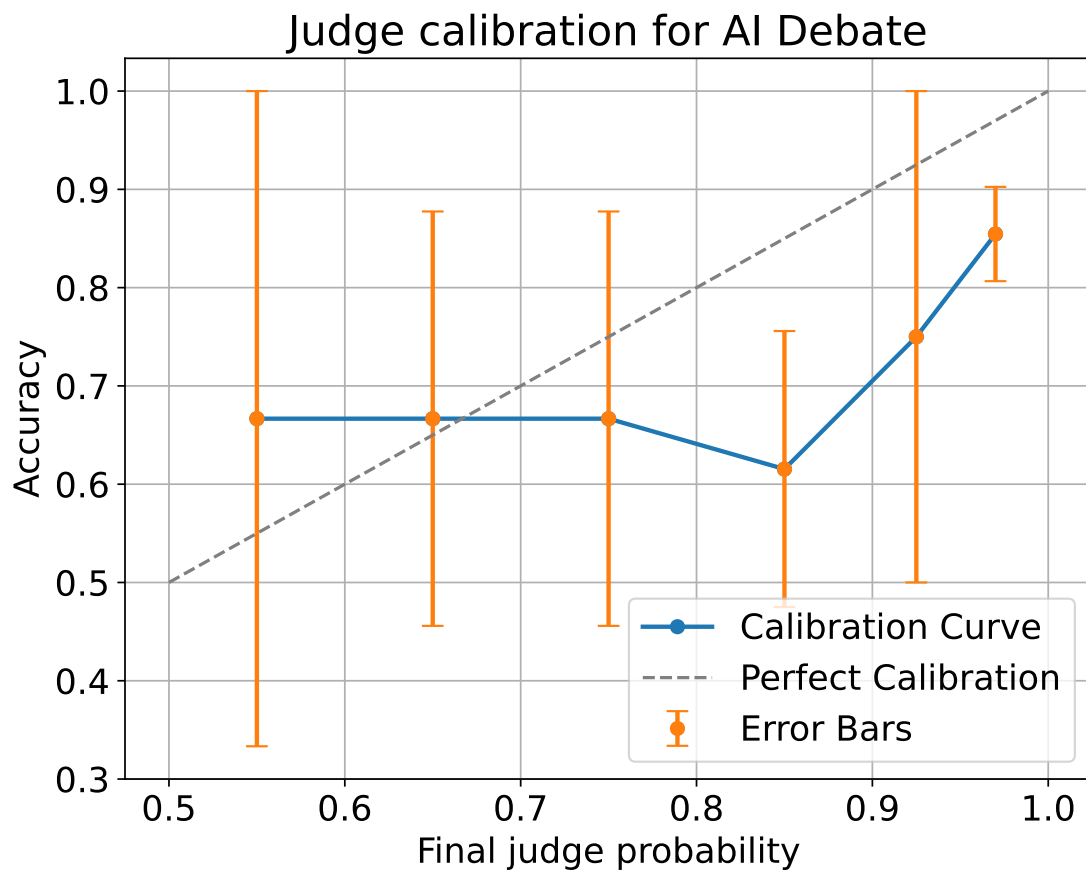
## Expected Calibration Error (ECE) for AI Consultancy: 0.0213
## Expected Calibration Error (ECE) for Human Debate: 0.0152
## Expected Calibration Error (ECE) for AI Debate: 0.0268
## Expected Calibration Error (ECE) for Human Consultancy: 0.0220
##
## <string>:4: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
## <string>:4: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
## <string>:4: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
## <string>:4: UserWarning: Boolean Series key will be reindexed to match DataFrame index.

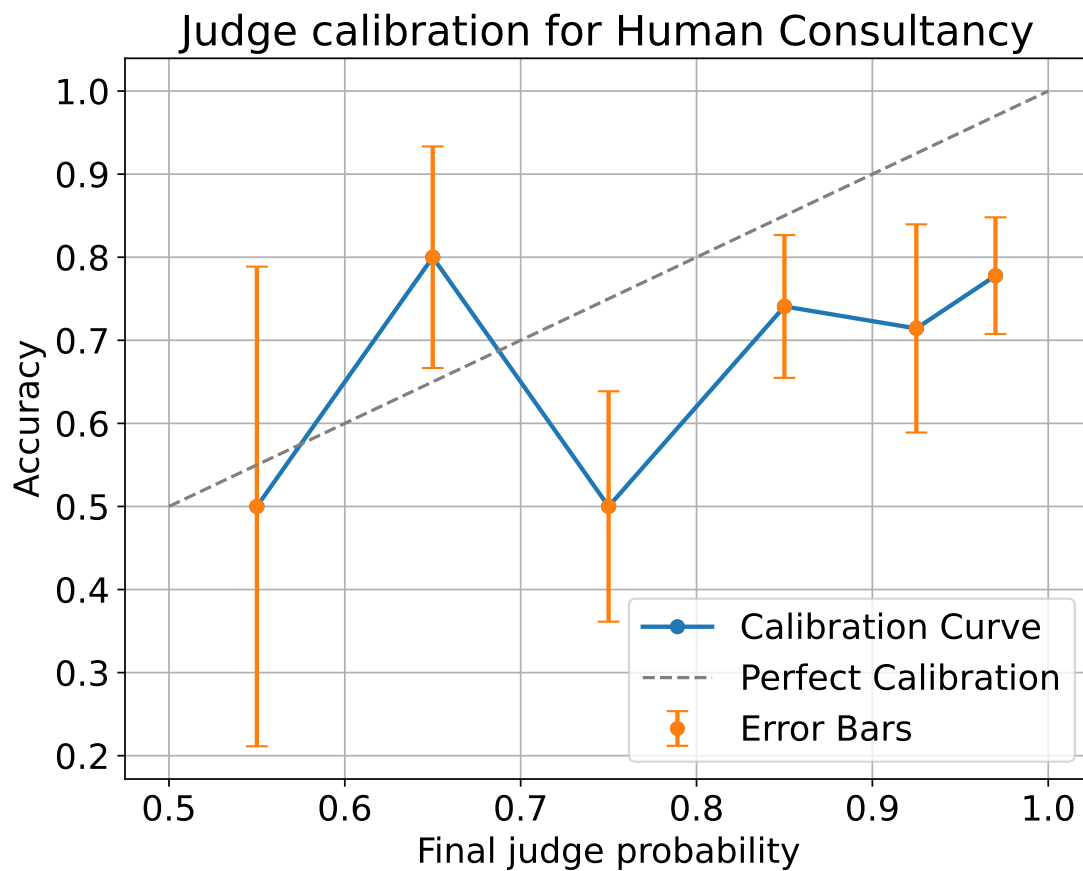
```











Judge Involvement

Judge Mistakes

Debater Skill

```
random.intercept.model = lmer(`Final probability correct` ~ (1|Final_Setting),
                              data = judgments, REML = TRUE)
```

```
judgments$random.intercept.preds = predict(random.intercept.model)
```

```
colnames(judgments)
```

```
## [1] "Participant"
## [2] "base_room_name"
## [3] "Room name"
## [4] "Room start time"
## [5] "Role"
## [6] "Is turn"
## [7] "Is over"
## [8] "Number of judge continues"
```

```

## [9] "Final probability correct"
## [10] "Offline judging start time"
## [11] "Offline judging end time"
## [12] "other"
## [13] "factual informativeness (comparative).1"
## [14] "factual informativeness (comparative).2"
## [15] "facts versus semantics (single)"
## [16] "factual accuracy (single)"
## [17] "clarity.1"
## [18] "clarity.2"
## [19] "factual accuracy.1"
## [20] "factual accuracy.2"
## [21] "judge reasoning"
## [22] "reason for outcome"
## [23] "protocol"
## [24] "evidence use.1"
## [25] "evidence use.2"
## [26] "evidence in story.1"
## [27] "evidence in story.2"
## [28] "other factors"
## [29] "judge adaptation (single)"
## [30] "evidence in debate.1"
## [31] "evidence in debate.2"
## [32] "interface"
## [33] "evidence in debate (single)"
## [34] "facts versus semantics.1"
## [35] "facts versus semantics.2"
## [36] "clash.1"
## [37] "clash.2"
## [38] "identity guesses.Judge"
## [39] "identity guesses.Debater A"
## [40] "identity guesses.Debater B"
## [41] "judge adaptation.1"
## [42] "judge adaptation.2"
## [43] "subjective correctness"
## [44] "evidence use (single)"
## [45] "factual informativeness (total)"
## [46] "judge strategies"
## [47] "clarity (single)"
## [48] "Debater A"
## [49] "Debater B"
## [50] "Honest debater"
## [51] "Dishonest debater"
## [52] "Is single debater"
## [53] "Has honest debater"
## [54] "Final_Setting"
## [55] "Setting"
## [56] "Question"
## [57] "Article ID"
## [58] "Speed annotator accuracy bins"
## [59] "Untimed annotator context bins"
## [60] "Speed annotator accuracy"
## [61] "Untimed annotator context"
## [62] "Is offline"

```

```
## [63] "End time"
## [64] "Last modified time"
## [65] "Final_Accuracy"
## [66] "random.intercept.preds"
```

```
dishonest <- judgments[!is.na(judgments$`Dishonest debater`), ]
model3 <- glm(Final_Accuracy ~ relevel(factor(`Dishonest debater`), 'Shlomo Kofman') + relevel(factor(Final_Setting), 'Human Debate'),
summary(model3)
```

```
##
## Call:
## glm(formula = Final_Accuracy ~ relevel(factor(`Dishonest debater`),
##      "Shlomo Kofman") + relevel(factor(Final_Setting), "Human Debate"),
##      family = "binomial", data = judgments[!is.na(judgments$`Dishonest debater`),
##      ])
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4478   0.3203   0.5083   0.6395   1.5320
##
## Coefficients: (1 not defined because of singularities)
##
## (Intercept)                                Estimate
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Adelle Fernando    0.52739
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Aliyaah Toussaint    2.41514
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Anuj Jain              1.47707
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")David Rein             1.41852
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Emmanuel Makinde    17.03868
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Ethan Rosen            1.45361
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")GPT-4                 0.75355
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Jackson Petty          2.08187
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Jessica Li             0.53268
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Julian Michael          2.41705
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Julien Dirani           1.55205
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Max Layden             17.03868
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Noor Mirza-Rashid     -0.05738
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Reeya Kansra           1.44916
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Salsabila Mahdi         1.47874
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Sam Jin                1.30012
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Sean Wang              1.43988
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Shreeram Modi           1.45605
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Vishakh Padmakumar     17.03868
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy                0.66498
## relevel(factor(Final_Setting), "Human Debate")AI Debate                     NA
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy             -1.33091
##
## Std. Error
## (Intercept)                                0.66115
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Adelle Fernando    0.73718
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Aliyaah Toussaint    1.23691
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Anuj Jain              0.84884
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")David Rein             0.90447
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Emmanuel Makinde    2797.44202
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")Ethan Rosen            0.84947
## relevel(factor(`Dishonest debater`), "Shlomo Kofman")GPT-4                 0.70782
```

```

## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jackson Petty      0.98698
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jessica Li          0.74081
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julian Michael       1.22055
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julien Dirani         1.24985
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Max Layden            3956.18038
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Noor Mirza-Rashid     0.87300
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Reeya Kansra          0.90748
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Salsabila Mahdi       0.79085
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sam Jin               0.93690
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sean Wang             0.75579
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Shreeram Modi         0.75586
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Vishakh Padmakumar    863.30958
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy              0.54080
## relevel(factor(Final_Setting), "Human Debate")AI Debate                   NA
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy           0.32388
##
## z value
## (Intercept)
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Adelle Fernando      1.297
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Aliyaah Toussaint     1.953
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Anuj Jain              1.740
## relevel(factor('Dishonest debater'), "Shlomo Kofman")David Rein             1.568
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Emmanuel Makinde       0.006
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Ethan Rosen            1.711
## relevel(factor('Dishonest debater'), "Shlomo Kofman")GPT-4                 1.065
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jackson Petty          2.109
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jessica Li             0.719
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julian Michael         1.980
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julien Dirani          1.242
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Max Layden             0.004
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Noor Mirza-Rashid     -0.066
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Reeya Kansra          1.597
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Salsabila Mahdi       1.870
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sam Jin               1.388
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sean Wang             1.905
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Shreeram Modi         1.926
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Vishakh Padmakumar    0.020
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy              1.230
## relevel(factor(Final_Setting), "Human Debate")AI Debate                   NA
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy           -4.109
##
## Pr(>|z|)
## (Intercept)
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Adelle Fernando      0.1948
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Aliyaah Toussaint     0.0509
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Anuj Jain              0.0818
## relevel(factor('Dishonest debater'), "Shlomo Kofman")David Rein             0.1168
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Emmanuel Makinde       0.9951
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Ethan Rosen            0.0870
## relevel(factor('Dishonest debater'), "Shlomo Kofman")GPT-4                 0.2871
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jackson Petty          0.0349
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jessica Li             0.4721
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julian Michael         0.0477
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julien Dirani          0.2143
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Max Layden             0.9966
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Noor Mirza-Rashid     0.9476

```

```

## relevel(factor('Dishonest debater'), "Shlomo Kofman")Reeya Kansra      0.1103
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Salsabila Mahdi    0.0615
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sam Jin             0.1652
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sean Wang          0.0568
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Shreeram Modi       0.0541
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Vishakh Padmakumar  0.9843
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy            0.2188
## relevel(factor(Final_Setting), "Human Debate")AI Debate                 NA
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy         0.0000397
##
## (Intercept)
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Adelle Fernando
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Aliyaah Toussaint .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Anuj Jain           .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")David Rein
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Emmanuel Makinde
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Ethan Rosen          .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")GPT-4
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jackson Petty      *
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Jessica Li
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julian Michael      *
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Julien Dirani
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Max Layden
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Noor Mirza-Rashid
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Reeya Kansra
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Salsabila Mahdi .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sam Jin
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Sean Wang           .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Shreeram Modi       .
## relevel(factor('Dishonest debater'), "Shlomo Kofman")Vishakh Padmakumar
## relevel(factor(Final_Setting), "Human Debate")AI Consultancy
## relevel(factor(Final_Setting), "Human Debate")AI Debate
## relevel(factor(Final_Setting), "Human Debate")Human Consultancy         ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 541.37 on 576 degrees of freedom
## Residual deviance: 487.85 on 555 degrees of freedom
## AIC: 531.85
##
## Number of Fisher Scoring iterations: 16

```

```

result <- judgments_online %>%
  group_by(`Dishonest debater`) %>%
  summarize(
    Win_Rate = sum(Final_Accuracy == "FALSE") / n()
  ) %>%
  ungroup() %>%
  arrange(desc(Win_Rate))

result

```

```
## # A tibble: 20 x 2
##   'Dishonest debater' Win_Rate
##   <chr>               <dbl>
## 1 Shlomo Kofman       0.545
## 2 Salsabila Mahdi     0.357
## 3 Jessica Li          0.353
## 4 Noor Mirza-Rashid   0.333
## 5 Adelle Fernando     0.296
## 6 Sean Wang           0.28
## 7 Reeya Kansra        0.273
## 8 Sam Jin             0.25
## 9 Shreeram Modi       0.24
## 10 GPT-4              0.192
## 11 <NA>               0.184
## 12 Anuj Jain          0.143
## 13 Julian Michael      0.125
## 14 Aliyaah Toussaint   0.111
## 15 Ethan Rosen         0.0909
## 16 Jackson Petty       0.0769
## 17 David Rein          0
## 18 Julien Dirani       0
## 19 Max Layden          0
## 20 Vishakh Padmakumar 0
```

```
result1 <- judgments_online %>%
  group_by(`Honest debater`) %>%
  summarize(
    Win_Rate = sum(Final_Accuracy == "TRUE") / n()
  ) %>%
  ungroup() %>%
  arrange(desc(Win_Rate))

result1
```

```
## # A tibble: 20 x 2
##   'Honest debater' Win_Rate
##   <chr>           <dbl>
## 1 Julian Michael   1
## 2 Julien Dirani    1
## 3 Noor Mirza-Rashid 1
## 4 Sean Wang       0.96
## 5 Jessica Li       0.923
## 6 Salsabila Mahdi  0.917
## 7 Adelle Fernando 0.905
## 8 Reeya Kansra     0.9
## 9 Vishakh Padmakumar 0.857
## 10 Shlomo Kofman    0.833
## 11 Anuj Jain        0.8
## 12 David Rein       0.8
## 13 Shreeram Modi    0.8
## 14 Ethan Rosen      0.786
## 15 GPT-4            0.775
## 16 <NA>             0.680
## 17 Jackson Petty    0.667
```

```
## 18 Sam Jin 0.667
## 19 Aliyaah Toussaint 0.625
## 20 Emmanuel Makinde 0

# Filter for high win rate debaters
high_win_rate_debaters <- result1 %>%
  filter(Win_Rate > 0.90) # Set the threshold for high win rate

# Filter original data for debates with 'Debate' in Final_Setting
filtered_data <- judgments_online %>%
  filter(grepl("Debate", Final_Setting))

# Find cases where high win rate debaters lost
cases_high_win_rate_lost <- filtered_data %>%
  filter(`Honest debater` %in% high_win_rate_debaters$`Honest debater` & Final_Accuracy != "TRUE")

cases_high_win_rate_lost
```

```
## Participant base_room_name Room name
## 146 Anuj Jain survival-type- survival-type-5
## 214 Ethan Rosen the-great-nebraska-sea- the-great-nebraska-sea-0
## 289 Jessica Li rx- rx-1
## Room start time Role Is turn Is over Number of judge continues
## 146 1681159356736 Judge FALSE TRUE 5
## 214 1683321454611 Judge FALSE TRUE 2
## 289 1683298141840 Judge FALSE TRUE 2
## Final probability correct Offline judging start time
## 146 0.33 NaN
## 214 0.01 NaN
## 289 0.01 NaN
## Offline judging end time other factual informativeness (comparative).1
## 146 NaN <NA> 4
## 214 NaN <NA> 1
## 289 NaN <NA> 4
## factual informativeness (comparative).2 facts versus semantics (single)
## 146 4 NaN
## 214 1 NaN
## 289 2 NaN
## factual accuracy (single) clarity.1 clarity.2 factual accuracy.1
## 146 NaN 3 3 NaN
## 214 NaN 2 2 NaN
## 289 NaN 4 1 NaN
## factual accuracy.2 judge reasoning
## 146 NaN 3
## 214 NaN 1
## 289 NaN 4
## reason for out
## 146 i
## 214 I thought "like" was over-technical compared to what these questions typically ask for. I was wr
## 289 B's last arg was literally 2 sentences, and A's ev was very convinci
## protocol evidence use.1 evidence use.2 evidence in story.1
## 146 <NA> NaN NaN NaN
## 214 <NA> NaN NaN NaN
## 289 <NA> NaN NaN NaN
```



```

##      evidence in story.2 other factors judge adaptation (single)
## 146      NaN      <NA>      NaN
## 214      NaN      <NA>      NaN
## 289      NaN      <NA>      NaN
##      evidence in debate.1 evidence in debate.2 interface
## 146      2      4      <NA>
## 214      3      2      <NA>
## 289      4      2      <NA>
##      evidence in debate (single) facts versus semantics.1
## 146      NaN      3
## 214      NaN      1
## 289      NaN      1
##      facts versus semantics.2 clash.1 clash.2 identity guesses.Judge
## 146      3      3      3      <NA>
## 214      3      4      4      <NA>
## 289      3      4      2      <NA>
##      identity guesses.Debater A identity guesses.Debater B judge adaptation.1
## 146      <NA>      <NA>      3
## 214      <NA>      <NA>      4
## 289      <NA>      <NA>      4
##      judge adaptation.2 subjective correctness evidence use (single)
## 146      3      NaN      NaN
## 214      3      NaN      NaN
## 289      2      NaN      NaN
##      factual informativeness (total) judge strategies clarity (single)
## 146      3      <NA>      NaN
## 214      1      <NA>      NaN
## 289      3      <NA>      NaN
##      Debater A      Debater B      Honest debater Dishonest debater
## 146 Adelle Fernando      Ethan Rosen Adelle Fernando      Ethan Rosen
## 214 Salsabila Mahdi      Sean Wang      Sean Wang      Salsabila Mahdi
## 289 Reeya Kansra Adelle Fernando Adelle Fernando      Reeya Kansra
##      Is single debater Has honest debater Final_Setting      Setting
## 146      FALSE      TRUE      Human Debate Human Debate
## 214      FALSE      TRUE      Human Debate Human Debate
## 289      FALSE      TRUE      Human Debate Human Debate
##
##      Question
## 146 How did the planet of Niobe compare to others that Earth was exploring?
## 214      How is this article written?
## 289 What were the specialties of the Red and Green Doctors, respectively?
##      Article ID Speed annotator accuracy bins Untimed annotator context bins
## 146      51395      0.2      3
## 214      50893      0.2      3
## 289      60412      0.2      3
##      Speed annotator accuracy Untimed annotator context Is offline
## 146      0.166667      2.750000      FALSE
## 214      0.200000      3.333333      FALSE
## 289      0.200000      2.800000      FALSE
##      End time      Last modified time      Final_Accuracy
## 146 2023-04-17 17:12:59 2023-04-28 12:29:25      FALSE
## 214 2023-05-08 17:14:28 2023-06-22 15:20:58      FALSE
## 289 2023-06-22 15:18:02 2023-06-22 15:18:02      FALSE
##      Human Consultancy Sample AI Consultancy Sample Human Debate Sample
## 146      FALSE      FALSE      FALSE

```

```

## 214          FALSE          FALSE          TRUE
## 289          FALSE          FALSE          TRUE
##      AI Debate Sample Sample Consultancy Sample initial_question_weights
## 146          FALSE FALSE          FALSE          0.5000000
## 214          FALSE  TRUE          TRUE          0.2000000
## 289          FALSE  TRUE          TRUE          0.3333333
##      initial_question_weights_grouped_setting
## 146                                0.5
## 214                                0.5
## 289                                0.5
##      sampled_consultancies_all_debates_weights
## 146                                0.5000000
## 214                                0.2500000
## 289                                0.3333333
##      sampled_consultancies_all_debates_weights_grouped_setting
## 146                                0.5
## 214                                0.5
## 289                                0.5
##      sampled_consultancies_all_debates_weights_setting
## 146                                0.5
## 214                                0.5
## 289                                0.5
##      sampled_consultancies_debates_weights_grouped_setting
## 146                                0
## 214                                1
## 289                                1
##      sampled_consultancies_debates_weights Final_Accuracy_char
## 146                                0.0000000      NA
## 214                                0.3333333      NA
## 289                                0.5000000      NA

```

```

# Filter for high win rate debaters
high_win_rate_debaters <- result %>%
  filter(Win_Rate > 0.20) # Set the threshold for high win rate

# Filter original data for debates with 'Debate' in Final_Setting
filtered_data <- judgments_online %>%
  filter(grepl("Debate", Final_Setting))

# Find cases where high win rate debaters lost
cases_high_win_rate_lost <- filtered_data %>%
  filter(`Dishonest debater` %in% high_win_rate_debaters$`Dishonest debater` & Final_Accuracy != "FALSE")

cases_high_win_rate_lost

```

```

##      Participant          base_room_name
## 21      Adelle Fernando      monopoly-
## 43      Adelle Fernando      tollivers-orbit-
## 78      Aliyaah Toussaint          rx-
## 81      Aliyaah Toussaint      stranger-from-space-
## 91      Aliyaah Toussaint      the-long-remembered-thunder-
## 94      Aliyaah Toussaint      the-princess-and-the-physicist-
## 99      Aliyaah Toussaint      the-starsent-knaves-
## 113      Anuj Jain          cosmic-yoyo-

```

## 136	Anuj Jain	out-of-the-iron-womb-
## 140	Anuj Jain	planet-of-dread-
## 149	Anuj Jain	the-air-of-castor-oil-
## 177	David Rein	monopoly-
## 179	David Rein	peggy-finds-the-theatre-
## 185	David Rein	stalemate-in-space-
## 186	David Rein	stranger-from-space-
## 191	David Rein	the-great-nebraska-sea-
## 202	Ethan Rosen	cosmic-yoyo-
## 211	Ethan Rosen	stranger-from-space-
## 215	Ethan Rosen	the-man-who-was-six-
## 216	Ethan Rosen	the-monster-maker-
## 219	Jackson Petty	atom-mystery-young-atom-detective-
## 236	Jackson Petty	muck-man-
## 240	Jackson Petty	rx-
## 241	Jackson Petty	silence-isdeadly-
## 254	Jackson Petty	the-princess-and-the-physicist-
## 270	Jessica Li	doctor-universe-
## 276	Jessica Li	how-to-make-friends-1
## 290	Jessica Li	silence-isdeadly-
## 306	Jessica Li	the-princess-and-the-physicist-
## 324	Julian Michael	monopoly-
## 331	Julian Michael	stranger-from-space-
## 332	Julian Michael	survival-type-
## 338	Julian Michael	the-monster-maker-
## 342	Julian Michael	the-spicy-sound-of-success-
## 348	Julien Dirani	manners-and-customs-
## 356	Noor Mirza-Rashid	doctor-universe-
## 366	Noor Mirza-Rashid	volpla-
## 378	Reeya Kansra	how-to-make-friends-
## 387	Reeya Kansra	muck-man-
## 401	Reeya Kansra	the-monster-maker-
## 411	Salsabila Mahdi	break-a-leg-
## 414	Salsabila Mahdi	cosmic-yoyo-
## 421	Salsabila Mahdi	manners-and-customs-
## 424	Salsabila Mahdi	muck-man-
## 425	Salsabila Mahdi	planet-of-dread-
## 429	Salsabila Mahdi	silence-isdeadly-
## 431	Salsabila Mahdi	stranger-from-space-
## 433	Salsabila Mahdi	the-happy-castaway-
## 436	Salsabila Mahdi	the-reluctant-heroes-
## 439	Salsabila Mahdi	the-starsent-knaves-
## 448	Sam Jin	coming-of-the-gods-
## 510	Sam Jin	venus-is-a-mans-world-
## 533	Sean Wang	lost-in-translation-
## 538	Sean Wang	peggy-finds-the-theatre-
## 544	Sean Wang	survival-type-
## 550	Sean Wang	the-cool-war-
## 561	Sean Wang	volpla-
## 598	Shlomo Kofman	out-of-the-iron-womb-
## 602	Shlomo Kofman	pied-piper-of-mars-
## 606	Shlomo Kofman	rx-
## 626	Shlomo Kofman	the-starbusters-
## 637	Shreeram Modi	cosmic-yoyo-

## 641	Shreeram Modi	in-the-garden-
## 647	Shreeram Modi	peggy-finds-the-theatre-
## 648	Shreeram Modi	phone-me-in-central-park-
## 658	Shreeram Modi	the-man-who-was-six-
## 677	Vishakh Padmakumar	stalemate-in-space-
## 679	Vishakh Padmakumar	the-air-of-castor-oil-
## 680	Vishakh Padmakumar	the-desert-and-the-stars-
## 683	Vishakh Padmakumar	the-monster-maker-
##	Room name	Room start time Role Is turn Is over
## 21	monopoly-1	1680552464768 Judge FALSE TRUE
## 43	tollivers-orbit-1	1681765942714 Judge FALSE TRUE
## 78	rx-3	1683298141840 Judge FALSE TRUE
## 81	stranger-from-space-0	1683298716462 Judge FALSE TRUE
## 91	the-long-remembered-thunder-1	1689876270711 Judge FALSE TRUE
## 94	the-princess-and-the-physicist-4	1682112300045 Judge FALSE TRUE
## 99	the-starsent-knives-2	1688757372245 Judge FALSE TRUE
## 113	cosmic-yoyo-0	1681159027164 Judge FALSE TRUE
## 136	out-of-the-iron-womb-0	1689876275997 Judge FALSE TRUE
## 140	planet-of-dread-2	1680829456935 Judge FALSE TRUE
## 149	the-air-of-castor-oil-5	1680552962919 Judge FALSE TRUE
## 177	monopoly-2	1680552464768 Judge FALSE TRUE
## 179	peggy-finds-the-theatre-4	1682110072206 Judge FALSE TRUE
## 185	stalemate-in-space-0	1677532762430 Judge FALSE TRUE
## 186	stranger-from-space-4	1683298716462 Judge FALSE TRUE
## 191	the-great-nebraska-sea-1	1683321454611 Judge FALSE TRUE
## 202	cosmic-yoyo-3	1681159027164 Judge FALSE TRUE
## 211	stranger-from-space-5	1683298716462 Judge FALSE TRUE
## 215	the-man-who-was-six-1	1676313105423 Judge FALSE TRUE
## 216	the-monster-maker-4	1681159292566 Judge FALSE TRUE
## 219	atom-mystery-young-atom-detective-0	1689949095893 Judge FALSE TRUE
## 236	muck-man-5	1687546720669 Judge FALSE TRUE
## 240	rx-4	1683298141840 Judge FALSE TRUE
## 241	silence-isdeadly-3	1688157095546 Judge FALSE TRUE
## 254	the-princess-and-the-physicist-0	1682112300045 Judge FALSE TRUE
## 270	doctor-universe-0	1680206097221 Judge FALSE TRUE
## 276	how-to-make-friends-11	1681724583153 Judge FALSE TRUE
## 290	silence-isdeadly-2	1688157095546 Judge FALSE TRUE
## 306	the-princess-and-the-physicist-2	1682112300045 Judge FALSE TRUE
## 324	monopoly-0	1680552464768 Judge FALSE TRUE
## 331	stranger-from-space-1	1683298716462 Judge FALSE TRUE
## 332	survival-type-4	1681159356736 Judge FALSE TRUE
## 338	the-monster-maker-3	1681159292566 Judge FALSE TRUE
## 342	the-spicy-sound-of-success-4	1679607458871 Judge FALSE TRUE
## 348	manners-and-customs-1	1676043334730 Judge FALSE TRUE
## 356	doctor-universe-5	1680206097221 Judge FALSE TRUE
## 366	volpla-2	1680205817615 Judge FALSE TRUE
## 378	how-to-make-friends-0	1681724583153 Judge FALSE TRUE
## 387	muck-man-7	1687546765239 Judge FALSE TRUE
## 401	the-monster-maker-1	1681159292566 Judge FALSE TRUE
## 411	break-a-leg-5	1682110823449 Judge FALSE TRUE
## 414	cosmic-yoyo-2	1681159027164 Judge FALSE TRUE
## 421	manners-and-customs-0	1676043281654 Judge FALSE TRUE
## 424	muck-man-4	1687546720669 Judge FALSE TRUE
## 425	planet-of-dread-1	1680829456935 Judge FALSE TRUE

## 429	silence-isdeadly-6	1688157095546	Judge	FALSE	TRUE
## 431	stranger-from-space-2	1683298716462	Judge	FALSE	TRUE
## 433	the-happy-castaway-2	1679606564549	Judge	FALSE	TRUE
## 436	the-reluctant-heroes-2	1682965111772	Judge	FALSE	TRUE
## 439	the-starsent-knaves-0	1688757372245	Judge	FALSE	TRUE
## 448	coming-of-the-gods-2	1689020073883	Judge	FALSE	TRUE
## 510	venus-is-a-mans-world-0	1691058680973	Judge	FALSE	TRUE
## 533	lost-in-translation-3	1678404069200	Judge	FALSE	TRUE
## 538	peggy-finds-the-theatre-0	1682090000149	Judge	FALSE	TRUE
## 544	survival-type-0	1681159356736	Judge	FALSE	TRUE
## 550	the-cool-war-0	1689949097911	Judge	FALSE	TRUE
## 561	volpla-3	1680205817615	Judge	FALSE	TRUE
## 598	out-of-the-iron-womb-1	1689876275999	Judge	FALSE	TRUE
## 602	pied-piper-of-mars-8	1689278492513	Judge	FALSE	TRUE
## 606	rx-5	1683298141840	Judge	FALSE	TRUE
## 626	the-starbusters-3	1689371609880	Judge	FALSE	TRUE
## 637	cosmic-yoyo-1	1681159027164	Judge	FALSE	TRUE
## 641	in-the-garden-6	1680206043370	Judge	FALSE	TRUE
## 647	peggy-finds-the-theatre-2	1682090000149	Judge	FALSE	TRUE
## 648	phone-me-in-central-park-5	1678684819928	Judge	FALSE	TRUE
## 658	the-man-who-was-six-5	1676645924826	Judge	FALSE	TRUE
## 677	stalemate-in-space-2	1677792427135	Judge	FALSE	TRUE
## 679	the-air-of-castor-oil-4	1680552962919	Judge	FALSE	TRUE
## 680	the-desert-and-the-stars-2	1677792315334	Judge	FALSE	TRUE
## 683	the-monster-maker-5	1681159292566	Judge	FALSE	TRUE
##	Number of judge continues Final probability correct				
## 21	4	0.70			
## 43	2	0.90			
## 78	1	0.99			
## 81	4	0.99			
## 91	3	0.98			
## 94	4	0.99			
## 99	4	0.85			
## 113	4	0.99			
## 136	4	0.99			
## 140	2	0.99			
## 149	3	0.85			
## 177	3	0.85			
## 179	4	0.90			
## 185	2	0.99			
## 186	4	0.95			
## 191	3	0.95			
## 202	2	0.90			
## 211	2	0.95			
## 215	2	0.80			
## 216	2	0.99			
## 219	6	0.80			
## 236	7	0.99			
## 240	3	0.90			
## 241	3	0.99			
## 254	4	0.95			
## 270	2	0.70			
## 276	2	0.99			
## 290	1	0.99			

## 306	2	0.99
## 324	3	0.99
## 331	2	0.99
## 332	2	0.99
## 338	3	0.99
## 342	4	0.99
## 348	3	0.85
## 356	4	0.85
## 366	3	0.95
## 378	3	0.98
## 387	4	0.88
## 401	2	0.96
## 411	2	0.99
## 414	2	0.99
## 421	3	0.99
## 424	3	0.99
## 425	3	0.99
## 429	4	0.99
## 431	2	0.99
## 433	3	0.99
## 436	4	0.99
## 439	6	0.95
## 448	3	0.99
## 510	3	0.99
## 533	2	0.98
## 538	2	0.90
## 544	1	0.98
## 550	3	0.99
## 561	2	0.95
## 598	1	0.94
## 602	4	0.91
## 606	4	0.86
## 626	3	0.97
## 637	4	0.95
## 641	2	0.99
## 647	1	0.99
## 648	2	0.99
## 658	3	0.99
## 677	3	0.80
## 679	2	0.75
## 680	3	0.75
## 683	5	0.80
##	Offline judging start time	Offline judging end time
## 21	NaN	NaN
## 43	NaN	NaN
## 78	NaN	NaN
## 81	NaN	NaN
## 91	NaN	NaN
## 94	NaN	NaN
## 99	NaN	NaN
## 113	NaN	NaN
## 136	NaN	NaN
## 140	NaN	NaN
## 149	NaN	NaN

## 177	NaN	NaN
## 179	NaN	NaN
## 185	NaN	NaN
## 186	NaN	NaN
## 191	NaN	NaN
## 202	NaN	NaN
## 211	NaN	NaN
## 215	NaN	NaN
## 216	NaN	NaN
## 219	NaN	NaN
## 236	NaN	NaN
## 240	NaN	NaN
## 241	NaN	NaN
## 254	NaN	NaN
## 270	NaN	NaN
## 276	NaN	NaN
## 290	NaN	NaN
## 306	NaN	NaN
## 324	NaN	NaN
## 331	NaN	NaN
## 332	NaN	NaN
## 338	NaN	NaN
## 342	NaN	NaN
## 348	NaN	NaN
## 356	NaN	NaN
## 366	NaN	NaN
## 378	NaN	NaN
## 387	NaN	NaN
## 401	NaN	NaN
## 411	NaN	NaN
## 414	NaN	NaN
## 421	NaN	NaN
## 424	NaN	NaN
## 425	NaN	NaN
## 429	NaN	NaN
## 431	NaN	NaN
## 433	NaN	NaN
## 436	NaN	NaN
## 439	NaN	NaN
## 448	NaN	NaN
## 510	NaN	NaN
## 533	NaN	NaN
## 538	NaN	NaN
## 544	NaN	NaN
## 550	NaN	NaN
## 561	NaN	NaN
## 598	NaN	NaN
## 602	NaN	NaN
## 606	NaN	NaN
## 626	NaN	NaN
## 637	NaN	NaN
## 641	NaN	NaN
## 647	NaN	NaN
## 648	1682713008576	1682713141741

## 658	NaN	NaN	
## 677	NaN	NaN	
## 679	NaN	NaN	
## 680	NaN	NaN	
## 683	NaN	NaN	
##			other
## 21			<NA>
## 43			<NA>
## 78			<NA>
## 81			<NA>
## 91			<NA>
## 94			<NA>
## 99			<NA>
## 113			<NA>
## 136			<NA>
## 140			<NA>
## 149			<NA>
## 177			<NA>
## 179			<NA>
## 185			<NA>
## 186			<NA>
## 191			<NA>
## 202			<NA>
## 211			<NA>
## 215			nope.
## 216			<NA>
## 219			<NA>
## 236			<NA>
## 240			<NA>
## 241			<NA>
## 254			<NA>
## 270			<NA>
## 276			<NA>
## 290			<NA>
## 306			<NA>
## 324			<NA>
## 331			<NA>
## 332	Maybe I could have decided sooner, even. but first round is a lot to go for.		
## 338			<NA>
## 342			<NA>
## 348			<NA>
## 356			<NA>
## 366			<NA>
## 378			<NA>
## 387			<NA>
## 401			<NA>
## 411			<NA>
## 414			<NA>
## 421			<NA>
## 424			<NA>
## 425			<NA>
## 429			<NA>
## 431			<NA>
## 433			<NA>



## 436		<NA>
## 439		<NA>
## 448		<NA>
## 510		<NA>
## 533		<NA>
## 538		<NA>
## 544		<NA>
## 550		<NA>
## 561		<NA>
## 598		<NA>
## 602		<NA>
## 606		<NA>
## 626		<NA>
## 637		<NA>
## 641		<NA>
## 647		<NA>
## 648		<NA>
## 658		<NA>
## 677		<NA>
## 679		<NA>
## 680		<NA>
## 683		<NA>
##	factual informativeness (comparative).1	
## 21	2	
## 43	2	
## 78	3	
## 81	3	
## 91	1	
## 94	4	
## 99	1	
## 113	2	
## 136	4	
## 140	4	
## 149	1	
## 177	3	
## 179	NaN	
## 185	2	
## 186	1	
## 191	1	
## 202	3	
## 211	4	
## 215	3	
## 216	2	
## 219	3	
## 236	3	
## 240	3	
## 241	4	
## 254	3	
## 270	2	
## 276	2	
## 290	2	
## 306	1	
## 324	2	
## 331	2	

## 332	1	
## 338	1	
## 342	3	
## 348	4	
## 356	2	
## 366	1	
## 378	4	
## 387	3	
## 401	4	
## 411	3	
## 414	3	
## 421	1	
## 424	3	
## 425	2	
## 429	3	
## 431	3	
## 433	3	
## 436	3	
## 439	3	
## 448	NaN	
## 510	NaN	
## 533	3	
## 538	4	
## 544	2	
## 550	3	
## 561	3	
## 598	4	
## 602	2	
## 606	2	
## 626	2	
## 637	3	
## 641	3	
## 647	3	
## 648	1	
## 658	2	
## 677	2	
## 679	2	
## 680	2	
## 683	0	
##	factual informativeness (comparative).2 facts versus semantics (single)	
## 21	2	NaN
## 43	2	NaN
## 78	4	NaN
## 81	3	NaN
## 91	3	NaN
## 94	2	NaN
## 99	3	NaN
## 113	2	NaN
## 136	3	NaN
## 140	3	NaN
## 149	3	NaN
## 177	3	NaN
## 179	NaN	NaN
## 185	2	NaN

## 186	1	NaN
## 191	1	NaN
## 202	4	NaN
## 211	2	NaN
## 215	2	NaN
## 216	2	NaN
## 219	3	NaN
## 236	3	NaN
## 240	3	NaN
## 241	4	NaN
## 254	3	NaN
## 270	3	NaN
## 276	3	NaN
## 290	4	NaN
## 306	0	NaN
## 324	4	NaN
## 331	3	NaN
## 332	4	NaN
## 338	4	NaN
## 342	4	NaN
## 348	4	NaN
## 356	1	NaN
## 366	2	NaN
## 378	4	NaN
## 387	4	NaN
## 401	4	NaN
## 411	3	NaN
## 414	3	NaN
## 421	3	NaN
## 424	3	NaN
## 425	2	NaN
## 429	2	NaN
## 431	3	NaN
## 433	3	NaN
## 436	3	NaN
## 439	3	NaN
## 448	NaN	NaN
## 510	NaN	NaN
## 533	2	NaN
## 538	4	NaN
## 544	2	NaN
## 550	3	NaN
## 561	3	NaN
## 598	2	NaN
## 602	2	NaN
## 606	3	NaN
## 626	4	NaN
## 637	3	NaN
## 641	1	NaN
## 647	3	NaN
## 648	3	NaN
## 658	3	NaN
## 677	2	NaN
## 679	2	NaN

## 680		1		NaN
## 683		3		NaN
##	factual accuracy (single)	clarity.1	clarity.2	factual accuracy.1
## 21	NaN	1	1	NaN
## 43	NaN	2	3	NaN
## 78	NaN	3	4	NaN
## 81	NaN	3	3	NaN
## 91	NaN	1	3	NaN
## 94	NaN	2	4	NaN
## 99	NaN	1	3	NaN
## 113	NaN	2	2	NaN
## 136	NaN	4	3	NaN
## 140	NaN	3	3	NaN
## 149	NaN	1	2	NaN
## 177	NaN	2	2	NaN
## 179	NaN	NaN	NaN	NaN
## 185	NaN	3	4	NaN
## 186	NaN	3	3	NaN
## 191	NaN	2	2	NaN
## 202	NaN	4	4	NaN
## 211	NaN	4	1	NaN
## 215	NaN	4	4	NaN
## 216	NaN	4	4	NaN
## 219	NaN	3	3	NaN
## 236	NaN	2	3	NaN
## 240	NaN	3	3	NaN
## 241	NaN	4	4	NaN
## 254	NaN	3	2	NaN
## 270	NaN	4	4	NaN
## 276	NaN	3	4	NaN
## 290	NaN	3	4	NaN
## 306	NaN	1	0	NaN
## 324	NaN	0	3	NaN
## 331	NaN	3	4	NaN
## 332	NaN	3	4	NaN
## 338	NaN	1	4	NaN
## 342	NaN	1	2	NaN
## 348	NaN	4	4	NaN
## 356	NaN	1	1	NaN
## 366	NaN	2	2	NaN
## 378	NaN	4	4	NaN
## 387	NaN	4	4	NaN
## 401	NaN	4	4	NaN
## 411	NaN	3	3	NaN
## 414	NaN	3	3	NaN
## 421	NaN	2	3	NaN
## 424	NaN	3	3	NaN
## 425	NaN	3	3	NaN
## 429	NaN	3	3	NaN
## 431	NaN	3	3	NaN
## 433	NaN	3	3	NaN
## 436	NaN	3	3	NaN
## 439	NaN	3	3	NaN
## 448	NaN	NaN	NaN	NaN

## 510	NaN	NaN	NaN	NaN
## 533	NaN	2	3	NaN
## 538	NaN	4	4	NaN
## 544	NaN	4	4	NaN
## 550	NaN	4	4	NaN
## 561	NaN	3	3	NaN
## 598	NaN	1	4	NaN
## 602	NaN	3	2	NaN
## 606	NaN	2	3	NaN
## 626	NaN	1	4	NaN
## 637	NaN	2	2	NaN
## 641	NaN	2	1	NaN
## 647	NaN	3	3	NaN
## 648	NaN	2	3	NaN
## 658	NaN	2	2	NaN
## 677	NaN	3	2	NaN
## 679	NaN	3	2	NaN
## 680	NaN	3	1	NaN
## 683	NaN	0	3	NaN
##	factual accuracy.2 judge reasoning			
## 21	NaN	3		
## 43	NaN	3		
## 78	NaN	4		
## 81	NaN	3		
## 91	NaN	4		
## 94	NaN	4		
## 99	NaN	4		
## 113	NaN	2		
## 136	NaN	4		
## 140	NaN	4		
## 149	NaN	4		
## 177	NaN	3		
## 179	NaN	NaN		
## 185	NaN	4		
## 186	NaN	4		
## 191	NaN	4		
## 202	NaN	4		
## 211	NaN	4		
## 215	NaN	4		
## 216	NaN	4		
## 219	NaN	3		
## 236	NaN	4		
## 240	NaN	4		
## 241	NaN	4		
## 254	NaN	3		
## 270	NaN	4		
## 276	NaN	4		
## 290	NaN	4		
## 306	NaN	4		
## 324	NaN	4		
## 331	NaN	4		
## 332	NaN	4		
## 338	NaN	4		
## 342	NaN	4		

## 348	NaN	4
## 356	NaN	4
## 366	NaN	3
## 378	NaN	4
## 387	NaN	4
## 401	NaN	4
## 411	NaN	3
## 414	NaN	3
## 421	NaN	NaN
## 424	NaN	3
## 425	NaN	3
## 429	NaN	3
## 431	NaN	3
## 433	NaN	3
## 436	NaN	4
## 439	NaN	3
## 448	NaN	NaN
## 510	NaN	NaN
## 533	NaN	3
## 538	NaN	4
## 544	NaN	4
## 550	NaN	4
## 561	NaN	4
## 598	NaN	4
## 602	NaN	3
## 606	NaN	4
## 626	NaN	4
## 637	NaN	3
## 641	NaN	3
## 647	NaN	3
## 648	NaN	3
## 658	NaN	2
## 677	NaN	3
## 679	NaN	2
## 680	NaN	4
## 683	NaN	3
##		
## 21		
## 43		
## 78		
## 81		
## 91		
## 94		
## 99		
## 113		
## 136		
## 140		
## 149		
## 177		
## 179		
## 185		
## 186		
## 191		
## 202		

## 211  
## 215  
## 216  
## 219  
## 236  
## 240  
## 241  
## 254  
## 270  
## 276  
## 290  
## 306  
## 324  
## 331  
## 332  
## 338  
## 342  
## 348  
## 356  
## 366  
## 378  
## 387  
## 401  
## 411  
## 414  
## 421  
## 424  
## 425  
## 429  
## 431  
## 433  
## 436  
## 439  
## 448  
## 510  
## 533  
## 538  
## 544  
## 550  
## 561  
## 598  
## 602  
## 606  
## 626  
## 637  
## 641  
## 647  
## 648  
## 658  
## 677  
## 679  
## 680

I think I continued the debate for an extra round just to see if any

Accidentally voted :

## 683 I think the factor which convinces me is that the evidence presented seems compelling that the m  
## protocol evidence use.1 evidence use.2 evidence in story.1

## 21	<NA>	NaN	NaN	NaN
## 43	<NA>	NaN	NaN	NaN
## 78	<NA>	NaN	NaN	NaN
## 81	<NA>	NaN	NaN	NaN
## 91	<NA>	NaN	NaN	NaN
## 94	<NA>	NaN	NaN	NaN
## 99	<NA>	NaN	NaN	NaN
## 113	<NA>	NaN	NaN	NaN
## 136	<NA>	NaN	NaN	NaN
## 140	<NA>	NaN	NaN	NaN
## 149	<NA>	NaN	NaN	NaN
## 177	<NA>	NaN	NaN	NaN
## 179	<NA>	NaN	NaN	NaN
## 185	<NA>	NaN	NaN	NaN
## 186	<NA>	NaN	NaN	NaN
## 191	<NA>	NaN	NaN	NaN
## 202	<NA>	NaN	NaN	NaN
## 211	<NA>	NaN	NaN	NaN
## 215	nope.	NaN	NaN	NaN
## 216	<NA>	NaN	NaN	NaN
## 219	<NA>	NaN	NaN	NaN
## 236	<NA>	NaN	NaN	NaN
## 240	<NA>	NaN	NaN	NaN
## 241	<NA>	NaN	NaN	NaN
## 254	<NA>	NaN	NaN	NaN
## 270	<NA>	NaN	NaN	NaN
## 276	<NA>	NaN	NaN	NaN
## 290	<NA>	NaN	NaN	NaN
## 306	<NA>	NaN	NaN	NaN
## 324	<NA>	NaN	NaN	NaN
## 331	<NA>	NaN	NaN	NaN
## 332	<NA>	NaN	NaN	NaN
## 338	<NA>	NaN	NaN	NaN
## 342	<NA>	NaN	NaN	NaN
## 348	<NA>	NaN	NaN	NaN
## 356	<NA>	NaN	NaN	NaN
## 366	<NA>	NaN	NaN	NaN
## 378	<NA>	NaN	NaN	NaN
## 387	<NA>	NaN	NaN	NaN
## 401	<NA>	NaN	NaN	NaN
## 411	<NA>	NaN	NaN	NaN
## 414	<NA>	NaN	NaN	NaN
## 421	<NA>	NaN	NaN	NaN
## 424	<NA>	NaN	NaN	NaN
## 425	<NA>	NaN	NaN	NaN
## 429	<NA>	NaN	NaN	NaN
## 431	<NA>	NaN	NaN	NaN
## 433	<NA>	NaN	NaN	NaN
## 436	<NA>	NaN	NaN	NaN
## 439	<NA>	NaN	NaN	NaN
## 448	<NA>	NaN	NaN	NaN
## 510	<NA>	NaN	NaN	NaN
## 533	<NA>	NaN	NaN	NaN
## 538	<NA>	NaN	NaN	NaN



## 544	<NA>	NaN	NaN	NaN
## 550	<NA>	NaN	NaN	NaN
## 561	<NA>	NaN	NaN	NaN
## 598	<NA>	NaN	NaN	NaN
## 602	<NA>	NaN	NaN	NaN
## 606	<NA>	NaN	NaN	NaN
## 626	<NA>	NaN	NaN	NaN
## 637	<NA>	NaN	NaN	NaN
## 641	<NA>	NaN	NaN	NaN
## 647	<NA>	NaN	NaN	NaN
## 648	<NA>	NaN	NaN	NaN
## 658	<NA>	NaN	NaN	NaN
## 677	<NA>	NaN	NaN	NaN
## 679	<NA>	NaN	NaN	NaN
## 680	<NA>	NaN	NaN	NaN
## 683	<NA>	NaN	NaN	NaN
##	evidence in story.2			
## 21		NaN		
## 43		NaN		
## 78		NaN		
## 81		NaN		
## 91		NaN		
## 94		NaN		
## 99		NaN		
## 113		NaN		
## 136		NaN		
## 140		NaN		
## 149		NaN		
## 177		NaN		
## 179		NaN		
## 185		NaN		
## 186		NaN		
## 191		NaN		
## 202		NaN		
## 211		NaN		
## 215		NaN		
## 216		NaN		
## 219		NaN		
## 236		NaN		
## 240		NaN		
## 241		NaN		
## 254		NaN		
## 270		NaN		
## 276		NaN		
## 290		NaN		
## 306		NaN		
## 324		NaN		
## 331		NaN		
## 332		NaN		
## 338		NaN		
## 342		NaN		
## 348		NaN		
## 356		NaN		
## 366		NaN		

## 378	NaN
## 387	NaN
## 401	NaN
## 411	NaN
## 414	NaN
## 421	NaN
## 424	NaN
## 425	NaN
## 429	NaN
## 431	NaN
## 433	NaN
## 436	NaN
## 439	NaN
## 448	NaN
## 510	NaN
## 533	NaN
## 538	NaN
## 544	NaN
## 550	NaN
## 561	NaN
## 598	NaN
## 602	NaN
## 606	NaN
## 626	NaN
## 637	NaN
## 641	NaN
## 647	NaN
## 648	NaN
## 658	NaN
## 677	NaN
## 679	NaN
## 680	NaN
## 683	NaN
##	
## 21	
## 43	
## 78	
## 81	
## 91	
## 94	
## 99	
## 113	
## 136	
## 140	
## 149	
## 177	
## 179	
## 185	
## 186	
## 191	
## 202	
## 211	
## 215	
## 216	

```

## 219
## 236
## 240
## 241
## 254
## 270
## 276
## 290
## 306
## 324
## 331
## 332
## 338
## 342
## 348
## 356
## 366
## 378
## 387
## 401
## 411
## 414
## 421
## 424
## 425
## 429
## 431
## 433
## 436
## 439
## 448
## 510
## 533
## 538
## 544
## 550
## 561
## 598
## 602
## 606
## 626
## 637
## 641
## 647
## 648
## 658
## 677
## 679 I definitely dropped the ball here and got back to judging the debate after a few weeks. I think
## 680 I sensed towards the end that the dishonest debate.
## 683
## judge adaptation (single) evidence in debate.1 evidence in debate.2
## 21 NaN 2 2
## 43 NaN 2 3
## 78 NaN 1 4

```

## 81	NaN	3	3
## 91	NaN	2	3
## 94	NaN	1	3
## 99	NaN	1	4
## 113	NaN	2	2
## 136	NaN	4	3
## 140	NaN	4	3
## 149	NaN	2	3
## 177	NaN	3	2
## 179	NaN	NaN	NaN
## 185	NaN	0	3
## 186	NaN	3	2
## 191	NaN	2	3
## 202	NaN	3	4
## 211	NaN	4	0
## 215	NaN	4	1
## 216	NaN	0	4
## 219	NaN	3	2
## 236	NaN	2	4
## 240	NaN	4	2
## 241	NaN	2	4
## 254	NaN	3	2
## 270	NaN	2	4
## 276	NaN	2	4
## 290	NaN	2	4
## 306	NaN	2	0
## 324	NaN	0	3
## 331	NaN	1	4
## 332	NaN	0	3
## 338	NaN	0	3
## 342	NaN	3	4
## 348	NaN	2	4
## 356	NaN	2	1
## 366	NaN	1	3
## 378	NaN	4	4
## 387	NaN	2	4
## 401	NaN	4	4
## 411	NaN	3	3
## 414	NaN	3	2
## 421	NaN	1	3
## 424	NaN	2	3
## 425	NaN	3	3
## 429	NaN	3	3
## 431	NaN	4	3
## 433	NaN	3	2
## 436	NaN	3	3
## 439	NaN	2	3
## 448	NaN	NaN	NaN
## 510	NaN	NaN	NaN
## 533	NaN	3	1
## 538	NaN	1	4
## 544	NaN	0	3
## 550	NaN	4	3
## 561	NaN	2	4

## 598	NaN	4	2
## 602	NaN	2	2
## 606	NaN	1	4
## 626	NaN	1	4
## 637	NaN	2	2
## 641	NaN	3	1
## 647	NaN	1	3
## 648	NaN	2	4
## 658	NaN	2	2
## 677	NaN	4	1
## 679	NaN	3	1
## 680	NaN	3	1
## 683	NaN	4	2
##			interface
## 21			<NA>
## 43			<NA>
## 78			<NA>
## 81			<NA>
## 91			<NA>
## 94			<NA>
## 99			<NA>
## 113			<NA>
## 136			<NA>
## 140			<NA>
## 149			<NA>
## 177			<NA>
## 179			<NA>
## 185	I accidentally entered the probabilities backwards		
## 186			<NA>
## 191			<NA>
## 202			<NA>
## 211			<NA>
## 215	The interface is great!		
## 216			<NA>
## 219			<NA>
## 236			<NA>
## 240			<NA>
## 241			<NA>
## 254			<NA>
## 270			<NA>
## 276			<NA>
## 290			<NA>
## 306			<NA>
## 324			<NA>
## 331			<NA>
## 332			<NA>
## 338			<NA>
## 342			<NA>
## 348			<NA>
## 356			<NA>
## 366			<NA>
## 378			<NA>
## 387			<NA>
## 401			<NA>

## 411			<NA>
## 414			<NA>
## 421			<NA>
## 424			<NA>
## 425			<NA>
## 429			<NA>
## 431			<NA>
## 433			<NA>
## 436			<NA>
## 439			<NA>
## 448			<NA>
## 510			<NA>
## 533			<NA>
## 538			<NA>
## 544			<NA>
## 550			<NA>
## 561			<NA>
## 598			<NA>
## 602			<NA>
## 606			<NA>
## 626			<NA>
## 637			<NA>
## 641			<NA>
## 647			<NA>
## 648			<NA>
## 658			<NA>
## 677	Quote limits seemed to hamper both debaters? Unclear if they agree		
## 679			<NA>
## 680			<NA>
## 683			<NA>
##	evidence in debate (single) facts versus semantics.1		
## 21	NaN	3	
## 43	NaN	3	
## 78	NaN	2	
## 81	NaN	1	
## 91	NaN	3	
## 94	NaN	2	
## 99	NaN	2	
## 113	NaN	2	
## 136	NaN	4	
## 140	NaN	1	
## 149	NaN	3	
## 177	NaN	1	
## 179	NaN	NaN	
## 185	NaN	2	
## 186	NaN	2	
## 191	NaN	3	
## 202	NaN	0	
## 211	NaN	0	
## 215	NaN	0	
## 216	NaN	0	
## 219	NaN	2	
## 236	NaN	1	
## 240	NaN	1	

## 241	NaN	0
## 254	NaN	1
## 270	NaN	2
## 276	NaN	2
## 290	NaN	3
## 306	NaN	4
## 324	NaN	0
## 331	NaN	0
## 332	NaN	0
## 338	NaN	0
## 342	NaN	0
## 348	NaN	3
## 356	NaN	1
## 366	NaN	2
## 378	NaN	3
## 387	NaN	4
## 401	NaN	1
## 411	NaN	3
## 414	NaN	1
## 421	NaN	1
## 424	NaN	1
## 425	NaN	2
## 429	NaN	1
## 431	NaN	1
## 433	NaN	1
## 436	NaN	2
## 439	NaN	2
## 448	NaN	NaN
## 510	NaN	NaN
## 533	NaN	2
## 538	NaN	3
## 544	NaN	3
## 550	NaN	2
## 561	NaN	2
## 598	NaN	2
## 602	NaN	2
## 606	NaN	2
## 626	NaN	0
## 637	NaN	1
## 641	NaN	1
## 647	NaN	3
## 648	NaN	2
## 658	NaN	3
## 677	NaN	1
## 679	NaN	2
## 680	NaN	2
## 683	NaN	1
##	facts versus semantics.2 clash.1 clash.2 identity guesses.Judge	
## 21	3 1 2	<NA>
## 43	2 2 2	<NA>
## 78	1 1 4	<NA>
## 81	1 3 3	<NA>
## 91	1 0 3	<NA>
## 94	1 1 4	<NA>

## 99	3	1	3	<NA>
## 113	2	2	2	<NA>
## 136	3	4	3	<NA>
## 140	3	3	2	<NA>
## 149	1	2	2	<NA>
## 177	3	2	2	<NA>
## 179	NaN	NaN	NaN	<NA>
## 185	1	2	3	<NA>
## 186	1	4	2	<NA>
## 191	3	3	3	<NA>
## 202	0	3	4	<NA>
## 211	4	4	1	<NA>
## 215	1	3	2	<NA>
## 216	0	0	4	<NA>
## 219	2	3	2	<NA>
## 236	1	3	3	<NA>
## 240	1	4	3	<NA>
## 241	0	3	4	<NA>
## 254	3	3	2	<NA>
## 270	0	2	2	<NA>
## 276	0	2	4	<NA>
## 290	0	1	2	<NA>
## 306	3	1	0	<NA>
## 324	0	1	4	<NA>
## 331	0	1	4	<NA>
## 332	0	0	4	<NA>
## 338	0	0	4	<NA>
## 342	0	2	4	<NA>
## 348	1	4	4	<NA>
## 356	2	1	1	<NA>
## 366	1	2	3	<NA>
## 378	0	4	4	<NA>
## 387	4	4	4	<NA>
## 401	0	4	4	<NA>
## 411	2	3	3	<NA>
## 414	2	3	3	<NA>
## 421	1	2	3	<NA>
## 424	1	3	3	<NA>
## 425	2	3	3	<NA>
## 429	1	3	2	<NA>
## 431	2	3	2	<NA>
## 433	1	3	3	<NA>
## 436	2	3	3	<NA>
## 439	1	2	2	<NA>
## 448	NaN	NaN	NaN	<NA>
## 510	NaN	NaN	NaN	<NA>
## 533	1	3	3	<NA>
## 538	3	4	2	<NA>
## 544	0	0	0	<NA>
## 550	1	4	2	<NA>
## 561	0	4	4	<NA>
## 598	1	2	2	<NA>
## 602	2	3	3	<NA>
## 606	2	3	2	<NA>



## 626	0	0	4	<NA>
## 637	3	3	3	<NA>
## 641	3	2	0	<NA>
## 647	2	2	3	<NA>
## 648	1	2	4	<NA>
## 658	3	1	3	<NA>
## 677	3	4	2	<NA>
## 679	3	3	2	<NA>
## 680	3	4	2	<NA>
## 683	3	1	3	<NA>
##	identity guesses.Debater A identity guesses.Debater B			judge adaptation.1
## 21	<NA>		<NA>	1
## 43	<NA>		<NA>	2
## 78	<NA>		<NA>	2
## 81	<NA>		<NA>	3
## 91	<NA>		<NA>	2
## 94	<NA>		<NA>	1
## 99	<NA>		<NA>	0
## 113	<NA>		<NA>	2
## 136	<NA>		<NA>	4
## 140	<NA>		<NA>	3
## 149	Emmanuel Makinde		<NA>	1
## 177	<NA>		<NA>	1
## 179	<NA>		<NA>	NaN
## 185	<NA>		<NA>	2
## 186	<NA>		<NA>	2
## 191	<NA>		<NA>	4
## 202	<NA>		<NA>	2
## 211	<NA>		<NA>	4
## 215	<NA>		<NA>	4
## 216	<NA>		<NA>	2
## 219	<NA>		<NA>	3
## 236	<NA>		<NA>	3
## 240	<NA>		<NA>	4
## 241	<NA>		<NA>	2
## 254	<NA>		<NA>	3
## 270	<NA>		<NA>	3
## 276	<NA>		<NA>	2
## 290	<NA>		<NA>	2
## 306	<NA>		<NA>	1
## 324	Reeya Kansra		Sean Wang	2
## 331	Reeya Kansra		Sean Wang	0
## 332	<NA>		<NA>	4
## 338	<NA>		<NA>	0
## 342	<NA>		<NA>	2
## 348	<NA>		<NA>	3
## 356	<NA>		<NA>	2
## 366	<NA>		<NA>	3
## 378	Jessica Li		Adelle Fernando	4
## 387	Julien Dirani		Ethan Rosen	4
## 401	Emmanuel Makinde		Adelle Fernando	4
## 411	<NA>		<NA>	3
## 414	<NA>		<NA>	2
## 421	<NA>		<NA>	3

## 424	Shlomo Kofman	Sam Jin	3
## 425	Jessica Li	Anuj Jain	3
## 429	Jessica Li	Shreeram Modi	3
## 431	<NA>	<NA>	3
## 433	<NA>	<NA>	3
## 436	<NA>	<NA>	4
## 439	Sean Wang	Reeya Kansra	3
## 448	<NA>	<NA>	NaN
## 510	<NA>	<NA>	NaN
## 533	<NA>	<NA>	2
## 538	<NA>	<NA>	4
## 544	<NA>	<NA>	2
## 550	<NA>	<NA>	3
## 561	<NA>	<NA>	3
## 598	<NA>	<NA>	2
## 602	<NA>	<NA>	3
## 606	<NA>	<NA>	2
## 626	<NA>	<NA>	0
## 637	<NA>	<NA>	3
## 641	<NA>	<NA>	2
## 647	<NA>	<NA>	2
## 648	<NA>	<NA>	1
## 658	<NA>	<NA>	1
## 677	<NA>	<NA>	4
## 679	<NA>	<NA>	3
## 680	<NA>	<NA>	4
## 683	<NA>	<NA>	1
##	judge adaptation.2 subjective correctness evidence use (single)		
## 21	1	NaN	NaN
## 43	3	NaN	NaN
## 78	4	NaN	NaN
## 81	3	NaN	NaN
## 91	3	NaN	NaN
## 94	4	NaN	NaN
## 99	4	NaN	NaN
## 113	2	NaN	NaN
## 136	3	NaN	NaN
## 140	2	NaN	NaN
## 149	2	NaN	NaN
## 177	1	NaN	NaN
## 179	NaN	NaN	NaN
## 185	2	NaN	NaN
## 186	2	NaN	NaN
## 191	4	NaN	NaN
## 202	4	NaN	NaN
## 211	1	NaN	NaN
## 215	4	NaN	NaN
## 216	4	NaN	NaN
## 219	2	NaN	NaN
## 236	4	NaN	NaN
## 240	2	NaN	NaN
## 241	4	NaN	NaN
## 254	1	NaN	NaN
## 270	4	NaN	NaN

## 276	4	NaN	NaN
## 290	2	NaN	NaN
## 306	0	NaN	NaN
## 324	4	NaN	NaN
## 331	4	NaN	NaN
## 332	4	NaN	NaN
## 338	4	NaN	NaN
## 342	3	NaN	NaN
## 348	3	NaN	NaN
## 356	2	NaN	NaN
## 366	3	NaN	NaN
## 378	4	NaN	NaN
## 387	4	NaN	NaN
## 401	4	NaN	NaN
## 411	3	NaN	NaN
## 414	3	NaN	NaN
## 421	4	NaN	NaN
## 424	3	NaN	NaN
## 425	3	NaN	NaN
## 429	2	NaN	NaN
## 431	3	NaN	NaN
## 433	2	NaN	NaN
## 436	3	NaN	NaN
## 439	3	NaN	NaN
## 448	NaN	NaN	NaN
## 510	NaN	NaN	NaN
## 533	3	NaN	NaN
## 538	4	NaN	NaN
## 544	2	NaN	NaN
## 550	3	NaN	NaN
## 561	3	NaN	NaN
## 598	2	NaN	NaN
## 602	3	NaN	NaN
## 606	3	NaN	NaN
## 626	4	NaN	NaN
## 637	3	NaN	NaN
## 641	1	NaN	NaN
## 647	2	NaN	NaN
## 648	4	NaN	NaN
## 658	3	NaN	NaN
## 677	1	NaN	NaN
## 679	0	NaN	NaN
## 680	2	NaN	NaN
## 683	3	NaN	NaN
##	factual informativeness (total)		
## 21	1		
## 43	2		
## 78	3		
## 81	3		
## 91	3		
## 94	3		
## 99	3		
## 113	2		
## 136	4		

## 140	3
## 149	3
## 177	1
## 179	NaN
## 185	1
## 186	1
## 191	1
## 202	4
## 211	3
## 215	2
## 216	0
## 219	3
## 236	4
## 240	4
## 241	4
## 254	3
## 270	3
## 276	4
## 290	3
## 306	0
## 324	4
## 331	3
## 332	4
## 338	3
## 342	3
## 348	3
## 356	2
## 366	2
## 378	4
## 387	4
## 401	4
## 411	3
## 414	3
## 421	3
## 424	3
## 425	3
## 429	3
## 431	3
## 433	3
## 436	4
## 439	3
## 448	NaN
## 510	NaN
## 533	3
## 538	4
## 544	0
## 550	3
## 561	4
## 598	4
## 602	3
## 606	3
## 626	4
## 637	3
## 641	2

## 647  
## 648  
## 658  
## 677  
## 679  
## 680  
## 683  
##  
## 21  
## 43  
## 78  
## 81  
## 91  
## 94  
## 99  
## 113  
## 136  
## 140  
## 149  
## 177  
## 179  
## 185  
## 186  
## 191  
## 202  
## 211  
## 215  
## 216  
## 219  
## 236  
## 240  
## 241  
## 254  
## 270  
## 276  
## 290  
## 306  
## 324  
## 331  
## 332  
## 338  
## 342  
## 348  
## 356  
## 366  
## 378  
## 387  
## 401  
## 411  
## 414  
## 421  
## 424  
## 425  
## 429

3  
3  
3  
1  
2  
3  
3

I said this to debater A: Are there any other resources mentioned, or context

```

## 431
## 433
## 436
## 439
## 448
## 510
## 533
## 538
## 544
## 550
## 561
## 598
## 602
## 606
## 626
## 637
## 641
## 647
## 648
## 658 Yes. I indicated particular pieces of evidence that both were missing and that would help me gre
## 677
## 679
## 680
## 683
##      clarity (single)      Debater A      Debater B      Honest debater
## 21      NaN      Ethan Rosen      Sean Wang      Ethan Rosen
## 43      NaN      Jessica Li      Ethan Rosen      Ethan Rosen
## 78      NaN      Reeya Kansra      Julian Michael      Julian Michael
## 81      NaN      Shreeram Modi      Sean Wang      Shreeram Modi
## 91      NaN      Shlomo Kofman      Sean Wang      Sean Wang
## 94      NaN      Sean Wang      Anuj Jain      Anuj Jain
## 99      NaN      Adelle Fernando      Shreeram Modi      Shreeram Modi
## 113     NaN      Noor Mirza-Rashid      Sean Wang      Noor Mirza-Rashid
## 136     NaN      Shreeram Modi      Adelle Fernando      Shreeram Modi
## 140     NaN      Reeya Kansra      Jessica Li      Reeya Kansra
## 149     NaN      Salsabila Mahdi      Jessica Li      Jessica Li
## 177     NaN      Ethan Rosen      Reeya Kansra      Ethan Rosen
## 179     NaN      Reeya Kansra      Jackson Petty      Jackson Petty
## 185     NaN      Shreeram Modi      Ethan Rosen      Ethan Rosen
## 186     NaN      Shreeram Modi      Adelle Fernando      Shreeram Modi
## 191     NaN      Sean Wang      Salsabila Mahdi      Salsabila Mahdi
## 202     NaN      Adelle Fernando      Sean Wang      Sean Wang
## 211     NaN      Sean Wang      Shreeram Modi      Sean Wang
## 215     NaN      David Rein      Sean Wang      David Rein
## 216     NaN      Noor Mirza-Rashid      Shreeram Modi      Shreeram Modi
## 219     NaN      Anuj Jain      Sam Jin      Anuj Jain
## 236     NaN      Sam Jin      Shlomo Kofman      Shlomo Kofman
## 240     NaN      Adelle Fernando      Reeya Kansra      Adelle Fernando
## 241     NaN      Sam Jin      Anuj Jain      Anuj Jain
## 254     NaN      Anuj Jain      Reeya Kansra      Anuj Jain
## 270     NaN      Reeya Kansra      Anuj Jain      Anuj Jain
## 276     NaN      Adelle Fernando      Ethan Rosen      Ethan Rosen
## 290     NaN      Adelle Fernando      Sam Jin      Sam Jin
## 306     NaN      Anuj Jain      Sean Wang      Anuj Jain

```

## 324	NaN	Reeya Kansra	Sean Wang	Sean Wang
## 331	NaN	Shreeram Modi	Sean Wang	Sean Wang
## 332	NaN	Adelle Fernando	Ethan Rosen	Ethan Rosen
## 338	NaN	Shreeram Modi	Anuj Jain	Anuj Jain
## 342	NaN	Jessica Li	Anuj Jain	Anuj Jain
## 348	NaN	Sean Wang	Jessica Li	Jessica Li
## 356	NaN	Reeya Kansra	Shreeram Modi	Reeya Kansra
## 366	NaN	Shreeram Modi	Salsabila Mahdi	Salsabila Mahdi
## 378	NaN	Salsabila Mahdi	Ethan Rosen	Ethan Rosen
## 387	NaN	Sam Jin	Shlomo Kofman	Shlomo Kofman
## 401	NaN	Anuj Jain	Noor Mirza-Rashid	Anuj Jain
## 411	NaN	Sean Wang	Anuj Jain	Anuj Jain
## 414	NaN	Sean Wang	Adelle Fernando	Adelle Fernando
## 421	NaN	Shreeram Modi	Julian Michael	Julian Michael
## 424	NaN	Shlomo Kofman	Sam Jin	Sam Jin
## 425	NaN	Jessica Li	Shreeram Modi	Jessica Li
## 429	NaN	Sam Jin	Adelle Fernando	Sam Jin
## 431	NaN	Shreeram Modi	Adelle Fernando	Shreeram Modi
## 433	NaN	Aliyaah Toussaint	Adelle Fernando	Aliyaah Toussaint
## 436	NaN	Vishakh Padmakumar	Shreeram Modi	Vishakh Padmakumar
## 439	NaN	Sam Jin	Adelle Fernando	Adelle Fernando
## 448	NaN	Adelle Fernando	Jessica Li	Adelle Fernando
## 510	NaN	Anuj Jain	Shlomo Kofman	Anuj Jain
## 533	NaN	Shreeram Modi	Salsabila Mahdi	Salsabila Mahdi
## 538	NaN	Salsabila Mahdi	Vishakh Padmakumar	Vishakh Padmakumar
## 544	NaN	Adelle Fernando	Jackson Petty	Jackson Petty
## 550	NaN	Jessica Li	Shlomo Kofman	Jessica Li
## 561	NaN	Shreeram Modi	Aliyaah Toussaint	Aliyaah Toussaint
## 598	NaN	Shreeram Modi	Adelle Fernando	Adelle Fernando
## 602	NaN	Jessica Li	Sean Wang	Jessica Li
## 606	NaN	Adelle Fernando	Reeya Kansra	Reeya Kansra
## 626	NaN	Sam Jin	Adelle Fernando	Adelle Fernando
## 637	NaN	Sean Wang	Adelle Fernando	Sean Wang
## 641	NaN	David Rein	Jessica Li	David Rein
## 647	NaN	Salsabila Mahdi	Vishakh Padmakumar	Vishakh Padmakumar
## 648	NaN	Sean Wang	Ethan Rosen	Ethan Rosen
## 658	NaN	Sean Wang	Julian Michael	Julian Michael
## 677	NaN	Julian Michael	Jessica Li	Julian Michael
## 679	NaN	Jessica Li	Salsabila Mahdi	Salsabila Mahdi
## 680	NaN	Julian Michael	Salsabila Mahdi	Julian Michael
## 683	NaN	Anuj Jain	Shreeram Modi	Anuj Jain
##	Dishonest debater	Is single debater	Has honest debater	Final_Setting
## 21	Sean Wang	FALSE	TRUE	Human Debate
## 43	Jessica Li	FALSE	TRUE	Human Debate
## 78	Reeya Kansra	FALSE	TRUE	Human Debate
## 81	Sean Wang	FALSE	TRUE	Human Debate
## 91	Shlomo Kofman	FALSE	TRUE	Human Debate
## 94	Sean Wang	FALSE	TRUE	Human Debate
## 99	Adelle Fernando	FALSE	TRUE	Human Debate
## 113	Sean Wang	FALSE	TRUE	Human Debate
## 136	Adelle Fernando	FALSE	TRUE	Human Debate
## 140	Jessica Li	FALSE	TRUE	Human Debate
## 149	Salsabila Mahdi	FALSE	TRUE	Human Debate
## 177	Reeya Kansra	FALSE	TRUE	Human Debate

## 179	Reeya Kansra	FALSE	TRUE	Human Debate
## 185	Shreeram Modi	FALSE	TRUE	Human Debate
## 186	Adelle Fernando	FALSE	TRUE	Human Debate
## 191	Sean Wang	FALSE	TRUE	Human Debate
## 202	Adelle Fernando	FALSE	TRUE	Human Debate
## 211	Shreeram Modi	FALSE	TRUE	Human Debate
## 215	Sean Wang	FALSE	TRUE	Human Debate
## 216	Noor Mirza-Rashid	FALSE	TRUE	Human Debate
## 219	Sam Jin	FALSE	TRUE	Human Debate
## 236	Sam Jin	FALSE	TRUE	Human Debate
## 240	Reeya Kansra	FALSE	TRUE	Human Debate
## 241	Sam Jin	FALSE	TRUE	Human Debate
## 254	Reeya Kansra	FALSE	TRUE	Human Debate
## 270	Reeya Kansra	FALSE	TRUE	Human Debate
## 276	Adelle Fernando	FALSE	TRUE	Human Debate
## 290	Adelle Fernando	FALSE	TRUE	Human Debate
## 306	Sean Wang	FALSE	TRUE	Human Debate
## 324	Reeya Kansra	FALSE	TRUE	Human Debate
## 331	Shreeram Modi	FALSE	TRUE	Human Debate
## 332	Adelle Fernando	FALSE	TRUE	Human Debate
## 338	Shreeram Modi	FALSE	TRUE	Human Debate
## 342	Jessica Li	FALSE	TRUE	Human Debate
## 348	Sean Wang	FALSE	TRUE	Human Debate
## 356	Shreeram Modi	FALSE	TRUE	Human Debate
## 366	Shreeram Modi	FALSE	TRUE	Human Debate
## 378	Salsabila Mahdi	FALSE	TRUE	Human Debate
## 387	Sam Jin	FALSE	TRUE	Human Debate
## 401	Noor Mirza-Rashid	FALSE	TRUE	Human Debate
## 411	Sean Wang	FALSE	TRUE	Human Debate
## 414	Sean Wang	FALSE	TRUE	Human Debate
## 421	Shreeram Modi	FALSE	TRUE	Human Debate
## 424	Shlomo Kofman	FALSE	TRUE	Human Debate
## 425	Shreeram Modi	FALSE	TRUE	Human Debate
## 429	Adelle Fernando	FALSE	TRUE	Human Debate
## 431	Adelle Fernando	FALSE	TRUE	Human Debate
## 433	Adelle Fernando	FALSE	TRUE	Human Debate
## 436	Shreeram Modi	FALSE	TRUE	Human Debate
## 439	Sam Jin	FALSE	TRUE	Human Debate
## 448	Jessica Li	FALSE	TRUE	Human Debate
## 510	Shlomo Kofman	FALSE	TRUE	Human Debate
## 533	Shreeram Modi	FALSE	TRUE	Human Debate
## 538	Salsabila Mahdi	FALSE	TRUE	Human Debate
## 544	Adelle Fernando	FALSE	TRUE	Human Debate
## 550	Shlomo Kofman	FALSE	TRUE	Human Debate
## 561	Shreeram Modi	FALSE	TRUE	Human Debate
## 598	Shreeram Modi	FALSE	TRUE	Human Debate
## 602	Sean Wang	FALSE	TRUE	Human Debate
## 606	Adelle Fernando	FALSE	TRUE	Human Debate
## 626	Sam Jin	FALSE	TRUE	Human Debate
## 637	Adelle Fernando	FALSE	TRUE	Human Debate
## 641	Jessica Li	FALSE	TRUE	Human Debate
## 647	Salsabila Mahdi	FALSE	TRUE	Human Debate
## 648	Sean Wang	FALSE	TRUE	Human Debate
## 658	Sean Wang	FALSE	TRUE	Human Debate



## 677	Jessica Li	FALSE	TRUE	Human Debate
## 679	Jessica Li	FALSE	TRUE	Human Debate
## 680	Salsabila Mahdi	FALSE	TRUE	Human Debate
## 683	Shreeram Modi	FALSE	TRUE	Human Debate
##	Setting			
## 21	Human Debate			
## 43	Human Debate			
## 78	Human Debate			
## 81	Human Debate			
## 91	Human Debate			
## 94	Human Debate			
## 99	Human Debate			
## 113	Human Debate			
## 136	Human Debate			
## 140	Human Debate			
## 149	Human Debate			
## 177	Human Debate			
## 179	Human Debate			
## 185	Human Debate			
## 186	Human Debate			
## 191	Human Debate			
## 202	Human Debate			
## 211	Human Debate			
## 215	Human Debate			
## 216	Human Debate			
## 219	Human Debate			
## 236	Human Debate			
## 240	Human Debate			
## 241	Human Debate			
## 254	Human Debate			
## 270	Human Debate			
## 276	Human Debate			
## 290	Human Debate			
## 306	Human Debate			
## 324	Human Debate			
## 331	Human Debate			
## 332	Human Debate			
## 338	Human Debate			
## 342	Human Debate			
## 348	Human Debate			
## 356	Human Debate			
## 366	Human Debate			
## 378	Human Debate			
## 387	Human Debate			
## 401	Human Debate			
## 411	Human Debate			
## 414	Human Debate			
## 421	Human Debate			
## 424	Human Debate			
## 425	Human Debate			
## 429	Human Debate			
## 431	Human Debate			
## 433	Human Debate			
## 436	Human Debate			

## 439 Human Debate  
 ## 448 Human Debate  
 ## 510 Human Debate  
 ## 533 Human Debate  
 ## 538 Human Debate  
 ## 544 Human Debate  
 ## 550 Human Debate  
 ## 561 Human Debate  
 ## 598 Human Debate  
 ## 602 Human Debate  
 ## 606 Human Debate  
 ## 626 Human Debate  
 ## 637 Human Debate  
 ## 641 Human Debate  
 ## 647 Human Debate  
 ## 648 Human Debate  
 ## 658 Human Debate  
 ## 677 Human Debate  
 ## 679 Human Debate  
 ## 680 Human Debate  
 ## 683 Human Debate

##

## 21

## 43

## 78

## 81

## 91

## 94

## 99

## 113

## 136

## 140

## 149

## 177

## 179

## 185

## 186

## 191

## 202

## 211

## 215

## 216

## 219

## 236

## 240

## 241

## 254

## 270

## 276

## 290

## 306

## 324

## 331

## 332

Which is

Which

How did Earth

Why does Koroby

Did the questions Tremain

Why did the physicist an

What was the bl

What is l

Why was

Why was the main character daydream

Generally, which of the following b

Which of these sets of d

What was the

Why do Bob and Quezy ha

Why was Dr. Crander so p

What is not a type techn

What best describes how the overall tone changed f

What would best describe Asa's m

Why did the Earth c

Who are the four to blame

What did Zen think of the plan the r

Why is Grannie Annie so concerned abo

How many compan

Who are the four to blame

What was the population o

Which is

Why does Koroby

How did the planet of Niobe compare to

## 338 Which best describes the relat  
 ## 342 What is the relationship between  
 ## 348 What is the  
 ## 356 Why is  
 ## 366 What does the narrat  
 ## 378  
 ## 387 What happens to a changelin  
 ## 401 What makes the protagonists become less concerned ab  
 ## 411 Why was the approach that Charlie took to eng  
 ## 414 Why do Bob and Quezy ha  
 ## 421 Why is Jor  
 ## 424 What would best describe Asa's  
 ## 425  
 ## 429 What is Androka's motivation  
 ## 431 Which of the following is not a reason why Koroby is impressed by the s  
 ## 433 Johnathan doesn't tell the Interstellar Cosmography Society about the twenty-seven women who are  
 ## 436 How many people  
 ## 439 What was the bl  
 ## 448  
 ## 510 What was the relationship like between  
 ## 533 Why did Korvin have to word his  
 ## 538 How would you describe  
 ## 544  
 ## 550 Why did Pashk  
 ## 561 What does the narrat  
 ## 598 Why wa  
 ## 602 What would be the main reason Mr. Ranson wants to find th  
 ## 606 Why did the Earth  
 ## 626 How did H  
 ## 637 What is l  
 ## 641 What is likely to happen to the crew  
 ## 647 Why  
 ## 648 What is the true explanation for Charles  
 ## 658 If Dan and Erica had been seen together before the accident, what  
 ## 677 Of the following situations, what was th  
 ## 679 Why was the main character daydream  
 ## 680 What is the style of  
 ## 683 What is not a type techn  
 ## Article ID Speed annotator accuracy bins Untimed annotator context bins  
 ## 21 61499 0 4  
 ## 43 61053 0 4  
 ## 78 60412 0 2  
 ## 81 62314 0.2 3  
 ## 91 52844 0.2 4  
 ## 94 51126 0.2 2  
 ## 99 52855 0.2 3  
 ## 113 63527 0 3  
 ## 136 63633 0.2 4  
 ## 140 43046 0.4 2  
 ## 149 51688 0.2 2  
 ## 177 61499 0.2 3  
 ## 179 55933 0.4 3  
 ## 185 63862 0.2 2  
 ## 186 62314 0.2 3

## 191	50893	0.2	3
## 202	63527	0.2	2
## 211	62314	0.2	3
## 215	51295	0.4	3
## 216	62569	0.4	3
## 219	53269	0.2	4
## 236	61467	0.4	2
## 240	60412	0.2	3
## 241	61481	0.2	3
## 254	51126	0	2
## 270	63109	0.2	2
## 276	50818	0.2	3
## 290	61481	0.2	3
## 306	51126	0.2	2
## 324	61499	0	4
## 331	62314	0.2	3
## 332	51395	0.2	3
## 338	62569	0.2	3
## 342	51351	0.2	3
## 348	61430	0	2
## 356	63109	0.2	3
## 366	51201	0	3
## 378	50818	0.4	4
## 387	61467	0.4	2
## 401	62569	0	2
## 411	51320	0.2	2
## 414	63527	0.2	2
## 421	61430	0.4	2
## 424	61467	0.4	2
## 425	43046	0.4	3
## 429	61481	0	3
## 431	62314	0.2	2
## 433	63401	0.2	2
## 436	51483	0.2	2
## 439	52855	0.2	3
## 448	63523	0.2	3
## 510	51150	0.2	3
## 533	30029	0.4	2
## 538	55933	0	4
## 544	51395	0.2	2
## 550	51256	0.4	3
## 561	51201	0	3
## 598	63633	0.2	4
## 602	62085	0.2	2
## 606	60412	0.2	3
## 626	63855	0	2
## 637	63527	0	3
## 641	61007	0.2	2
## 647	55933	0.2	2
## 648	63631	0.2	3
## 658	51295	0.4	4
## 677	63862	0.4	3
## 679	51688	0.2	2
## 680	61285	0.4	2

## 683	62569	0.4	3
##	Speed annotator accuracy	Untimed annotator context	Is offline
## 21	0.000000	3.666667	FALSE
## 43	0.000000	3.666667	FALSE
## 78	0.000000	2.000000	FALSE
## 81	0.200000	3.000000	FALSE
## 91	0.200000	4.000000	FALSE
## 94	0.200000	1.800000	FALSE
## 99	0.200000	2.600000	FALSE
## 113	0.000000	3.000000	FALSE
## 136	0.200000	4.000000	FALSE
## 140	0.400000	1.600000	FALSE
## 149	0.200000	2.333333	FALSE
## 177	0.200000	3.333333	FALSE
## 179	0.400000	3.333333	FALSE
## 185	0.200000	2.000000	FALSE
## 186	0.200000	2.600000	FALSE
## 191	0.200000	3.333333	FALSE
## 202	0.200000	1.666667	FALSE
## 211	0.200000	2.600000	FALSE
## 215	0.400000	3.000000	FALSE
## 216	0.400000	3.000000	FALSE
## 219	0.200000	3.666667	FALSE
## 236	0.400000	2.333333	FALSE
## 240	0.200000	2.600000	FALSE
## 241	0.200000	3.333333	FALSE
## 254	0.000000	2.200000	FALSE
## 270	0.200000	1.666667	FALSE
## 276	0.200000	3.400000	FALSE
## 290	0.200000	3.333333	FALSE
## 306	0.200000	2.200000	FALSE
## 324	0.000000	3.666667	FALSE
## 331	0.200000	3.000000	FALSE
## 332	0.166667	2.750000	FALSE
## 338	0.200000	3.000000	FALSE
## 342	0.166667	2.800000	FALSE
## 348	0.000000	1.600000	FALSE
## 356	0.200000	2.666667	FALSE
## 366	0.000000	2.600000	FALSE
## 378	0.400000	3.600000	FALSE
## 387	0.400000	2.000000	FALSE
## 401	0.000000	2.000000	FALSE
## 411	0.166667	2.400000	FALSE
## 414	0.200000	1.666667	FALSE
## 421	0.400000	2.200000	FALSE
## 424	0.400000	2.333333	FALSE
## 425	0.400000	3.200000	FALSE
## 429	0.000000	3.333333	FALSE
## 431	0.200000	2.200000	FALSE
## 433	0.200000	2.200000	FALSE
## 436	0.200000	2.200000	FALSE
## 439	0.200000	2.600000	FALSE
## 448	0.200000	3.400000	FALSE
## 510	0.200000	3.000000	FALSE

## 533	0.4000000	1.800000	FALSE
## 538	0.0000000	4.000000	FALSE
## 544	0.2000000	2.250000	FALSE
## 550	0.4000000	3.000000	FALSE
## 561	0.0000000	2.600000	FALSE
## 598	0.2000000	4.000000	FALSE
## 602	0.2000000	2.333333	FALSE
## 606	0.2000000	2.600000	FALSE
## 626	0.0000000	2.000000	FALSE
## 637	0.0000000	3.000000	FALSE
## 641	0.2000000	1.666667	FALSE
## 647	0.2000000	2.000000	FALSE
## 648	0.2000000	2.666667	FALSE
## 658	0.4000000	3.666667	FALSE
## 677	0.4000000	3.400000	FALSE
## 679	0.2000000	2.333333	FALSE
## 680	0.4000000	2.000000	FALSE
## 683	0.4000000	3.000000	FALSE
##	End time	Last modified time	Final_Accuracy
## 21	2023-04-10 16:16:41	2023-04-28 11:30:24	TRUE
## 43	2023-05-21 14:03:16	2023-05-26 10:54:34	TRUE
## 78	2023-05-19 15:40:18	2023-05-19 16:20:39	TRUE
## 81	2023-06-22 17:38:01	2023-06-23 11:56:33	TRUE
## 91	2023-07-27 16:36:48	2023-07-27 16:36:48	TRUE
## 94	2023-06-29 18:36:11	2023-06-29 18:41:52	TRUE
## 99	2023-07-13 17:57:20	2023-07-31 15:39:55	TRUE
## 113	2023-04-21 16:43:34	2023-04-21 16:48:05	TRUE
## 136	2023-07-24 15:45:08	2023-07-24 15:45:08	TRUE
## 140	2023-04-17 16:40:55	2023-06-12 16:25:09	TRUE
## 149	2023-04-10 17:33:21	2023-04-12 17:18:09	TRUE
## 177	2023-04-18 15:05:57	2023-04-28 10:25:57	TRUE
## 179	2023-07-20 15:41:51	2023-07-20 15:41:51	TRUE
## 185	2023-02-27 17:02:34	2023-04-28 16:44:08	TRUE
## 186	2023-05-12 16:09:16	2023-05-12 16:09:16	TRUE
## 191	2023-05-09 16:15:12	2023-05-19 16:52:53	TRUE
## 202	2023-04-14 18:04:29	2023-04-29 18:16:46	TRUE
## 211	2023-05-12 16:15:12	2023-05-18 11:38:29	TRUE
## 215	2023-02-13 16:41:56	2023-02-13 16:41:56	TRUE
## 216	2023-04-14 16:31:19	2023-05-01 16:31:54	TRUE
## 219	2023-07-28 15:39:59	2023-07-28 15:39:59	TRUE
## 236	2023-06-26 17:15:36	2023-06-26 17:15:36	TRUE
## 240	2023-06-16 16:50:59	2023-06-23 23:14:19	TRUE
## 241	2023-07-17 16:33:07	2023-07-17 16:33:07	TRUE
## 254	2023-07-17 15:04:00	2023-07-17 15:04:00	TRUE
## 270	2023-04-14 17:10:57	2023-04-28 16:50:44	TRUE
## 276	2023-05-15 16:10:35	2023-05-15 16:10:35	TRUE
## 290	2023-07-06 15:47:04	2023-07-06 15:47:04	TRUE
## 306	2023-06-29 17:10:29	2023-07-17 18:30:49	TRUE
## 324	2023-05-01 17:55:02	2023-05-11 16:49:22	TRUE
## 331	2023-05-05 11:55:03	2023-05-11 15:50:12	TRUE
## 332	2023-04-15 06:30:53	2023-04-29 17:56:08	TRUE
## 338	2023-06-22 18:58:39	2023-06-22 18:58:39	TRUE
## 342	2023-06-26 15:43:46	2023-06-26 15:57:14	TRUE
## 348	2023-02-24 11:44:11	2023-04-28 16:45:16	TRUE

##	356	2023-04-21	16:49:20	2023-04-21	16:49:20	TRUE
##	366	2023-05-12	10:15:53	2023-05-12	10:15:53	TRUE
##	378	2023-05-12	11:42:59	2023-06-12	16:33:57	TRUE
##	387	2023-07-07	17:37:10	2023-07-07	17:37:10	TRUE
##	401	2023-04-21	16:27:51	2023-04-21	16:27:51	TRUE
##	411	2023-04-28	13:51:32	2023-05-12	10:49:32	TRUE
##	414	2023-04-14	16:42:51	2023-06-12	16:48:26	TRUE
##	421	2023-02-17	11:51:02	2023-05-15	17:10:36	TRUE
##	424	2023-06-26	18:59:34	2023-06-26	18:59:34	TRUE
##	425	2023-04-14	17:20:04	2023-04-28	10:10:59	TRUE
##	429	2023-07-06	17:58:47	2023-07-06	17:58:47	TRUE
##	431	2023-05-12	11:47:45	2023-06-12	16:01:09	TRUE
##	433	2023-04-07	16:34:58	2023-04-07	16:34:58	TRUE
##	436	2023-05-11	14:57:46	2023-05-11	14:57:46	TRUE
##	439	2023-07-13	13:02:18	2023-07-13	13:02:18	TRUE
##	448	2023-07-14	16:51:09	2023-07-14	16:51:09	TRUE
##	510	2023-08-04	16:36:03	2023-08-04	16:36:03	TRUE
##	533	2023-03-10	11:53:42	2023-04-13	16:46:04	TRUE
##	538	2023-04-28	10:13:44	2023-06-12	16:24:31	TRUE
##	544	2023-04-17	17:06:13	2023-04-18	13:42:45	TRUE
##	550	2023-08-03	16:36:15	2023-08-03	16:36:15	TRUE
##	561	2023-04-17	17:45:31	2023-04-29	22:45:31	TRUE
##	598	2023-07-24	17:40:02	2023-07-24	17:40:02	TRUE
##	602	2023-07-17	19:39:59	2023-07-17	19:39:59	TRUE
##	606	2023-07-07	18:12:21	2023-07-07	21:30:24	TRUE
##	626	2023-07-17	19:00:09	2023-07-17	19:00:09	TRUE
##	637	2023-04-17	18:48:16	2023-04-18	14:26:39	TRUE
##	641	2023-05-12	10:16:04	2023-05-12	10:16:04	TRUE
##	647	2023-04-24	17:33:24	2023-05-24	16:28:55	TRUE
##	648	2023-03-20	17:06:51	2023-04-28	16:39:55	TRUE
##	658	2023-02-22	17:30:45	2023-02-22	17:30:45	TRUE
##	677	2023-03-07	21:04:25	2023-04-28	17:01:26	TRUE
##	679	2023-06-22	21:37:32	2023-06-22	21:37:32	TRUE
##	680	2023-03-07	17:00:26	2023-04-28	17:38:19	TRUE
##	683	2023-04-21	11:01:01	2023-06-12	16:05:11	TRUE
##		Human Consultancy	Sample AI Consultancy	Sample	Human Debate	Sample
##	21		FALSE	FALSE		FALSE
##	43		FALSE	FALSE		FALSE
##	78		FALSE	FALSE		FALSE
##	81		FALSE	FALSE		FALSE
##	91		FALSE	FALSE		TRUE
##	94		FALSE	FALSE		FALSE
##	99		FALSE	FALSE		FALSE
##	113		FALSE	FALSE		FALSE
##	136		FALSE	FALSE		FALSE
##	140		FALSE	FALSE		TRUE
##	149		FALSE	FALSE		FALSE
##	177		FALSE	FALSE		FALSE
##	179		FALSE	FALSE		FALSE
##	185		FALSE	FALSE		TRUE
##	186		FALSE	FALSE		FALSE
##	191		FALSE	FALSE		FALSE
##	202		FALSE	FALSE		FALSE
##	211		FALSE	FALSE		TRUE

## 215	FALSE	FALSE	TRUE
## 216	FALSE	FALSE	FALSE
## 219	FALSE	FALSE	TRUE
## 236	FALSE	FALSE	FALSE
## 240	FALSE	FALSE	FALSE
## 241	FALSE	FALSE	FALSE
## 254	FALSE	FALSE	FALSE
## 270	FALSE	FALSE	TRUE
## 276	FALSE	FALSE	TRUE
## 290	FALSE	FALSE	TRUE
## 306	FALSE	FALSE	FALSE
## 324	FALSE	FALSE	TRUE
## 331	FALSE	FALSE	TRUE
## 332	FALSE	FALSE	TRUE
## 338	FALSE	FALSE	TRUE
## 342	FALSE	FALSE	FALSE
## 348	FALSE	FALSE	TRUE
## 356	FALSE	FALSE	TRUE
## 366	FALSE	FALSE	FALSE
## 378	FALSE	FALSE	TRUE
## 387	FALSE	FALSE	FALSE
## 401	FALSE	FALSE	FALSE
## 411	FALSE	FALSE	FALSE
## 414	FALSE	FALSE	TRUE
## 421	FALSE	FALSE	TRUE
## 424	FALSE	FALSE	TRUE
## 425	FALSE	FALSE	TRUE
## 429	FALSE	FALSE	FALSE
## 431	FALSE	FALSE	TRUE
## 433	FALSE	FALSE	TRUE
## 436	FALSE	FALSE	TRUE
## 439	FALSE	FALSE	TRUE
## 448	FALSE	FALSE	TRUE
## 510	FALSE	FALSE	TRUE
## 533	FALSE	FALSE	TRUE
## 538	FALSE	FALSE	TRUE
## 544	FALSE	FALSE	TRUE
## 550	FALSE	FALSE	TRUE
## 561	FALSE	FALSE	TRUE
## 598	FALSE	FALSE	TRUE
## 602	FALSE	FALSE	TRUE
## 606	FALSE	FALSE	TRUE
## 626	FALSE	FALSE	TRUE
## 637	FALSE	FALSE	TRUE
## 641	FALSE	FALSE	TRUE
## 647	FALSE	FALSE	TRUE
## 648	FALSE	FALSE	TRUE
## 658	FALSE	FALSE	TRUE
## 677	FALSE	FALSE	TRUE
## 679	FALSE	FALSE	TRUE
## 680	FALSE	FALSE	TRUE
## 683	FALSE	FALSE	TRUE
##	AI Debate Sample	Sample Consultancy Sample	initial_question_weights
## 21	FALSE	FALSE	FALSE 0.5000000



## 43	FALSE	FALSE	FALSE	0.5000000
## 78	FALSE	FALSE	FALSE	0.5000000
## 81	FALSE	FALSE	FALSE	0.2500000
## 91	FALSE	TRUE	TRUE	0.1666667
## 94	FALSE	FALSE	FALSE	0.5000000
## 99	FALSE	FALSE	FALSE	0.2500000
## 113	FALSE	FALSE	FALSE	0.3333333
## 136	FALSE	FALSE	FALSE	0.1428571
## 140	FALSE	TRUE	TRUE	1.0000000
## 149	FALSE	FALSE	FALSE	0.2500000
## 177	FALSE	FALSE	FALSE	0.5000000
## 179	FALSE	FALSE	FALSE	0.5000000
## 185	FALSE	TRUE	TRUE	1.0000000
## 186	FALSE	FALSE	FALSE	0.5000000
## 191	FALSE	FALSE	FALSE	0.2000000
## 202	FALSE	FALSE	FALSE	0.5000000
## 211	FALSE	TRUE	TRUE	0.5000000
## 215	FALSE	TRUE	TRUE	1.0000000
## 216	FALSE	FALSE	FALSE	0.5000000
## 219	FALSE	TRUE	TRUE	0.1666667
## 236	FALSE	FALSE	FALSE	0.5000000
## 240	FALSE	FALSE	FALSE	0.5000000
## 241	FALSE	FALSE	FALSE	0.2500000
## 254	FALSE	FALSE	FALSE	0.5000000
## 270	FALSE	TRUE	TRUE	1.0000000
## 276	FALSE	TRUE	TRUE	0.5000000
## 290	FALSE	TRUE	TRUE	0.2500000
## 306	FALSE	FALSE	FALSE	0.5000000
## 324	FALSE	TRUE	TRUE	0.5000000
## 331	FALSE	TRUE	TRUE	0.2500000
## 332	FALSE	TRUE	TRUE	0.5000000
## 338	FALSE	TRUE	TRUE	0.5000000
## 342	FALSE	FALSE	FALSE	0.5000000
## 348	FALSE	TRUE	TRUE	1.0000000
## 356	FALSE	TRUE	TRUE	0.3333333
## 366	FALSE	FALSE	FALSE	0.2500000
## 378	FALSE	TRUE	TRUE	0.3333333
## 387	FALSE	FALSE	FALSE	0.5000000
## 401	FALSE	FALSE	FALSE	0.2500000
## 411	FALSE	FALSE	FALSE	0.5000000
## 414	FALSE	TRUE	TRUE	0.5000000
## 421	FALSE	TRUE	TRUE	1.0000000
## 424	FALSE	TRUE	TRUE	0.5000000
## 425	FALSE	TRUE	TRUE	0.3333333
## 429	FALSE	FALSE	FALSE	0.2000000
## 431	FALSE	TRUE	TRUE	1.0000000
## 433	FALSE	TRUE	TRUE	0.3333333
## 436	FALSE	TRUE	TRUE	1.0000000
## 439	FALSE	TRUE	TRUE	0.2500000
## 448	FALSE	TRUE	TRUE	0.2000000
## 510	FALSE	TRUE	TRUE	0.1666667
## 533	FALSE	TRUE	TRUE	0.5000000
## 538	FALSE	TRUE	TRUE	0.5000000
## 544	FALSE	TRUE	TRUE	1.0000000

## 550	FALSE	TRUE	TRUE	0.2500000
## 561	FALSE	TRUE	TRUE	0.2500000
## 598	FALSE	TRUE	TRUE	0.1428571
## 602	FALSE	TRUE	TRUE	0.5000000
## 606	FALSE	TRUE	TRUE	0.5000000
## 626	FALSE	TRUE	TRUE	0.2500000
## 637	FALSE	TRUE	TRUE	0.3333333
## 641	FALSE	TRUE	TRUE	0.3333333
## 647	FALSE	TRUE	TRUE	0.5000000
## 648	FALSE	TRUE	TRUE	0.2000000
## 658	FALSE	TRUE	TRUE	0.3333333
## 677	FALSE	TRUE	TRUE	0.3333333
## 679	FALSE	TRUE	TRUE	0.2500000
## 680	FALSE	TRUE	TRUE	1.0000000
## 683	FALSE	TRUE	TRUE	0.5000000
##	initial_question_weights_grouped_setting			
## 21			0.5	
## 43			0.5	
## 78			0.5	
## 81			0.5	
## 91			1.0	
## 94			0.5	
## 99			0.5	
## 113			0.5	
## 136			0.5	
## 140			1.0	
## 149			0.5	
## 177			0.5	
## 179			0.5	
## 185			1.0	
## 186			0.5	
## 191			0.5	
## 202			0.5	
## 211			0.5	
## 215			1.0	
## 216			0.5	
## 219			1.0	
## 236			0.5	
## 240			0.5	
## 241			0.5	
## 254			0.5	
## 270			1.0	
## 276			0.5	
## 290			0.5	
## 306			0.5	
## 324			0.5	
## 331			0.5	
## 332			0.5	
## 338			0.5	
## 342			0.5	
## 348			1.0	
## 356			1.0	
## 366			0.5	
## 378			0.5	

## 387	0.5
## 401	0.5
## 411	0.5
## 414	0.5
## 421	1.0
## 424	0.5
## 425	0.5
## 429	0.5
## 431	1.0
## 433	1.0
## 436	1.0
## 439	0.5
## 448	1.0
## 510	1.0
## 533	1.0
## 538	0.5
## 544	1.0
## 550	1.0
## 561	0.5
## 598	0.5
## 602	1.0
## 606	0.5
## 626	0.5
## 637	0.5
## 641	0.5
## 647	0.5
## 648	0.5
## 658	1.0
## 677	1.0
## 679	0.5
## 680	1.0
## 683	0.5
## sampled_consultancies_all_debates_weights	
## 21	0.5000000
## 43	0.5000000
## 78	0.5000000
## 81	0.3333333
## 91	0.2000000
## 94	0.5000000
## 99	0.2500000
## 113	0.3333333
## 136	0.1666667
## 140	1.0000000
## 149	0.2500000
## 177	0.5000000
## 179	0.5000000
## 185	1.0000000
## 186	0.5000000
## 191	0.2500000
## 202	0.5000000
## 211	0.5000000
## 215	1.0000000
## 216	0.5000000
## 219	0.2000000

##	236	0.5000000
##	240	0.5000000
##	241	0.2500000
##	254	0.5000000
##	270	1.0000000
##	276	0.5000000
##	290	0.2500000
##	306	0.5000000
##	324	0.5000000
##	331	0.3333333
##	332	0.5000000
##	338	0.5000000
##	342	0.5000000
##	348	1.0000000
##	356	0.5000000
##	366	0.3333333
##	378	0.3333333
##	387	0.5000000
##	401	0.2500000
##	411	0.5000000
##	414	0.5000000
##	421	1.0000000
##	424	0.5000000
##	425	0.3333333
##	429	0.3333333
##	431	1.0000000
##	433	0.5000000
##	436	1.0000000
##	439	0.2500000
##	448	0.3333333
##	510	0.2000000
##	533	0.5000000
##	538	0.5000000
##	544	1.0000000
##	550	0.2500000
##	561	0.3333333
##	598	0.1666667
##	602	0.5000000
##	606	0.5000000
##	626	0.2500000
##	637	0.3333333
##	641	0.3333333
##	647	0.5000000
##	648	0.2500000
##	658	0.5000000
##	677	0.5000000
##	679	0.2500000
##	680	1.0000000
##	683	0.5000000
##	sampled_consultancies_all_debates_weights_grouped_setting	
##	21	0.5
##	43	0.5
##	78	0.5
##	81	0.5

## 91	1.0
## 94	0.5
## 99	0.5
## 113	0.5
## 136	0.5
## 140	1.0
## 149	0.5
## 177	0.5
## 179	0.5
## 185	1.0
## 186	0.5
## 191	0.5
## 202	0.5
## 211	0.5
## 215	1.0
## 216	0.5
## 219	1.0
## 236	0.5
## 240	0.5
## 241	0.5
## 254	0.5
## 270	1.0
## 276	0.5
## 290	0.5
## 306	0.5
## 324	0.5
## 331	0.5
## 332	0.5
## 338	0.5
## 342	0.5
## 348	1.0
## 356	1.0
## 366	0.5
## 378	0.5
## 387	0.5
## 401	0.5
## 411	0.5
## 414	0.5
## 421	1.0
## 424	0.5
## 425	0.5
## 429	0.5
## 431	1.0
## 433	1.0
## 436	1.0
## 439	0.5
## 448	1.0
## 510	1.0
## 533	1.0
## 538	0.5
## 544	1.0
## 550	1.0
## 561	0.5
## 598	0.5

## 602	1.0
## 606	0.5
## 626	0.5
## 637	0.5
## 641	0.5
## 647	0.5
## 648	0.5
## 658	1.0
## 677	1.0
## 679	0.5
## 680	1.0
## 683	0.5
## sampled_consultancies_all_debates_weights_setting	
## 21	0.5
## 43	0.5
## 78	0.5
## 81	0.5
## 91	1.0
## 94	0.5
## 99	0.5
## 113	0.5
## 136	0.5
## 140	1.0
## 149	0.5
## 177	0.5
## 179	0.5
## 185	1.0
## 186	0.5
## 191	0.5
## 202	0.5
## 211	0.5
## 215	1.0
## 216	0.5
## 219	1.0
## 236	0.5
## 240	0.5
## 241	0.5
## 254	0.5
## 270	1.0
## 276	0.5
## 290	0.5
## 306	0.5
## 324	0.5
## 331	0.5
## 332	0.5
## 338	0.5
## 342	0.5
## 348	1.0
## 356	1.0
## 366	0.5
## 378	0.5
## 387	0.5
## 401	0.5
## 411	0.5

## 414	0.5
## 421	1.0
## 424	0.5
## 425	0.5
## 429	0.5
## 431	1.0
## 433	1.0
## 436	1.0
## 439	0.5
## 448	1.0
## 510	1.0
## 533	1.0
## 538	0.5
## 544	1.0
## 550	1.0
## 561	0.5
## 598	0.5
## 602	1.0
## 606	0.5
## 626	0.5
## 637	0.5
## 641	0.5
## 647	0.5
## 648	0.5
## 658	1.0
## 677	1.0
## 679	0.5
## 680	1.0
## 683	0.5
## sampled_consultancies_debates_weights_grouped_setting	
## 21	0
## 43	0
## 78	0
## 81	0
## 91	1
## 94	0
## 99	0
## 113	0
## 136	0
## 140	1
## 149	0
## 177	0
## 179	0
## 185	1
## 186	0
## 191	0
## 202	0
## 211	1
## 215	1
## 216	0
## 219	1
## 236	0
## 240	0
## 241	0

## 254		0
## 270		1
## 276		1
## 290		1
## 306		0
## 324		1
## 331		1
## 332		1
## 338		1
## 342		0
## 348		1
## 356		1
## 366		0
## 378		1
## 387		0
## 401		0
## 411		0
## 414		1
## 421		1
## 424		1
## 425		1
## 429		0
## 431		1
## 433		1
## 436		1
## 439		1
## 448		1
## 510		1
## 533		1
## 538		1
## 544		1
## 550		1
## 561		1
## 598		1
## 602		1
## 606		1
## 626		1
## 637		1
## 641		1
## 647		1
## 648		1
## 658		1
## 677		1
## 679		1
## 680		1
## 683		1
##	sampled_consultancies_debates_weights	Final_Accuracy_char
## 21	0.0000000	NA
## 43	0.0000000	NA
## 78	0.0000000	NA
## 81	0.0000000	NA
## 91	0.2500000	NA
## 94	0.0000000	NA
## 99	0.0000000	NA



## 113	0.0000000	NA
## 136	0.0000000	NA
## 140	1.0000000	NA
## 149	0.0000000	NA
## 177	0.0000000	NA
## 179	0.0000000	NA
## 185	1.0000000	NA
## 186	0.0000000	NA
## 191	0.0000000	NA
## 202	0.0000000	NA
## 211	1.0000000	NA
## 215	1.0000000	NA
## 216	0.0000000	NA
## 219	0.2500000	NA
## 236	0.0000000	NA
## 240	0.0000000	NA
## 241	0.0000000	NA
## 254	0.0000000	NA
## 270	1.0000000	NA
## 276	1.0000000	NA
## 290	0.3333333	NA
## 306	0.0000000	NA
## 324	1.0000000	NA
## 331	0.5000000	NA
## 332	1.0000000	NA
## 338	1.0000000	NA
## 342	0.0000000	NA
## 348	1.0000000	NA
## 356	0.5000000	NA
## 366	0.0000000	NA
## 378	0.5000000	NA
## 387	0.0000000	NA
## 401	0.0000000	NA
## 411	0.0000000	NA
## 414	1.0000000	NA
## 421	1.0000000	NA
## 424	1.0000000	NA
## 425	0.5000000	NA
## 429	0.0000000	NA
## 431	1.0000000	NA
## 433	0.5000000	NA
## 436	1.0000000	NA
## 439	0.3333333	NA
## 448	0.3333333	NA
## 510	0.2500000	NA
## 533	0.5000000	NA
## 538	1.0000000	NA
## 544	1.0000000	NA
## 550	0.2500000	NA
## 561	0.5000000	NA
## 598	0.2500000	NA
## 602	0.5000000	NA
## 606	1.0000000	NA
## 626	0.3333333	NA

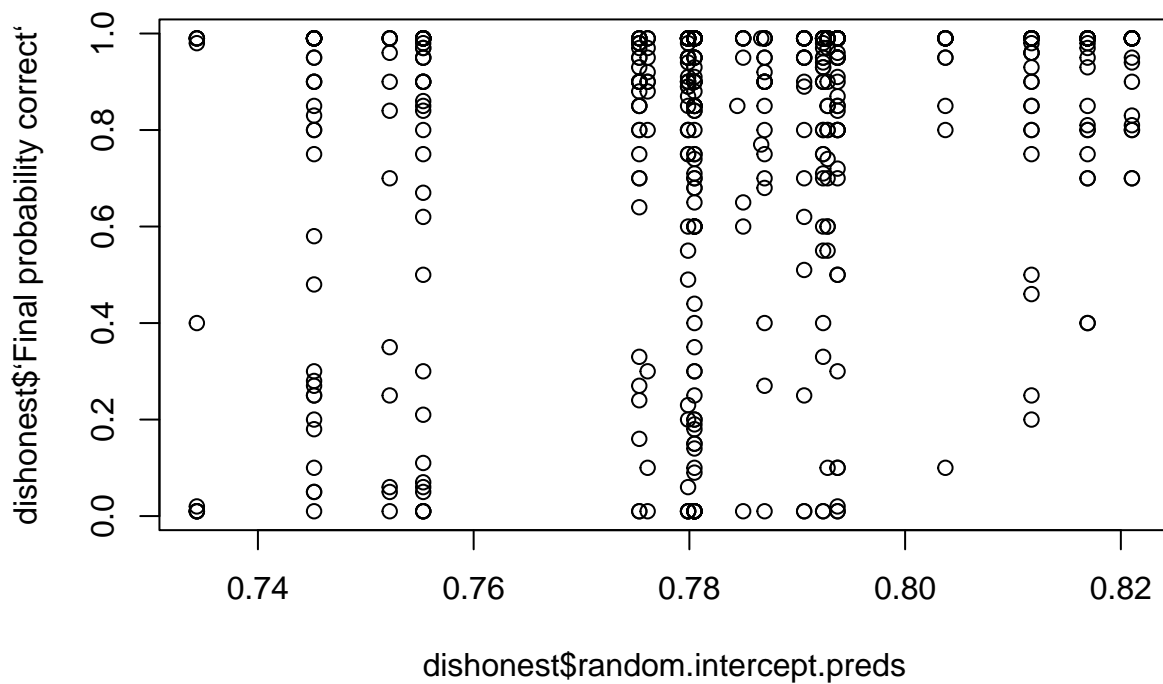
```
## 637          0.5000000          NA
## 641          0.5000000          NA
## 647          1.0000000          NA
## 648          0.3333333          NA
## 658          0.5000000          NA
## 677          0.5000000          NA
## 679          0.3333333          NA
## 680          1.0000000          NA
## 683          1.0000000          NA
```

```
# Fit the random intercept model and only remove missing values for 'Dishonest debater'
random_intercept_model <- lmer(`Final probability correct` ~ (1|`Dishonest debater`),
                               data = dishonest,
                               REML = TRUE)
```

```
# Summary of the model
summary(random_intercept_model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: 'Final probability correct' ~ (1 | 'Dishonest debater')
## Data: dishonest
##
## REML criterion at convergence: 302.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.5213 -0.1985  0.5027  0.6588  0.8225
##
## Random effects:
## Groups           Name          Variance Std.Dev.
## Dishonest debater (Intercept) 0.001765 0.04201
## Residual                    0.096628 0.31085
## Number of obs: 577, groups: Dishonest debater, 20
##
## Fixed effects:
##              Estimate Std. Error    df t value      Pr(>|t|)
## (Intercept)  0.78325    0.01719  7.54926  45.58 0.000000000172 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dishonest$random.intercept.preds = predict(random_intercept_model)
plot(dishonest$random.intercept.preds, dishonest$`Final probability correct`)
```



Debater “Experience”, ratings - how many wins?

AI vs Humans

Old vs New

possibly unnecessary

Finally, these are how many we get correct in each setting

```
judgments_online <- py$judgments_online
table(judgments_online$Final_Accuracy, judgments_online$Final_Setting)
```

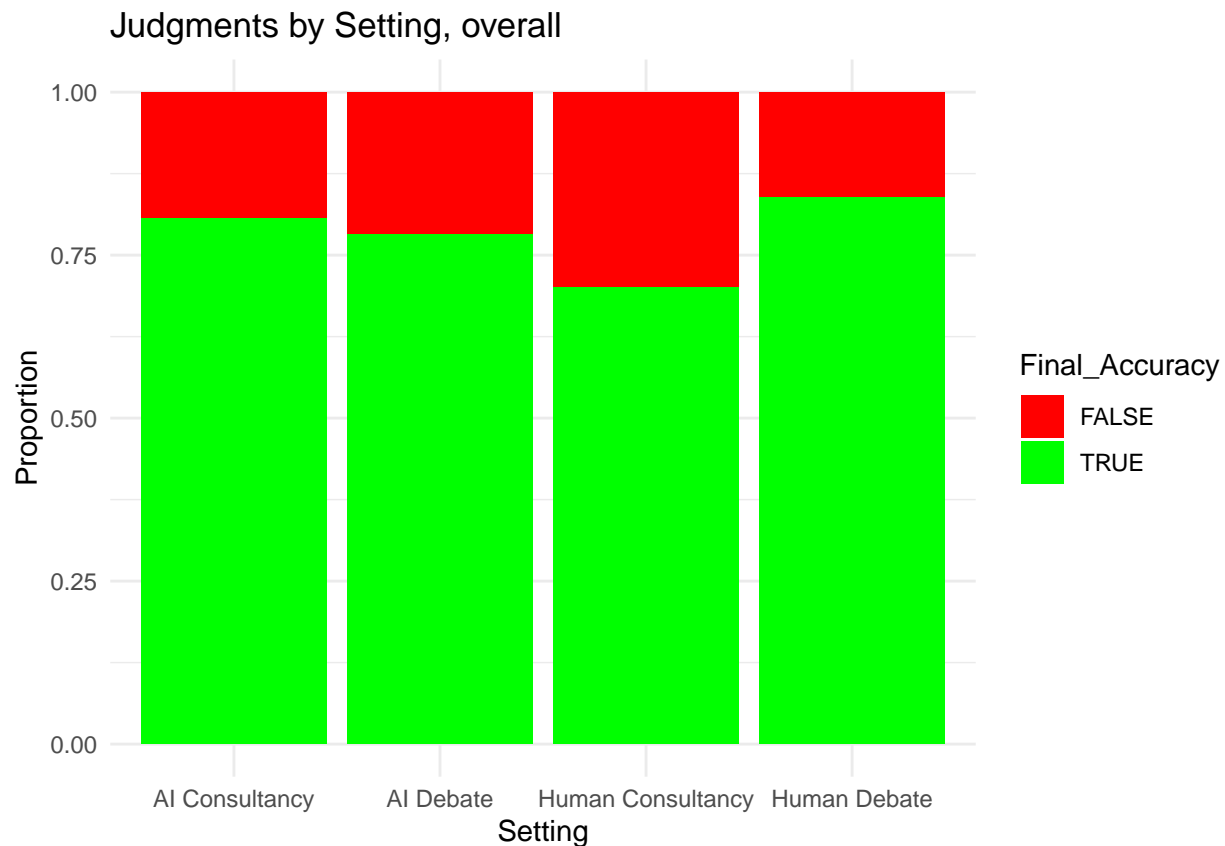
```
##
##      AI Consultancy AI Debate Human Consultancy Human Debate
## FALSE           18         19             32             25
## TRUE            75         68             75            130
```

```
table(judgments_online$Final_Accuracy, judgments_online$Setting)
```

```
##
##      AI Consultancy Dishonest AI Consultancy Honest AI Debate
## FALSE              5             13         19
## TRUE              33             42         68
```

```
##
##           Human Consultancy Dishonest Human Consultancy Honest Human Debate
## FALSE                26                        6                25
## TRUE                 33                       42               130
```

```
ggplot(judgments_online, aes(x = Final_Setting, fill = Final_Accuracy)) +
  geom_bar(position = "fill") +
  scale_fill_manual(values = c("TRUE" = "green", "FALSE" = "red")) +
  labs(title = "Judgments by Setting, overall", x = "Setting", y = "Proportion", fill = "Final_Accuracy") +
  theme_minimal() +
  theme(axis.text.x = element_text())
```



Sneak peak of accuracy differences between judges, but we won't get to that again until models

```
ggplot(judgments_online, aes(x = Final_Setting, fill = Final_Accuracy)) +
  geom_bar(position = "fill") +
  scale_fill_manual(values = c("TRUE" = "green", "FALSE" = "red")) +
  labs(title = "Judgments by Setting, per judge", x = "Setting", y = "Proportion", fill = "Final_Accuracy") +
  theme_minimal() +
  theme(axis.text.x = element_text(), #angle = 90, hjust = 1),
        axis.text.y = element_blank(),
        strip.text.y.right = element_text(angle = 0)) +
  facet_grid(rows = "Participant")
```

