In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy.spatial import distance
import seaborn as sns
```

In [6]:

```
acceptable_submission = df['survey_id'].value_counts() >= 5
usable_id = []
for i in range(len(acceptable_submission.to_list())):
    if acceptable_submission.to_list()[i] == True:
        usable_id.append(acceptable_submission.index[i])

model_df = df[df['survey_id'].isin(usable_id)]
model_df
```

Out[6]:

	id	survey_id	x	y	sound	concept	concept_other	date_
0	29	13	36.372925	195.413374	Where	other	where	
1	30	13	76.832695	176.607903	Want	WANT	None	
2	31	13	110.753512	196.639818	Love you	LOVE-YOU	None	
3	32	13	39.642401	154.531915	Bye	BYE	None	
4	33	13	114.022989	155.758359	Hi	HI	None	
...	...	...	...	...	...	...	...	
2575	2643	424	3556.363636	702.752885	Spielen	Play	None	00:0
2576	2644	424	545.454545	1592.506731	Garten	Garden	None	00:0
2577	2645	424	1479.272727	1649.206731	Später	Lat	None	00:0
2578	2646	424	2592.000000	1623.037500	Nein	No	None	00:0
2579	2647	424	3512.727273	1614.314423	Ja	yes	None	00:0

2166 rows × 8 columns

In [7]:

```
# Change sound to lowercase
model_df['sound'] = model_df['sound'].str.lower()
model_df.head()
```

```
<ipython-input-7-1ba44d8f91d3>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  model_df['sound'] = model_df['sound'].str.lower()
```

Out[7]:

	id	survey_id	x	y	sound	concept	concept_other	date_introduced
0	29	13	36.372925	195.413374	where	other	where	Na
1	30	13	76.832695	176.607903	want	WANT	None	Na
2	31	13	110.753512	196.639818	love you	LOVE-YOU	None	Na
3	32	13	39.642401	154.531915	bye	BYE	None	Na
4	33	13	114.022989	155.758359	hi	HI	None	Na

In [8]:

```
dictionary = {"love you": "love", "chill out": "calm",
              "all done": "done", "grammy brenda": "brenda",
             "tennis ball": "ball", "jon snow": "jon",
             "i love you": "love", "go for a walk": "walk",
             "???": "question", "golf cart": "cart",
              "dog park": "park", "clean up": "tidy",
             "high five": "hand", "brush teeth": "teeth",
             "chew toy": "toy",
             "excuse me, i have something to say": "talk",
             "put down your phone please": "attention",
             "thank you": "gratitude", "??? hmmm": "question",
             "hmm?": "question", "yes!": "yes",
             "it's raining": "raining", "go upstairs": "upstairs",
             "flirt pole": "toy", "find it": "find",
             "go potty": "bathroom",
             "mousey": "mouse", "go walk": "walk",
             "go for a ride": "ride", "walkies": "walk",
             "mmm": "question", "go to bed": "bed",
             "its okay": "okay", "grandma and jessie": "relatives",
             "n'nights": "bed", "poopsie": "poop",
             "hug me": "hug", "hmmm???": "question",
             "num nums": "food", "won-ton": "toy",
             "i love you's": "love", "ice cube": "treat",
             "play toys": "play", "loveyou": "love",
             "pet me": "pet", "booboo": "hurt",
              "momia": "mama", "bike ride": "bike",
             "sniffsniff": "sniff", "stuffy": "toy",
             "alldone": "done", "hmmmm": "question",
             "mmm?": "question", "chicken with gravy": "food",
             "people food": "food", "go fetch": "fetch",
             "water please": "water",
              "head scratches please": "scratches",
             "scritches": "scratches", "outside please": "outside",
             "let's play": "play", "antler please": "chew",
             "call mom": "mom", "play doggie music": "music",
             "when?": "when", "nite nite": "bed",
             "treat ball": "ball", "snug snug": "cuddle",
             "snicky snack": "snack", "choo choo": "trick",
             "night night": "goodnight", "supper time": "eat",
```

```
                "come here": "come", "huh?": "question",
                "mumma": "mom", "porch ": "porch", "poopoo": "poop", "peepee"
```

In [9]:

```
model_df = model_df.replace(dictionary)
model_df.head()
```

Out[9]:

	id	survey_id	x	y	sound	concept	concept_other	date_introduced
0	29	13	36.372925	195.413374	where	other	where	Na
1	30	13	76.832695	176.607903	want	WANT	None	Na
2	31	13	110.753512	196.639818	love	LOVE-YOU	None	Na
3	32	13	39.642401	154.531915	bye	BYE	None	Na
4	33	13	114.022989	155.758359	hi	HI	None	Na

In [10]:

```
import gensim.downloader as api

word_vectors = api.load("glove-wiki-gigaword-100")  # load pre-trained wor
```

In [11]:

```
### Use language model with linear cost function

def linear_cost(x):
    x = (x.max(axis=1) - x + 1)
    sum_of_rows = x.sum(axis=1)
    normalized_array = x / sum_of_rows[:, np.newaxis]
    return normalized_array
```

In [12]:

```
### semantic similarity
def semantic(list_of_words):
    percentage = []
    for i in np.arange(len(list_of_words)):
        each_word = []
        for j in np.arange(len(list_of_words)):
            each_word.append(word_vectors.similarity(list_of_words[i], lis
        percentage.append(each_word)
    return np.array(percentage)
```

In [13]:

```
### Compute kendall tau of a single board
def correlation_of_single_board(button_df):

    # get buttons from df
    buttons = [Button(row["sound"], row["concept"], row["x"], row["y"]) fo
    b = ButtonsBiGramLanguageModel(buttons, linear_cost)

    # Rankings of semantic similarity
    semantic_ranks = np.argsort(np.abs(semantic(button_df['sound'].to_list
    for i in np.arange(button_df.shape[0]):
```

```
        semantic_ranks[i] = semantic_ranks[i][::-1]

    # Rankings of physical distance
    distance_ranks = np.argsort(b.G)

    # kendall tau correlation between semantic and physical distance
    tau_list = []
    for i in np.arange(button_df.shape[0]):
        tau, p_value = scipy.stats.kendalltau(semantic_ranks[i], distance_
        tau_list.append(tau)

    # Mean and variance of a single board
    tau_avg = np.mean(tau_list)
    tau_var = np.var(tau_list, ddof=1)

    return (tau_avg, tau_var)
```

In [14]:

```
### ideal:    permutation_test(survey_id)
def permutation_test(button_df, n_repetitions = 500, jaccard_top_numbers =

    # get buttons from df
    buttons = [Button(row["sound"], row["concept"], row["x"], row["y"]) fo
    b = ButtonsBiGramLanguageModel(buttons, linear_cost)

    # Rankings of semantic similarity
    semantic_ranks = np.argsort(np.abs(semantic(button_df['sound'].to_list
    for i in np.arange(button_df.shape[0]):
        semantic_ranks[i] = semantic_ranks[i][::-1]

    # Rankings of physical distance
    distance_ranks = np.argsort(b.G)

    # kendall tau correlation for og semantic ranking and og distance rank
    tau_list = []
    for i in np.arange(button_df.shape[0]):
        tau, p_value = scipy.stats.kendalltau(semantic_ranks[i], distance_
        tau_list.append(tau)

    # Average correlations
    tau_avg = np.mean(tau_list)

    ### Begin permutation test
    shuffled_tau_list = []          ### number of buttons of tau after 1 s

    shuffled_tau_variances = []     ### n_repetitions number of tau_varian
    shuffled_tau_means = []         ### n_repetitions number of tau after
    final_shuffled_jaccard = []     ### n_repetitions number of jaccard di

    for _ in range(n_repetitions):

        # Shuffle buttons
        indices = np.arange(button_df.shape[0])
        np.random.shuffle(indices)     ### Randomly shuffled indices
        shuffled_distance = b.G[indices,:][:, indices]
```

```
        # Rankings of shuffled physical distance
        shuffled_distance_ranks = np.argsort(shuffled_distance)

        # kendall tau correlation for og semantic ranking and shuffled dis
        shuffled_tau_list = []
        shuffled_jaccard = []
        for i in np.arange(button_df.shape[0]):
            ### compute kendall tau
            tau, p_value = scipy.stats.kendalltau(semantic_ranks[i], shuff
            shuffled_tau_list.append(tau)            ### button_df.shape[0

            ### compute jaccard
            top_5_semantic = list(semantic_ranks[i][:jaccard_top_numbers])
            top_5_distance = list(shuffled_distance_ranks[i][:jaccard_top_
            shuffled_jaccard.append(distance.jaccard(top_5_semantic, top_5


        # Get the mean and variance of tau, and jaccard distance from one
        shuffled_tau_means.append(np.mean(shuffled_tau_list))
        shuffled_tau_variances.append(np.var(shuffled_tau_list))
        final_shuffled_jaccard.append(np.mean(shuffled_jaccard))

    ### return list of tau means and tau variances
    return (shuffled_tau_means, shuffled_tau_variances, final_shuffled_jac
```

In [15]:

```
### Compute jaccard distance on button_df
def single_jaccard(button_df, top_numbers_of = 5):

    # get buttons from df
    buttons = [Button(row["sound"], row["concept"], row["x"], row["y"]) fo
    b = ButtonsBiGramLanguageModel(buttons, linear_cost)

    # Rankings of semantic similarity
    semantic_ranks = np.argsort(np.abs(semantic(button_df['sound'].to_list
    for i in np.arange(button_df.shape[0]):
        semantic_ranks[i] = semantic_ranks[i][::-1]

    # Rankings of physical distance
    distance_ranks = np.argsort(b.G)

    # Compute jaccard distance
    jaccard = []
    for button in np.arange(button_df.shape[0]):
        top_5_semantic = list(semantic_ranks[button][:top_numbers_of])
        top_5_distance = list(distance_ranks[button][:top_numbers_of])
        jaccard.append(distance.jaccard(top_5_semantic, top_5_distance))

    return np.mean(jaccard)
```

Survey_id 47

In [16]:

```
survey_47 = model_df.loc[model_df['survey_id'] == 47]
```

```
survey_47.head()
```

Out[16]:

	id	survey_id	x	y	sound	concept	concept_other	date_i
233	242	47	95.810277	102.554921	frustrated	Frustrated	None	2 00:00
234	243	47	91.541502	120.112039	concerned	Concerned	None	2 00:00
235	244	47	67.826087	220.235062	tidy	Put toys in toy box	None	2 00:00
236	245	47	73.517787	195.085677	hand	Someone will come high five Pixel	None	2 00:00
237	246	47	95.810277	186.069859	water	Pixel needs more water	None	2 00:00

In [17]:

```
### Permutation test on survey 47

survey_47_permutation = permutation_test(survey_47, 5000)
tau_mean_47 = survey_47_permutation[0]
tau_var_47 = survey_47_permutation[1]
jaccard_47 = survey_47_permutation[2]
```

In [18]:

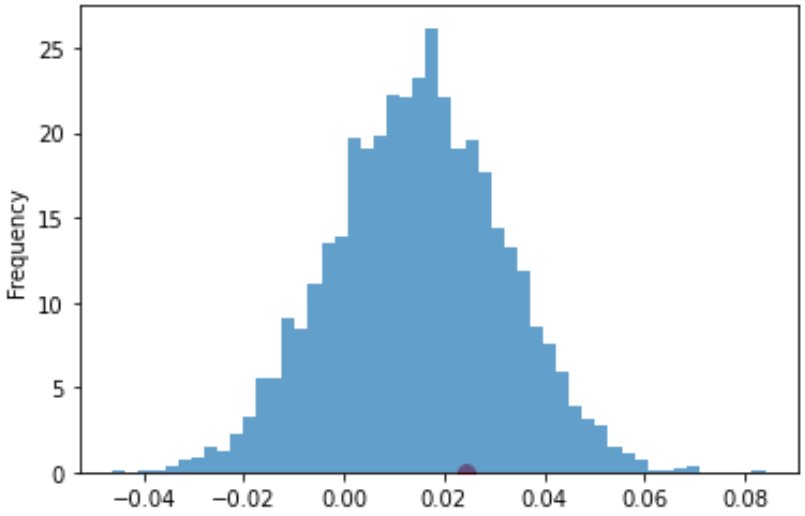
```
### A single correlation on survey 47

survey_47_correlation = correlation_of_single_board(survey_47)
observed_tau_mean_47 = survey_47_correlation[0]
observed_tau_var_47 = survey_47_correlation[1]
observed_jaccard_47 = single_jaccard(survey_47)
print(observed_tau_mean_47)
```

```
0.024268309484913497
```

Kendall tau correlation on survey_id 47

In [19]:

```
pd.Series(tau_mean_47).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_tau_mean_47, 0, color='red', s=60);
```

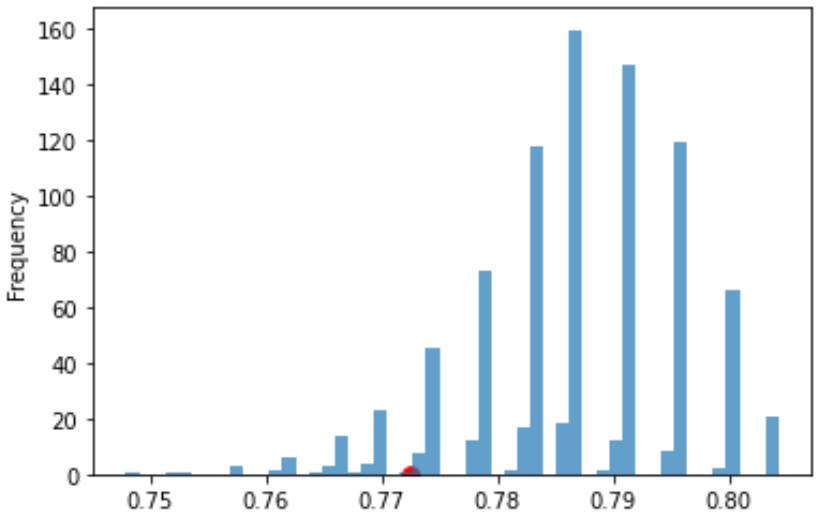


In [20]:
```
### number of tau_mean from simulation >= a single tau_mean from survey_47
np.count_nonzero(tau_mean_47 >= observed_tau_mean_47) / len(tau_mean_47)
```

Out[20]: 0.2916

Jaccard distance of survey_id 47

In [21]:
```
pd.Series(jaccard_47).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_47, 0, color='red', s=60);
```



In [22]:
```
np.count_nonzero(jaccard_47 >= observed_jaccard_47) / len(jaccard_47)
```

Out[22]: 0.9352

In [ ]:
```

```

In [23]:
```
### jaccard distance swapping in for kendall tau
```

```
A = [1,2,3]  #top 5 physical distance
B = [2,3,4]  # top 5 semantic distance
```

In [24]:

```
### set theory
len(set(A).intersection(set(B)))/len(set(A).union(set(B)))
```

Out[24]: 0.5

In [ ]:

Survey_id 44

In [25]:

```
survey_44 = model_df.loc[model_df['survey_id'] == 44]
survey_44.head()
```

Out[25]:

	id	survey_id	x	y	sound	concept	concept_other	date_intro
157	166	44	928.565217	343.636364	want	Want	None	2020-00:00:00+
158	167	44	1280.739130	357.734266	where	WHERE	None	2020-00:00:00+
159	168	44	1473.260870	705.482517	no	NO	None	2020-00:00:00+
160	169	44	1294.826087	987.440559	go	GO	None	2019 00:00:00+
161	170	44	1092.913043	672.587413	love	LOVE YOU	None	2019 00:00:00+

In [26]:

```
### Permutation test on survey_44

survey_44_permutation = permutation_test(survey_44, 5000)
tau_mean_44 = survey_44_permutation[0]
tau_var_44 = survey_44_permutation[1]
jaccard_44 = survey_44_permutation[2]
```

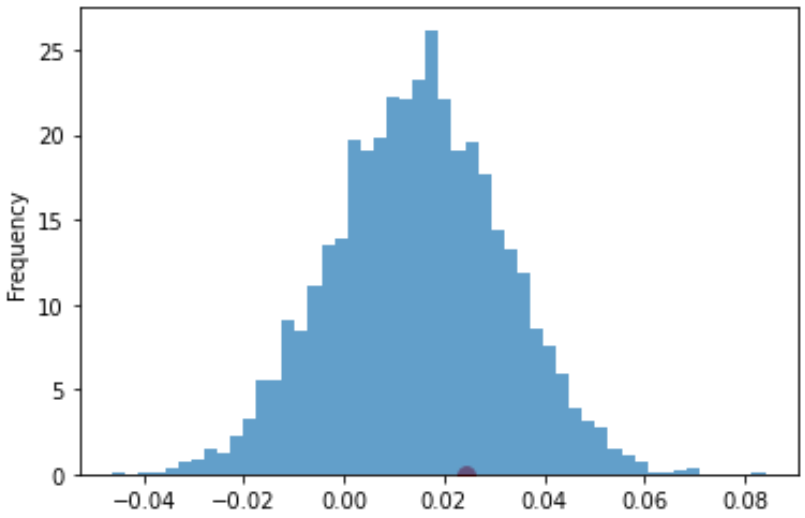
In [27]:

```
### A single correlation on survey 44

survey_44_correlation = correlation_of_single_board(survey_44)
observed_tau_mean_44 = survey_44_correlation[0]
observed_tau_var_44 = survey_44_correlation[1]
observed_jaccard_44 = single_jaccard(survey_44)
print(observed_tau_mean_44)
print(observed_jaccard_44)
```

```
0.06102292768959436
0.7666666666666668
```

Kendall tau correlation on survey_id 44

In [28]:
```
pd.Series(tau_mean_47).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_tau_mean_47, 0, color='red', s=60);
```

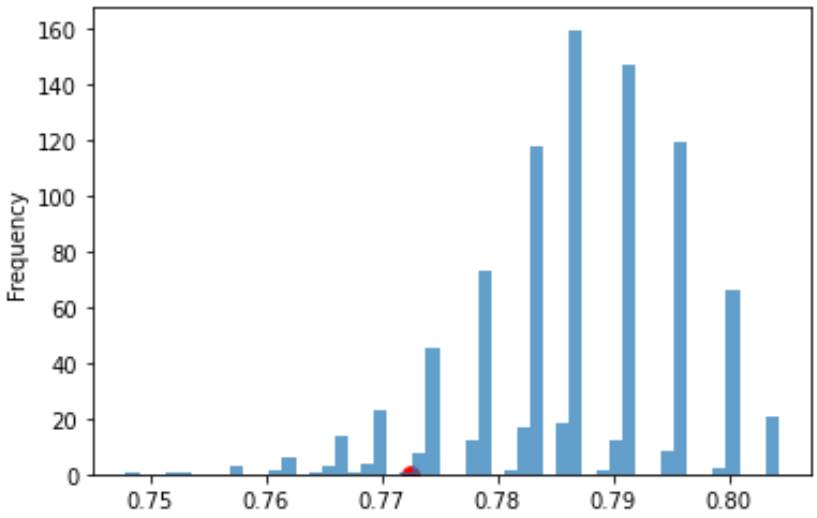


In [29]:
```
### number of tau_mean from simulation >= a single tau_mean from survey_44
np.count_nonzero(tau_mean_44 >= observed_tau_mean_44) / len(tau_mean_44)
```

Out[29]: 0.0156

Jaccard distance of survey_id 44

In [30]:
```
pd.Series(jaccard_47).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_47, 0, color='red', s=60);
```



In [31]:
```
np.count_nonzero(jaccard_44 >= observed_jaccard_44) / len(jaccard_44)
```

0.937

Out[31]:

In [ ]:

Survey_id 61

In [32]:
```
survey_61 = model_df.loc[model_df['survey_id'] == 61]
```

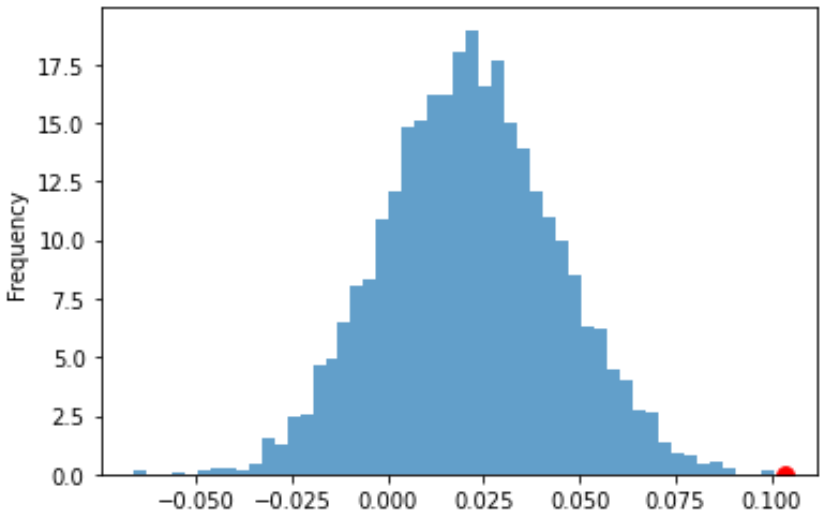
In [33]:
```
### Permutation test on survey_61

survey_61_permutation = permutation_test(survey_61, 5000)
tau_mean_61 = survey_61_permutation[0]
tau_var_61 = survey_61_permutation[1]
jaccard_61 = survey_61_permutation[2]
```

In [34]:
```
### A single correlation on survey 61

survey_61_correlation = correlation_of_single_board(survey_61)
observed_tau_mean_61 = survey_61_correlation[0]
observed_tau_var_61 = survey_61_correlation[1]
observed_jaccard_61 = single_jaccard(survey_61)
print(observed_tau_mean_61)
print(observed_jaccard_61)
```

```
0.10365013774104682
0.7696969696969698
```

Kendall tau correlation on survey_id 61

In [35]:
```
pd.Series(tau_mean_61).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_tau_mean_61, 0, color='red', s=60);
```



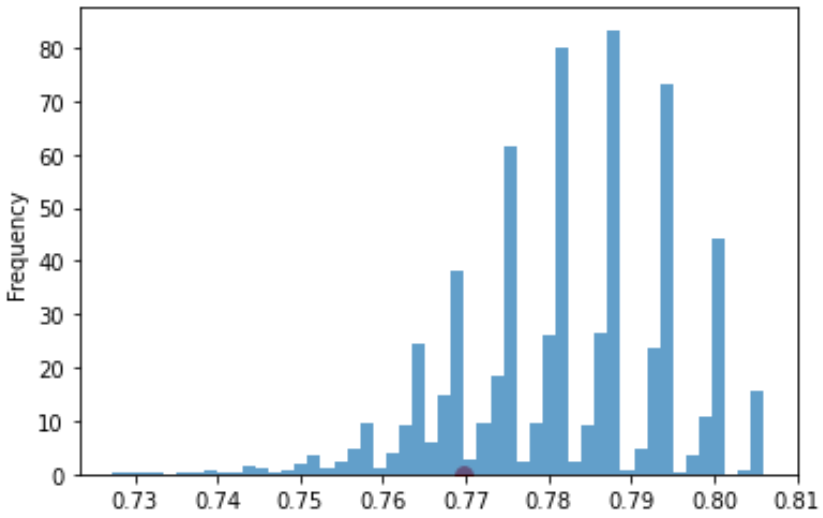
In [36]:

```
### number of tau_mean from simulation >= a single tau_mean from survey_44
np.count_nonzero(tau_mean_61 >= observed_tau_mean_61) / len(tau_mean_61)
```

Out[36]: 0.0

Jaccard distance on survey_id 61

In [37]:
```
pd.Series(jaccard_61).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_61, 0, color='red', s=60);
```



In [38]:
```
np.count_nonzero(jaccard_61 >= observed_jaccard_61) / len(jaccard_61)
```

Out[38]: 0.8602

In [ ]:
```

```

Survey_id 53

In [39]:
```
survey_53 = model_df.loc[model_df['survey_id'] == 53]
```

In [40]:
```
### Permutation test on survey_53

survey_53_permutation = permutation_test(survey_53, 5000)
tau_mean_53 = survey_53_permutation[0]
tau_var_53 = survey_53_permutation[1]
jaccard_53 = survey_53_permutation[2]
```

In [41]:
```
### A single correlation on survey 53

survey_53_correlation = correlation_of_single_board(survey_53)
observed_tau_mean_53 = survey_53_correlation[0]
```

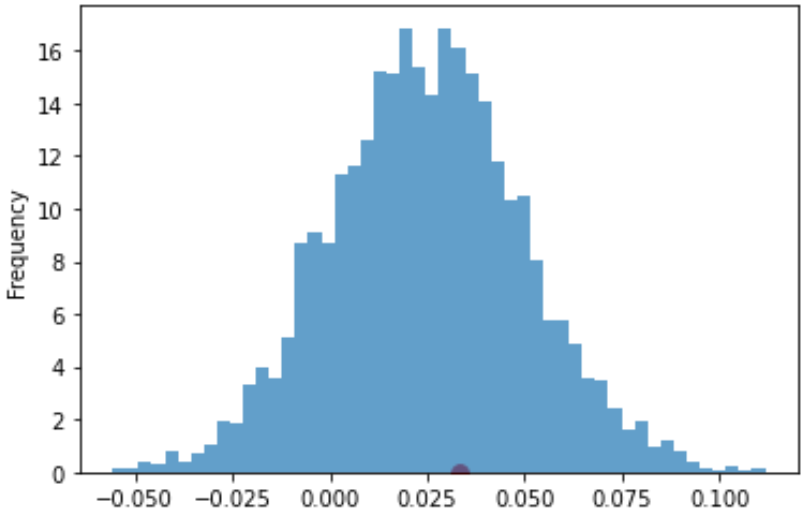
```
observed_tau_var_53 = survey_53_correlation[1]
observed_jaccard_53 = single_jaccard(survey_53)
print(observed_tau_mean_53)
print(observed_jaccard_53)
```

```
0.03346356378461015
0.7448275862068967
```

Kendall tau correlation on survey_id 53

In [42]:

```
pd.Series(tau_mean_53).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_tau_mean_53, 0, color='red', s=60);
```



In [43]:

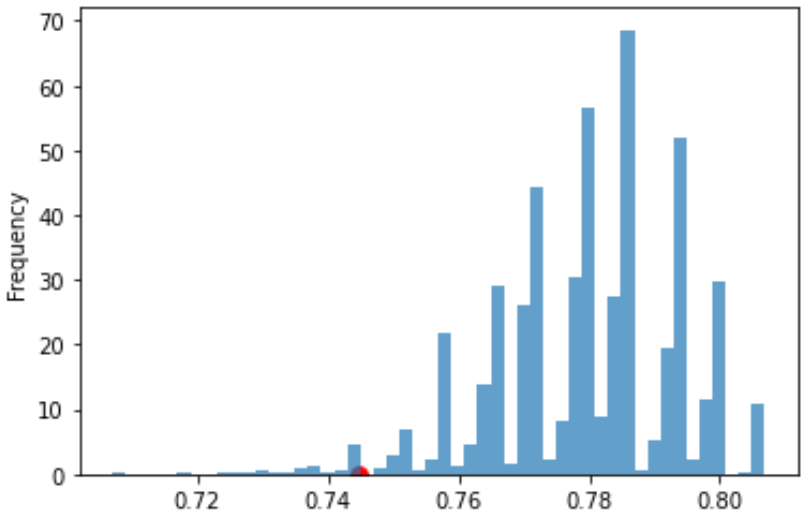
```
### number of tau_mean from simulation >= a single tau_mean from survey_53
np.count_nonzero(tau_mean_53 >= observed_tau_mean_53) / len(tau_mean_53)
```

Out[43]: 0.3638

Jaccard distance on survey_id 53

In [44]:

```
pd.Series(jaccard_53).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_53, 0, color='red', s=60);
```



In [45]:
```
np.count_nonzero(jaccard_53 >= observed_jaccard_53) / len(jaccard_53)
```

Out[45]: 0.987

In [ ]:
```
```

Survey_id 13

In [46]:
```
survey_13 = model_df.loc[model_df['survey_id'] == 13]
```

In [47]:
```
### Permutation test on survey_13

survey_13_permutation = permutation_test(survey_13, 5000)
tau_mean_13 = survey_13_permutation[0]
tau_var_13 = survey_13_permutation[1]
jaccard_13 = survey_13_permutation[2]
```

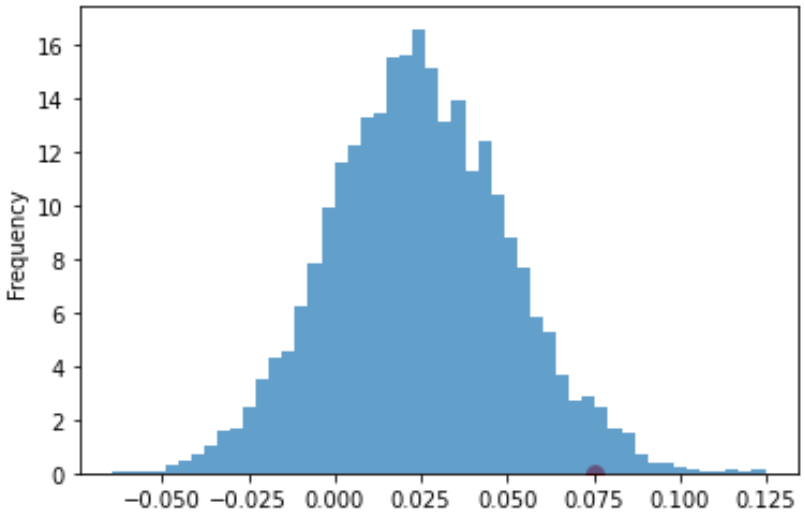
In [48]:
```
### A single correlation on survey 13

survey_13_correlation = correlation_of_single_board(survey_13)
observed_tau_mean_13 = survey_13_correlation[0]
observed_tau_var_13 = survey_13_correlation[1]
observed_jaccard_13 = single_jaccard(survey_13)
print(observed_tau_mean_13)
print(observed_jaccard_13)
```

```
0.07525055206386953
0.7517241379310348
```

Kendall tau correlation on survey_id 13

In [49]:
```
pd.Series(tau_mean_13).plot(kind='hist', bins=50, density=True, alpha=0.7)
```

```
plt.scatter(observed_tau_mean_13, 0, color='red', s=60);
```

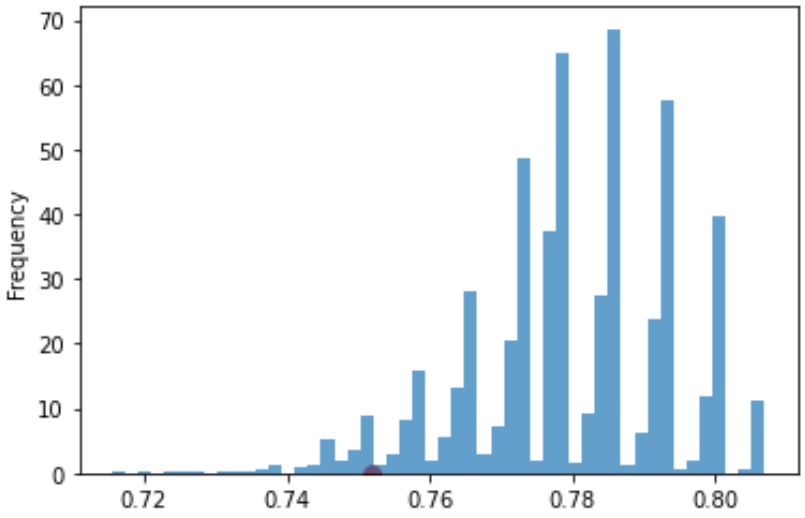


In [50]:
```
### number of tau_mean from simulation >= a single tau_mean from survey_13
np.count_nonzero(tau_mean_13 >= observed_tau_mean_13) / len(tau_mean_13)
```

Out[50]: 0.0302

Jaccard distance on survey_id 13

In [51]:
```
pd.Series(jaccard_13).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_13, 0, color='red', s=60);
```



In [52]:
```
np.count_nonzero(jaccard_13 >= observed_jaccard_13) / len(jaccard_13)
```

Out[52]: 0.9618

In [ ]:

Survey_id 57

In [53]:
```
survey_57 = model_df.loc[model_df['survey_id'] == 57]
survey_57.head()
```

Out[53]:

	id	survey_id	x	y	sound	concept	concept_other	da
302	352	57	447.006085	405.808219	want	WANT	None	0
303	353	57	661.691684	326.027397	what	what	None	0
304	354	57	864.109533	510.136986	no	NO	None	0
305	355	57	618.754564	577.643836	hmm	INTERROGATIVE-QUESTION	None	0
306	356	57	808.904665	743.342466	yes	YES	None	0

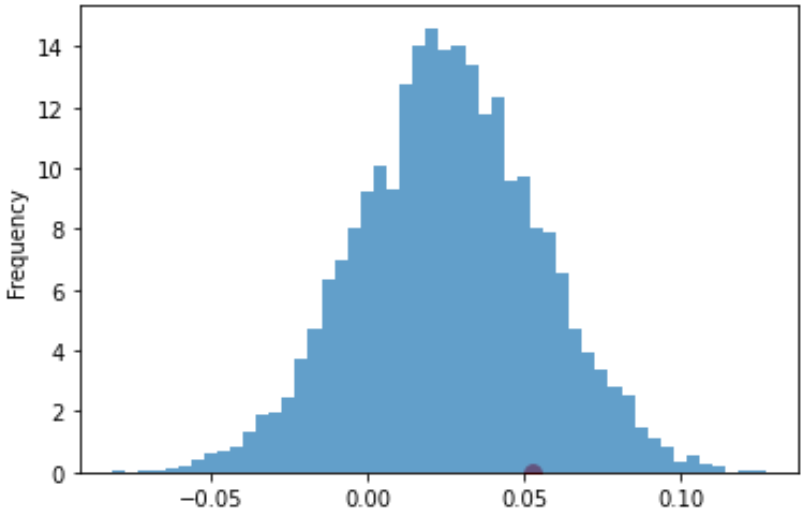
In [54]:
```
### Permutation test on survey_57

survey_57_permutation = permutation_test(survey_57, 5000)
tau_mean_57 = survey_57_permutation[0]
tau_var_57 = survey_57_permutation[1]
jaccard_57 = survey_57_permutation[2]
```

In [55]:
```
### A single correlation on survey 57

survey_57_correlation = correlation_of_single_board(survey_57)
observed_tau_mean_57 = survey_57_correlation[0]
observed_tau_var_57 = survey_57_correlation[1]
observed_jaccard_57 = single_jaccard(survey_57)
print(observed_tau_mean_57)
print(observed_jaccard_57)
```

```
0.05291005291005291
0.760714285714286
```

Kendall tau correlation on survey_id 57

In [56]:
```
pd.Series(tau_mean_57).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_tau_mean_57, 0, color='red', s=60);
```

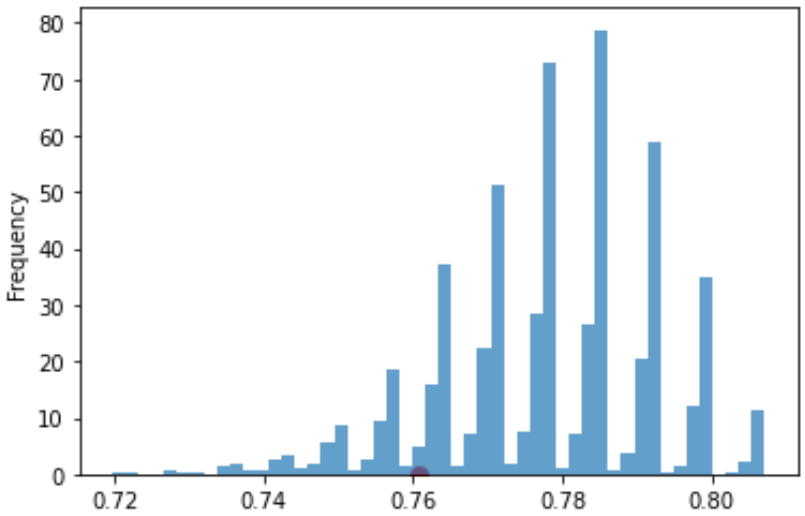


In [57]:
```
### number of tau_mean from simulation >= a single tau_mean from survey_57
np.count_nonzero(tau_mean_57 >= observed_tau_mean_57) / len(tau_mean_57)
```

Out[57]: 0.1782

Jaccard distance on survey_id 57

In [58]:
```
pd.Series(jaccard_57).plot(kind='hist', bins=50, density=True, alpha=0.7)
plt.scatter(observed_jaccard_57, 0, color='red', s=60);
```



In [59]:
```
np.count_nonzero(jaccard_57 >= observed_jaccard_57) / len(jaccard_57)
```

Out[59]: 0.8872

In [ ]: