Case Study - Multiple testing

Synopsis

The management team at AAVAIL is preparing to deploy a large number of teams each tasked with integration into a different new market. They claim to have a optimized the teams fairly with respect to skills and experience. They are asking you to come up with a framework to evaluate the makeup of their teams. They have not finished hiring and creating all of the teams so naturally before you even get the data you wanted to get a head start.

Getting a head start usually involves finding a similar dataset and writing the code in a way that the new data, once obtained can be added with little effort.

When we perform a large number of statistical tests, some will have p-values less than the designated level of α (e.g. 0.05) purely by chance, even if all the null hypotheses are really true. This is an inherent risk of using inferrential statistics. Fortunately, there are several techniques to mitigate the risk.

We are going to look at the 2018 world cup data in this example.

The case study is comprised of the following sections:

1. Data Cleaning
2. Data Visualization
3. NHT
4. Adjust NHT results for multiple comparisons

Data science work that focuses on creating a predictive model is perhaps the hallmark of the field today, but there are still many use cases where inferential statistics are the best tool available. One issue with statistical inference is that there are situations where performing multiple tests is a logical way to accomplish a task, but it comes at the expense of an increased rate of false positives or Type I errors.

In this case study you will apply techniques and knowledge from all of the units in Module 2.

Getting started

This unit is interactive. During this unit we encourage you to open this file as a notebook. Download the notebook from the following link then open it locally using a Jupyter server or use your IBM cloud account to login to Watson Studio. Inside of Waston Studio cloud if you have not already ensure that this notebook is loaded as part of the *project* for this course. As a reminder fill in all of the places in this notebook marked with ***YOUR CODE HERE*** or ***YOUR ANSWER HERE***. The data and notebook for this unit are available below.

- m2-u7-case-study.ipynb
- worldcup-2018.csv

This unit is organized into the following sections:

1. Data Processing
2. Data Summary
3. Investigative Visualization
4. Hypothesis testing

Resources

- [Creating or uploading a notebook in IBM cloud]
- [Resources for multiple testing in Python]

In [1]:
```
%%capture
! pip install pingouin
```

In [2]:
```
import os
import re
import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels as sm
import pingouin

import matplotlib.pyplot as plt
plt.style.use('seaborn')

%matplotlib inline

SMALL_SIZE = 8
MEDIUM_SIZE = 10
LARGE_SIZE = 12

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=LARGE_SIZE)   # fontsize of the figure title
```

Import the Data

Before we jump into the data it can be useful to give a little background so that you can better understand the features. Since the dawn of statistics practitioners have been trying to find advantages when it comes to games. Much of this was motivated by gambling---here we will look at the results from this tournament in a different way. We are going to ask the simple question

> Was the tournament setup in a fair way?

Of course the findings from an investigation centering around this question could be used to strategically place bets, but lets assume that we are simply interested in whether or not the tournament organizers did an adequate job. The reason for doing this is to prepare for the AAVAIL data that is coming. This exercise is an important reminder that you do not have to wait until the day that data arrive to start your work.

There are 32 teams, each representing a single country, that compete in groups or pools then the best teams from those groups compete in a single elimination tournament to see who will become world champions. This is by far the world's most popular sport so one would hope that the governing organization FIFA did a good job composing the pools. If for example there are 8 highly ranked teams then each of those teams should be in a different pool.

In our data set we have more than just rank so we can dig in a little deeper than that, but first let's have a look at the data

In [3]:

```
DATA_DIR = os.path.join("..","data")
df = pd.read_csv(os.path.join(DATA_DIR, 'worldcup-2018.csv'))
df.columns = [re.sub("\s+","_",col.lower()) for col in df.columns]
df.head()
```

Out[3]:

	team	group	previous_appearances	previous_titles	previous_finals	previous_semifinals	curre
0	Russia	A	10	0	0	1	
1	Saudi Arabia	A	4	0	0	0	
2	Egypt	A	2	0	0	0	
3	Uruguay	A	12	2	2	5	
4	Porugal	B	6	0	0	2	

To limit the dataset for educational purposes we create a new data frame that consists of only the following columns:

- team
- group
- previous_appearances
- previous_titles
- previous_finals
- previous_semifinals
- current_fifa_rank

Data Processing

QUESTION 1

Using the column names below create a new dataframe that uses only them.

In [4]:
```
columns = ['team', 'group','previous_appearances','previous_titles','previo
           'previous_semifinals','current_fifa_rank']

### YOUR CODE HERE
df = df.loc[:, columns]
df.head()
```

Out[4]:

	team	group	previous_appearances	previous_titles	previous_finals	previous_semifinals	curre
0	Russia	A	10	0	0	1	
1	Saudi Arabia	A	4	0	0	0	
2	Egypt	A	2	0	0	0	
3	Uruguay	A	12	2	2	5	
4	Porugal	B	6	0	0	2	

To help with this analysis we are going to engineer a feature that combines all of the data in the table. This feature represents the past performance of a team. Given the data we have it is the best proxy on hand for how good a team will perform. Feel free to change the multipliers, but let's just say that `past_performance` will be a linear combination of the related features we have.

Let X_1X1,...,
X_4X4 be `previous_titles` , `previous_finals` , `previous_semifinals` , `previous_appe`
let the corresponding vector αα be the multipliers. This will give us,

$$\text{past\_performance} = \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \alpha_4 X_4$$
past_performance=α1X1+α2X2+α3X3+α4X4

Modify αα if you wish. Then add to your dataframe the new feature `past_performance` .

QUESTION 2

create the engineered feature `past_performance` as a new column

In [5]:
```
alpha = np.array([16,8,4,1])

### YOUR CODE HERE
df['past_performance'] = alpha[0] * df['previous_titles'] + alpha[1] * df['
```

Data Summary

QUESTION 3

Using the `pivot_table` function create one or more **tabular summaries** of the data

In [6]:
```
### YOUR CODE HERE
columns_to_show =['previous_appearances','previous_titles','previous_finals
                  'previous_semifinals','current_fifa_rank','past_performar
group_members = pd.pivot_table(df, index = ['group', 'team'], values=columr
group_members
```

Out[6]:

group	team	current_fifa_rank	past_performance	previous_appearances	previous_finals	prev
A	Egypt	31	2	2	0	
	Russia	65	14	10	0	
	Saudi Arabia	63	4	4	0	
	Uruguay	21	80	12	2	
B	IRAN	32	4	4	0	
	Morocco	40	4	4	0	
	Porugal	3	14	6	0	
	Spain	6	46	14	1	
C	Australia	39	4	4	0	
	Denmark	12	4	4	0	
	France	9	66	14	2	
	Peru	11	4	4	0	
D	Argentina	4	108	16	5	
	Croatia	17	8	4	0	
	Iceland	22	0	0	0	
	Nigeria	50	5	5	0	
E	Brazil	2	200	20	7	
	Costarica	26	4	4	0	
	Serbia	37	19	11	0	
	Switzerland	8	10	10	0	
F	Germany	1	198	18	8	
	Korea	59	13	9	0	
	Mexico	16	15	15	0	
	Sweden	18	35	11	1	
G	Belgium	5	16	12	0	
	England	15	46	14	1	
	Panama	56	0	0	0	
	Tunisia	27	4	4	0	
H	Columbia	13	5	5	0	
	Japan	55	5	5	0	

group	team	current_fifa_rank	past_performance	previous_appearances	previous_finals	prev
	Poland	7	15	7	0	
	Senegal	23	1	1	0	

In [7]:
```
columns_to_show =['previous_appearances','previous_titles','previous_finals
                  'previous_semifinals','current_fifa_rank','past_performan
group_summary = pd.pivot_table(df, index = ['group'], values=columns_to_sho
group_summary
```

Out[7]:

group	current_fifa_rank	past_performance	previous_appearances	previous_finals	previous_semifina
A	45.00	25.00	7.00	0.50	1.!
B	20.25	17.00	7.00	0.25	1.(
C	17.75	19.50	6.50	0.50	1.:
D	23.25	30.25	6.25	1.25	1.!
E	18.25	58.25	11.25	1.75	3.:
F	23.50	65.25	13.25	2.25	4.!
G	25.75	16.50	7.50	0.25	0.7
H	24.50	6.50	4.50	0.00	0.!

QUESTION 4

Check for missing data. Write code to identify if there is any missing data.

In [8]:
```
### YOUR CODE HERE

row_with_missing = [row_idx for row_idx,row in df.isnull().iterrows() if Tr
if len(row_with_missing) > 0:
    print([df['team'].values[r] for r in row_with_missing])
else:
    print("There were no rows with missing data")

## missing values summary
print("\nMissing Value Summary\n{}".format("-"*35))
print(df.isnull().sum(axis = 0))
```

```
There were no rows with missing data

Missing Value Summary
-----------------------------------
team                    0
group                   0
previous_appearances    0
previous_titles         0
previous_finals         0
previous_semifinals     0
current_fifa_rank       0
past_performance        0
dtype: int64
```

Investigative Visualization

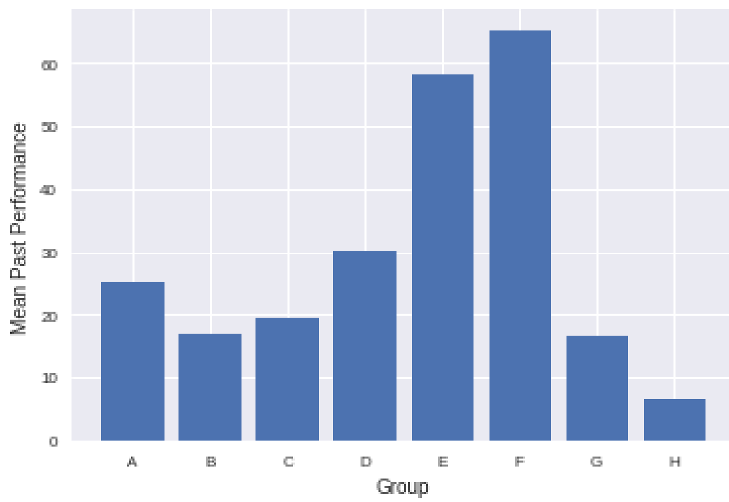
QUESTION 5

Come up with one or more plots that investigate the central question... Are the groups comprised in a fair way?

In [9]:
```
### YOUR CODE HERE

# The `group_summary` dataframe was created as part of Question 3's solutio
plt.bar(group_summary.index, group_summary['past_performance'].values)
plt.xlabel('Group')
plt.ylabel('Mean Past Performance')
```

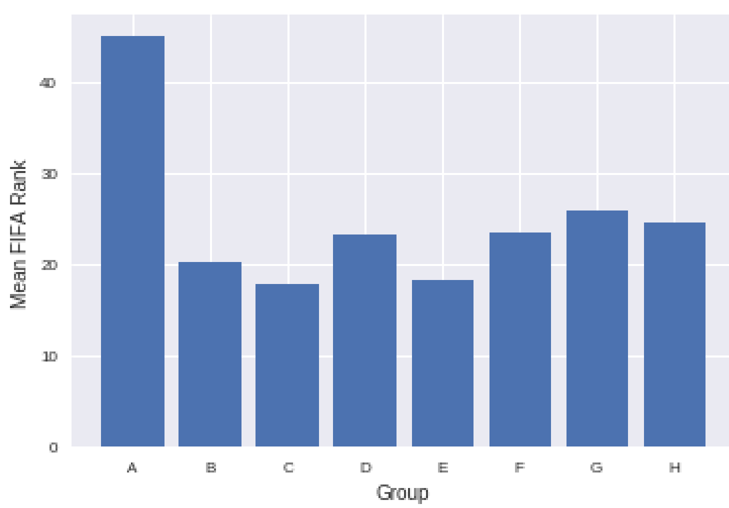
Out[9]: Text(0, 0.5, 'Mean Past Performance')



In [10]:
```
# The mean Past Performance score of teams in each group has a pretty wide
# (higher past performance scores indicate teams with more successful Worlc
# Let's compare our metric with 'current_fifa_rank', where a low number inc

plt.bar(group_summary.index, group_summary['current_fifa_rank'].values)
plt.xlabel('Group')
plt.ylabel('Mean FIFA Rank')
```

Out[10]: Text(0, 0.5, 'Mean FIFA Rank')



In [11]:
```
# In comparison the mean FIFA rank in each group seems much more tightly gr
# of Group A, which seems to be a weak group by this metric (though average
# Past Performance).

# To have a better idea if these differences are likely due to randomness c
# to apply some variant of hypothesis testing.
```

Hypothesis Testing

There are a number of ways to use hypothesis testing in this situation. There are certainly reasonable hypotheses tests and other methods like simulation approaches, that we have not discussed, but they would be appropriate here. If you choose to explore some of the methods that are outside the scope of this course then we encourage you to first try the simple approach proposed here and compare the results to any further additional approaches you choose to use.

We could use an ANOVA approach here that would signify a difference between groups, but we would not know which and how many teams were different. As we stated before there are a number of ways to approach the investigation, but lets use a simple approach. We are going to setup our investigation to look at all pairwise comparisons to provide as much insight as possible.

Recall that there are $\frac{(N-1)(N)}{2}$(N−1)(N)2 pairwise comparisons.

In [12]:
```
N = np.unique(df['group'].values).size
print("num comparisons: ",((N-1)*N) / 2.0)
```

```
num comparisons:  28.0
```

QUESTION 5

1. Choose a hypothesis test
2. State the null and alternative hypothesis, and choose a cutoff value αα
3. Run the test for all pairwise comparisons between teams. You can either loop over the different groups and use the [ttest_ind] function provided by the stats library or explore the [pairwise_ttests] function provided by the pingouin library.

In [13]:
```
### YOUR CODE HERE

# 1. Since there are only 4 teams in each group, we are well within the "sm
#    when you would use a t-test. We are comparing two samples in each t-te
#    whether the samples are *different* from each other -- meaning a two-t
#    there are separate t-tests for when the variances of the samples are e
#    As described in the NHT unit, it is generally safest to NOT assume equ

# 2. Null hypothesis: The mean FIFA rank (or past performance) within a gro
#    mean in another group in the tournament.
#    Alt. hypothesis: The mean FIFA rank (or past performance) within a gro
#    mean in another group in the tournament.
#    Set alpha = 0.1 (other values could also be reasonably chosen as well)

# 3. Write pairwise test as a function so that it can be reused for both 'p

def worldcup_pairwise_ttest(data, test_column = 'current_fifa_rank'):
    """Performs t-tests of independence pairwise between each of the 8 grou
    world cup data set. Returns a dictionary of the associated p-values."""
    pair_p_vals = {}
    grps = 'ABCDEFGH'
    for grp1_index, grp1 in enumerate(grps):
        for grp2 in grps[grp1_index+1:]:
            grp_key = '-'.join([grp1, grp2])
            grp1_data = data.loc[data.loc[:, 'group'] == grp1, test_column]
            grp2_data = data.loc[data.loc[:, 'group'] == grp2, test_column]

            pval = stats.ttest_ind(grp1_data, grp2_data, equal_var = False)
            pair_p_vals[grp_key] = pval

    return pair_p_vals

past_perf_p_vals = worldcup_pairwise_ttest(df, test_column = 'past_performa
fifa_rank_p_vals = worldcup_pairwise_ttest(df)

# Check that each dictionary has the right number of pairs:
print('past_performance pair count:', len(past_perf_p_vals))
print('current_fifa_rank pair count:', len(fifa_rank_p_vals))
```

```
past_performance pair count: 28
current_fifa_rank pair count: 28
```

In [14]:
```
# You can also use the pingouin library to do pairwise t-tests

test_results = pingouin.pairwise_ttests(data=df, dv='past_performance', bet
test_results.head()

# See the documentation of the pingouin library to learn more about this fu
# https://pingouin-stats.org/generated/pingouin.pairwise_ttests.html

# p-unc is the Uncorrected p-value for this test.
```

Out[14]:

	Contrast	A	B	Paired	Parametric	T	dof	Tail	p-unc	BF10	hedges
0	group	A	B	False	True	0.380521	4.598603	two-sided	0.720495	0.547	0.233973
1	group	A	C	False	True	0.227738	5.819394	two-sided	0.827640	0.532	0.140030
2	group	A	D	False	True	-0.164594	5.424466	two-sided	0.875223	0.528	-0.101205
3	group	A	E	False	True	-0.653967	3.896913	two-sided	0.549711	0.594	-0.402109
4	group	A	F	False	True	-0.834615	4.007803	two-sided	0.450795	0.64	-0.513184

QUESTION 6

For all of the pp-values obtained apply the Bonferroni and at least one other correction for multiple hypothesis tests. Then comment on the results.

In [15]:
```
### YOUR CODE HERE
def test_pvals_w_bonferroni(pvals_dict, alpha):
    """Applies the Bonferroni correction to the cutoff value alpha as deter
    by the number p-values contained in pvals_dict. Then tests whether thos
    p-values are at least as extreme as the cutoff. Returns a new dict with
    values. True: Reject the Null. False: Fail to reject the Null."""
    alpha_bonf = alpha / len(pvals_dict)
    return {k: v < alpha_bonf for k, v in pvals_dict.items()}

past_perf_bonf_p_vals = test_pvals_w_bonferroni(past_perf_p_vals, 0.1)
fifa_rank_bonf_p_vals = test_pvals_w_bonferroni(fifa_rank_p_vals, 0.1)

# In Python True evaluates to 1 and False evaluates to 0. So use that to co
print("Reject the null count, past_performance:", sum(past_perf_bonf_p_vals
print("Reject the null count, current_fifa_rank:", sum(fifa_rank_bonf_p_val
```

```
Reject the null count, past_performance: 0
Reject the null count, current_fifa_rank: 0
```

In [16]:
```
pingouin.pairwise_ttests(data=df, dv='past_performance', between='group', a
```

Out[16]:

	Contrast	A	B	Paired	Parametric	T	dof	Tail	p-unc	p-corr	p-adjust	BF10
0	group	A	B	False	True	0.380521	4.598603	two-sided	0.720495	1.0	bonf	0.547
1	group	A	C	False	True	0.227738	5.819394	two-sided	0.827640	1.0	bonf	0.532
2	group	A	D	False	True	-0.164594	5.424466	two-sided	0.875223	1.0	bonf	0.528
3	group	A	E	False	True	-0.653967	3.896913	two-sided	0.549711	1.0	bonf	0.594
4	group	A	F	False	True	-0.834615	4.007803	two-sided	0.450795	1.0	bonf	0.64
5	group	A	G	False	True	0.400138	4.722065	two-sided	0.706510	1.0	bonf	0.549
6	group	A	H	False	True	0.986170	3.155871	two-sided	0.393493	1.0	bonf	0.69
7	group	B	C	False	True	-0.135731	5.113540	two-sided	0.897216	1.0	bonf	0.526
8	group	B	D	False	True	-0.476447	3.862208	two-sided	0.659453	1.0	bonf	0.56
9	group	B	E	False	True	-0.852545	3.264419	two-sided	0.451908	1.0	bonf	0.646
10	group	B	F	False	True	-1.057512	3.298842	two-sided	0.361489	1.0	bonf	0.717
11	group	B	G	False	True	0.034731	5.988072	two-sided	0.973423	1.0	bonf	0.523
12	group	B	H	False	True	1.010753	3.536055	two-sided	0.376213	1.0	bonf	0.699
13	group	C	D	False	True	-0.355453	4.896751	two-sided	0.737048	1.0	bonf	0.544
14	group	C	E	False	True	-0.777756	3.635636	two-sided	0.484203	1.0	bonf	0.624
15	group	C	F	False	True	-0.970339	3.716508	two-sided	0.390704	1.0	bonf	0.684
16	group	C	G	False	True	0.160701	5.247215	two-sided	0.878334	1.0	bonf	0.527
17	group	C	H	False	True	0.823566	3.222378	two-sided	0.466729	1.0	bonf	0.637
18	group	D	E	False	True	-0.518478	4.655016	two-sided	0.627811	1.0	bonf	0.567
19	group	D	F	False	True	-0.678988	4.829188	two-sided	0.528313	1.0	bonf	0.6
20	group	D	G	False	True	0.491496	3.938894	two-sided	0.649203	1.0	bonf	0.563
21	group	D	H	False	True	0.908561	3.079316	two-sided	0.428994	1.0	bonf	0.663

	Contrast	A	B	Paired	Parametric	T	dof	Tail	p-unc	p-corr	p-adjust	BF10
22	group	E	F	False	True	-0.107695	5.977477	two-sided	0.917761	1.0	bonf	0.525
23	group	E	G	False	True	0.861179	3.289015	two-sided	0.447360	1.0	bonf	0.648
24	group	E	H	False	True	1.090748	3.023862	two-sided	0.354585	1.0	bonf	0.73
25	group	F	G	False	True	1.066104	3.326605	two-sided	0.357561	1.0	bonf	0.72
26	group	F	H	False	True	1.316442	3.026982	two-sided	0.278831	1.0	bonf	0.836
27	group	G	H	False	True	0.923843	3.490896	two-sided	0.414887	1.0	bonf	0.668

In [17]:

```
# Applying the Bonferroni correction in this case means just comparing our
# with the adjusted alpha: 0.1 / 28 = 0.00357, or equivalently multiplying e
# p-values in our dictionary by 28 and then comparing these to the original
# Since this is a simple calculation, we don't really need to use a stats l
# However for more sophisticated corrections, such as Benjamini-Hochberg, i
# convenient to use a library:

from statsmodels.stats.multitest import multipletests

# unpack dicts into lists of p-values
perf_pval_lst = list(past_perf_p_vals.values())
fifa_pval_lst = list(fifa_rank_p_vals.values())

perf_bh_tests = multipletests(perf_pval_lst, alpha = 0.1, method = 'fdr_bh'
fifa_bh_tests = multipletests(fifa_pval_lst, alpha = 0.1, method = 'fdr_bh'

# multipletests returns a tuple of items. The first item is an array of tes
print("Reject the null count, past_performance:", sum(perf_bh_tests[0]))
print("Reject the null count, current_fifa_rank:", sum(fifa_bh_tests[0]))
```

```
Reject the null count, past_performance: 0
Reject the null count, current_fifa_rank: 0
```

In [18]:
```
pingouin.pairwise_ttests(data=df, dv='past_performance', between='group', a
```

Out[18]:

	Contrast	A	B	Paired	Parametric	T	dof	Tail	p-unc	p-corr	p-adjust	Bl
0	group	A	B	False	True	0.380521	4.598603	two-sided	0.720495	0.938061	fdr_bh	0.
1	group	A	C	False	True	0.227738	5.819394	two-sided	0.827640	0.951753	fdr_bh	0.
2	group	A	D	False	True	-0.164594	5.424466	two-sided	0.875223	0.951753	fdr_bh	0.
3	group	A	E	False	True	-0.653967	3.896913	two-sided	0.549711	0.938061	fdr_bh	0.
4	group	A	F	False	True	-0.834615	4.007803	two-sided	0.450795	0.938061	fdr_bh	0
5	group	A	G	False	True	0.400138	4.722065	two-sided	0.706510	0.938061	fdr_bh	0.
6	group	A	H	False	True	0.986170	3.155871	two-sided	0.393493	0.938061	fdr_bh	0
7	group	B	C	False	True	-0.135731	5.113540	two-sided	0.897216	0.951753	fdr_bh	0.
8	group	B	D	False	True	-0.476447	3.862208	two-sided	0.659453	0.938061	fdr_bh	0
9	group	B	E	False	True	-0.852545	3.264419	two-sided	0.451908	0.938061	fdr_bh	0.
10	group	B	F	False	True	-1.057512	3.298842	two-sided	0.361489	0.938061	fdr_bh	0.
11	group	B	G	False	True	0.034731	5.988072	two-sided	0.973423	0.973423	fdr_bh	0.
12	group	B	H	False	True	1.010753	3.536055	two-sided	0.376213	0.938061	fdr_bh	0.
13	group	C	D	False	True	-0.355453	4.896751	two-sided	0.737048	0.938061	fdr_bh	0.
14	group	C	E	False	True	-0.777756	3.635636	two-sided	0.484203	0.938061	fdr_bh	0.
15	group	C	F	False	True	-0.970339	3.716508	two-sided	0.390704	0.938061	fdr_bh	0.
16	group	C	G	False	True	0.160701	5.247215	two-sided	0.878334	0.951753	fdr_bh	0.
17	group	C	H	False	True	0.823566	3.222378	two-sided	0.466729	0.938061	fdr_bh	0.
18	group	D	E	False	True	-0.518478	4.655016	two-sided	0.627811	0.938061	fdr_bh	0.
19	group	D	F	False	True	-0.678988	4.829188	two-sided	0.528313	0.938061	fdr_bh	
20	group	D	G	False	True	0.491496	3.938894	two-sided	0.649203	0.938061	fdr_bh	0.
21	group	D	H	False	True	0.908561	3.079316	two-sided	0.428994	0.938061	fdr_bh	0.

	Contrast	A	B	Paired	Parametric	T	dof	Tail	p-unc	p-corr	p-adjust	BI
22	group	E	F	False	True	-0.107695	5.977477	two-sided	0.917761	0.951753	fdr_bh	0.!
23	group	E	G	False	True	0.861179	3.289015	two-sided	0.447360	0.938061	fdr_bh	0.(
24	group	E	H	False	True	1.090748	3.023862	two-sided	0.354585	0.938061	fdr_bh	0
25	group	F	G	False	True	1.066104	3.326605	two-sided	0.357561	0.938061	fdr_bh	0
26	group	F	H	False	True	1.316442	3.026982	two-sided	0.278831	0.938061	fdr_bh	0.!
27	group	G	H	False	True	0.923843	3.490896	two-sided	0.414887	0.938061	fdr_bh	0.(

In [19]:

```
# The full contents returned from multipletests (and a similar tuple for
# for past performance):
fifa_bh_tests
```

Out[19]:

```
(array([False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False]),
 array([0.98788607, 0.98788607, 0.98788607, 0.98788607, 0.98788607,
        0.98788607, 0.98788607, 0.98788607, 0.98788607, 0.98788607,
        0.98788607, 0.98788607, 0.98788607, 0.98788607, 0.98788607,
        0.98788607, 0.98788607, 0.98788607, 0.98788607, 0.98788607,
        0.98788607, 0.98788607, 0.98788607, 0.98788607, 0.98788607,
        0.98788607, 0.98788607, 0.98788607]),
 0.0037558048145287515,
 0.0035714285714285718)
```

In [20]:

```
# This tuple contains: Test results, corrected p-values, corrected alpha
# using Sidak correction, corrected alpha using Bonferroni correction.
```

In [21]:

```
# Let's unpack these results. When comparing the strength of teams between
# our criteria of having statistically significantly different means in ter
# Past Performance scores. Which is good. This is the result we would expec
# up fairly.
```

In [22]:
```
# EXTRA:
# When we plotted the FIFA Rankings between groups we noted that Group A's
# The groups with the lowest mean with this metric (meaning they consist of
# Group E. Even though we have determined that the difference between Group
# meet our threshold for rejecting the Null, the difference is still somewh
# and examining these values is a reasonable thing to do.

df.loc[df['group'].isin(list('ACE')), ['team', 'group', 'current_fifa_rank'
```

Out[22]:

	team	group	current_fifa_rank
0	Russia	A	65
1	Saudi Arabia	A	63
2	Egypt	A	31
3	Uruguay	A	21
8	France	C	9
9	Australia	C	39
10	Peru	C	11
11	Denmark	C	12
16	Brazil	E	2
17	Switzerland	E	8
18	Costarica	E	26
19	Serbia	E	37

In [23]:
```
# One thing that stands out in these data is that Group A has 2 teams with
# while the worst ranked teams in groups C and E are in the high 30s.

# For the curious, Wikipedia sheds some light on this somewhat surprising s
# https://en.wikipedia.org/wiki/2018_FIFA_World_Cup_seeding
# According to Wikipedia: "The hosts [were] placed in Pot 1 and treated as
# and therefore Pot 1 consisted of hosts Russia and the seven highest-ranke
# qualify for the tournament."

# The host nation, Russia, got special treatment in the grouping method, wh
# based on their FIFA rank at the time. -- This treatment showed up in our
```

Additional Approaches

There is an allpairtest function in statsmodelsthat could be used here to combine the work from QUESTION 5 and QUESTION 6.

Generalized Linear Models (GLMs) are an appropriate tool to use here if we wanted to include the results of the tournament (maybe a ratio of wins/losses weighted by the final position in the tournament). `statsmodels` supports R-style formulas to fit generalized linear models. One additional variant of GLMs are hierarchical or multilevel models that provide even more insight into this types of dataset. See the tutorial on multilevel modeling.