# BasicStatisticalTesting

July 21, 2021

In this lecture we're going to review some of the basics of statistical testing in python. We're going to talk about hypothesis testing, statistical significance, and using scipy to run student's t-tests.

[1]:
```
# We use statistics in a lot of different ways in data science, and on this␣
 ↪lecture, I want to refresh your
# knowledge of hypothesis testing, which is a core data analysis activity␣
 ↪behind experimentation. The goal of
# hypothesis testing is to determine if, for instance, the two different␣
 ↪conditions we have in an experiment
# have resulted in different impacts

# Let's import our usual numpy and pandas libraries
import numpy as np
import pandas as pd

# Now let's bring in some new libraries from scipy
from scipy import stats
```

[2]:
```
# Now, scipy is an interesting collection of libraries for data science and␣
 ↪you'll use most or perpahs all of
# these libraries. It includes numpy and pandas, but also plotting libraries␣
 ↪such as matplotlib, and a
# number of scientific library functions as well
```

[3]:
```
# When we do hypothesis testing, we actually have two statements of interest:␣
 ↪the first is our actual
# explanation, which we call the alternative hypothesis, and the second is that␣
 ↪the explanation we have is not
# sufficient, and we call this the null hypothesis. Our actual testing method␣
 ↪is to determine whether the null
# hypothesis is true or not. If we find that there is a difference between␣
 ↪groups, then we can reject the null
# hypothesis and we accept our alternative.

# Let's see an example of this; we're going to use some grade data
df=pd.read_csv ('datasets/grades.csv')
df.head()
```

[3]:
```
                              student_id  assignment1_grade  \
0  B73F2C11-70F0-E37D-8B10-1D20AFED50B1          92.733946
1  98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1          86.790821
2  D0F62040-CEB0-904C-F563-2F8620916C4E          85.512541
3  FFDF2B2C-F514-EF7F-6538-A6A53518E9DC          86.030665
4  5ECBEEB6-F1CE-80AE-3164-E45E99473FB4          64.813800

           assignment1_submission  assignment2_grade  \
0  2015-11-02 06:55:34.282000000          83.030552
1  2015-11-29 14:57:44.429000000          86.290821
2  2016-01-09 05:36:02.389000000          85.512541
3  2016-04-30 06:50:39.801000000          68.824532
4  2015-12-13 17:06:10.750000000          51.491040

           assignment2_submission  assignment3_grade  \
0  2015-11-09 02:22:58.938000000          67.164441
1  2015-12-06 17:41:18.449000000          69.772657
2  2016-01-09 06:39:44.416000000          68.410033
3  2016-04-30 17:20:38.727000000          61.942079
4  2015-12-14 12:25:12.056000000          41.932832

           assignment3_submission  assignment4_grade  \
0  2015-11-12 08:58:33.998000000          53.011553
1  2015-12-10 08:54:55.904000000          55.098125
2  2016-01-15 20:22:45.882000000          54.728026
3  2016-05-12 07:47:16.326000000          49.553663
4  2015-12-29 14:25:22.594000000          36.929549

           assignment4_submission  assignment5_grade  \
0  2015-11-16 01:21:24.663000000          47.710398
1  2015-12-13 17:32:30.941000000          49.588313
2  2016-01-11 12:41:50.749000000          49.255224
3  2016-05-07 16:09:20.485000000          49.553663
4  2015-12-28 01:29:55.901000000          33.236594

           assignment5_submission  assignment6_grade  \
0  2015-11-20 13:24:59.692000000          38.168318
1  2015-12-19 23:26:39.285000000          44.629482
2  2016-01-11 17:31:12.489000000          44.329701
3  2016-05-24 12:51:18.016000000          44.598297
4  2015-12-29 14:46:06.628000000          33.236594

           assignment6_submission
0  2015-11-22 18:31:15.934000000
1  2015-12-21 17:07:24.275000000
2  2016-01-17 16:24:42.765000000
3  2016-05-26 08:09:12.058000000
```

```
4  2016-01-05 01:06:59.546000000
```

[4]:
```
# If we take a look at the data frame inside, we see we have six different␣
 ↪assignments. Lets look at some
# summary statistics for this DataFrame
print("There are {} rows and {} columns".format(df.shape[0], df.shape[1]))
```

```
There are 2315 rows and 13 columns
```

[5]:
```
# For the purpose of this lecture, let's segment this population into two␣
 ↪pieces. Let's say those who finish
# the first assignment by the end of December 2015, we'll call them early␣
 ↪finishers, and those who finish it
# sometime after that, we'll call them late finishers.

early_finishers=df[pd.to_datetime(df['assignment1_submission']) < '2016']
early_finishers.head()
```

[5]:
```
                             student_id  assignment1_grade  \
0  B73F2C11-70F0-E37D-8B10-1D20AFED50B1          92.733946
1  98A0FAE0-A19A-13D2-4BB5-CFBFD94031D1          86.790821
4  5ECBEEB6-F1CE-80AE-3164-E45E99473FB4          64.813800
5  D09000A0-827B-C0FF-3433-BF8FF286E15B          71.647278
8  C9D51293-BD58-F113-4167-A7C0BAFCB6E5          66.595568

          assignment1_submission  assignment2_grade  \
0  2015-11-02 06:55:34.282000000          83.030552
1  2015-11-29 14:57:44.429000000          86.290821
4  2015-12-13 17:06:10.750000000          51.491040
5  2015-12-28 04:35:32.836000000          64.052550
8  2015-12-25 02:29:28.415000000          52.916454

          assignment2_submission  assignment3_grade  \
0  2015-11-09 02:22:58.938000000          67.164441
1  2015-12-06 17:41:18.449000000          69.772657
4  2015-12-14 12:25:12.056000000          41.932832
5  2016-01-03 21:05:38.392000000          64.752550
8  2015-12-31 01:42:30.046000000          48.344809

          assignment3_submission  assignment4_grade  \
0  2015-11-12 08:58:33.998000000          53.011553
1  2015-12-10 08:54:55.904000000          55.098125
4  2015-12-29 14:25:22.594000000          36.929549
5  2016-01-07 08:55:43.692000000          57.467295
8  2016-01-05 23:34:02.180000000          47.444809

          assignment4_submission  assignment5_grade  \
0  2015-11-16 01:21:24.663000000          47.710398
```

```
1  2015-12-13 17:32:30.941000000          49.588313
4  2015-12-28 01:29:55.901000000          33.236594
5  2016-01-11 00:45:28.706000000          57.467295
8  2016-01-02 07:48:42.517000000          37.955847

          assignment5_submission  assignment6_grade  \
0  2015-11-20 13:24:59.692000000          38.168318
1  2015-12-19 23:26:39.285000000          44.629482
4  2015-12-29 14:46:06.628000000          33.236594
5  2016-01-11 00:54:13.579000000          57.467295
8  2016-01-03 21:27:04.266000000          37.955847

          assignment6_submission
0  2015-11-22 18:31:15.934000000
1  2015-12-21 17:07:24.275000000
4  2016-01-05 01:06:59.546000000
5  2016-01-20 19:54:46.166000000
8  2016-01-19 15:24:31.060000000
```

[6]:
```
# So, you have lots of skills now with pandas, how would you go about getting␣
 ↪the late_finishers dataframe?
# Why don't you pause the video and give it a try.
```

[7]:
```
# Here's my solution. First, the dataframe df and the early_finishers share␣
 ↪index values, so I really just
# want everything in the df which is not in early_finishers
late_finishers=df[~df.index.isin(early_finishers.index)]
late_finishers.head()
```

[7]:
```
                             student_id  assignment1_grade  \
2  D0F62040-CEB0-904C-F563-2F8620916C4E          85.512541
3  FFDF2B2C-F514-EF7F-6538-A6A53518E9DC          86.030665
6  3217BE3F-E4B0-C3B6-9F64-462456819CE4          87.498744
7  F1CB5AA1-B3DE-5460-FAFF-BE951FD38B5F          80.576090
9  E2C617C2-4654-622C-AB50-1550C4BE42A0          59.270882

          assignment1_submission  assignment2_grade  \
2  2016-01-09 05:36:02.389000000          85.512541
3  2016-04-30 06:50:39.801000000          68.824532
6  2016-03-05 11:05:25.408000000          69.998995
7  2016-01-24 18:24:25.619000000          72.518481
9  2016-03-06 12:06:26.185000000          59.270882

          assignment2_submission  assignment3_grade  \
2  2016-01-09 06:39:44.416000000          68.410033
3  2016-04-30 17:20:38.727000000          61.942079
6  2016-03-09 07:29:52.405000000          55.999196
7  2016-01-27 13:37:12.943000000          65.266633
```

```
9  2016-03-13 02:07:25.289000000          53.343794

          assignment3_submission  assignment4_grade  \
2  2016-01-15 20:22:45.882000000          54.728026
3  2016-05-12 07:47:16.326000000          49.553663
6  2016-03-16 22:31:24.316000000          50.399276
7  2016-01-30 14:34:36.581000000          65.266633
9  2016-03-17 07:30:09.241000000          53.343794

          assignment4_submission  assignment5_grade  \
2  2016-01-11 12:41:50.749000000          49.255224
3  2016-05-07 16:09:20.485000000          49.553663
6  2016-03-18 07:19:26.032000000          45.359349
7  2016-02-03 22:08:49.002000000          65.266633
9  2016-03-20 21:45:56.229000000          42.675035

          assignment5_submission  assignment6_grade  \
2  2016-01-11 17:31:12.489000000          44.329701
3  2016-05-24 12:51:18.016000000          44.598297
6  2016-03-19 10:35:41.869000000          45.359349
7  2016-02-16 14:22:23.664000000          65.266633
9  2016-03-27 15:55:04.414000000          38.407532

          assignment6_submission
2  2016-01-17 16:24:42.765000000
3  2016-05-26 08:09:12.058000000
6  2016-03-23 14:02:00.987000000
7  2016-02-18 08:35:04.796000000
9  2016-03-30 20:33:13.554000000
```

[8]:
```
# There are lots of other ways to do this. For instance, you could just copy␣
 ↪and paste the first projection
# and change the sign from less than to greater than or equal to. This is ok,␣
 ↪but if you decide you want to
# change the date down the road you have to remember to change it in two places.
 ↪ You could also do a join of
# the dataframe df with early_finishers - if you do a left join you only keep␣
 ↪the items in the left dataframe,
# so this would have been a good answer. You also could have written a function␣
 ↪that determines if someone is
# early or late, and then called .apply() on the dataframe and added a new␣
 ↪column to the dataframe. This is a
# pretty reasonable answer as well.
```

[9]:
```
# As you've seen, the pandas data frame object has a variety of statistical␣
 ↪functions associated with it. If
```

```
# we call the mean function directly on the data frame, we see that each of the␣
 ↪means for the assignments are
# calculated. Let's compare the means for our two populations

print(early_finishers['assignment1_grade'].mean())
print(late_finishers['assignment1_grade'].mean())
```

```
74.94728457024303
74.0450648477065
```

[10]:
```
# Ok, these look pretty similar. But, are they the same? What do we mean by␣
 ↪similar? This is where the
# students' t-test comes in. It allows us to form the alternative hypothesis␣
 ↪("These are different") as well
# as the null hypothesis ("These are the same") and then test that null␣
 ↪hypothesis.

# When doing hypothesis testing, we have to choose a significance level as a␣
 ↪threshold for how much of a
# chance we're willing to accept. This significance level is typically called␣
 ↪alpha. #For this example, let's
# use a threshold of 0.05 for our alpha or 5%. Now this is a commonly used␣
 ↪number but it's really quite
# arbitrary.

# The SciPy library contains a number of different statistical tests and forms␣
 ↪a basis for hypothesis testing
# in Python and we're going to use the ttest_ind() function which does an␣
 ↪independent t-test (meaning the
# populations are not related to one another). The result of ttest_index() are␣
 ↪the t-statistic and a p-value.
# It's this latter value, the probability, which is most important to us, as it␣
 ↪indicates the chance (between
# 0 and 1) of our null hypothesis being True.

# Let's bring in our ttest_ind function
from scipy.stats import ttest_ind

# Let's run this function with our two populations, looking at the assignment 1␣
 ↪grades
ttest_ind(early_finishers['assignment1_grade'],␣
 ↪late_finishers['assignment1_grade'])
```

[10]:
```
Ttest_indResult(statistic=1.322354085372139, pvalue=0.1861810110171455)
```

[11]:
```
# So here we see that the probability is 0.18, and this is above our alpha␣
 ↪value of 0.05. This means that we
```

```
# cannot reject the null hypothesis. The null hypothesis was that the two␣
 ↪populations are the same, and we
# don't have enough certainty in our evidence (because it is greater than␣
 ↪alpha) to come to a conclusion to
# the contrary. This doesn't mean that we have proven the populations are the␣
 ↪same.
```

[12]:
```
# Why don't we check the other assignment grades?
print(ttest_ind(early_finishers['assignment2_grade'],␣
 ↪late_finishers['assignment2_grade']))
print(ttest_ind(early_finishers['assignment3_grade'],␣
 ↪late_finishers['assignment3_grade']))
print(ttest_ind(early_finishers['assignment4_grade'],␣
 ↪late_finishers['assignment4_grade']))
print(ttest_ind(early_finishers['assignment5_grade'],␣
 ↪late_finishers['assignment5_grade']))
print(ttest_ind(early_finishers['assignment6_grade'],␣
 ↪late_finishers['assignment6_grade']))
```

```
Ttest_indResult(statistic=1.2514717608216366, pvalue=0.2108889627004424)
Ttest_indResult(statistic=1.6133726558705392, pvalue=0.10679998102227865)
Ttest_indResult(statistic=0.049671157386456125, pvalue=0.960388729789337)
Ttest_indResult(statistic=-0.05279315545404755, pvalue=0.9579012739746492)
Ttest_indResult(statistic=-0.11609743352612056, pvalue=0.9075854011989656)
```

[13]:
```
# Ok, so it looks like in this data we do not have enough evidence to suggest␣
 ↪the populations differ with
# respect to grade. Let's take a look at those p-values for a moment though,␣
 ↪because they are saying things
# that can inform experimental design down the road. For instance, one of the␣
 ↪assignments, assignment 3, has a
# p-value around 0.1. This means that if we accepted a level of chance␣
 ↪similarity of 11% this would have been
# considered statistically significant. As a research, this would suggest to me␣
 ↪that there is something here
# worth considering following up on. For instance, if we had a small number of␣
 ↪participants (we don't) or if
# there was something unique about this assignment as it relates to our␣
 ↪experiment (whatever it was) then
# there may be followup experiments we could run.
```

[14]:
```
# P-values have come under fire recently for being insuficient for telling us␣
 ↪enough about the interactions
# which are happening, and two other techniques, confidence intervalues and␣
 ↪bayesian analyses, are being used
# more regularly. One issue with p-values is that as you run more tests you are␣
 ↪likely to get a value which
```

```
# is statistically significant just by chance.

# Lets see a simulation of this. First, lets create a data frame of 100␣
 ↪columns, each with 100 numbers
df1=pd.DataFrame([np.random.random(100) for x in range(100)])
df1.head()
```

[14]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.249058 | 0.748065 | 0.245871 | 0.164790 | 0.321661 | 0.355034 | 0.882641 | |
| 1 | 0.731003 | 0.592047 | 0.488109 | 0.969236 | 0.873551 | 0.976036 | 0.296337 | |
| 2 | 0.315190 | 0.081075 | 0.326315 | 0.166250 | 0.257618 | 0.406736 | 0.996209 | |
| 3 | 0.512193 | 0.639647 | 0.241200 | 0.819644 | 0.439495 | 0.111223 | 0.982239 | |
| 4 | 0.506768 | 0.799246 | 0.233428 | 0.849028 | 0.682260 | 0.904570 | 0.962683 | |

| | 7 | 8 | 9 | ... | 90 | 91 | 92 | 93 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.304377 | 0.409110 | 0.080049 | ... | 0.900937 | 0.740939 | 0.353839 | 0.928663 | |
| 1 | 0.398915 | 0.258550 | 0.925547 | ... | 0.478983 | 0.242443 | 0.225005 | 0.803914 | |
| 2 | 0.986399 | 0.339623 | 0.828867 | ... | 0.157851 | 0.496227 | 0.983652 | 0.971251 | |
| 3 | 0.574194 | 0.935731 | 0.609363 | ... | 0.373206 | 0.271515 | 0.942992 | 0.701573 | |
| 4 | 0.734952 | 0.933580 | 0.904864 | ... | 0.263317 | 0.895435 | 0.327082 | 0.872009 | |

| | 94 | 95 | 96 | 97 | 98 | 99 |
|---|---|---|---|---|---|---|
| 0 | 0.011866 | 0.876878 | 0.609879 | 0.721572 | 0.589627 | 0.554882 |
| 1 | 0.576076 | 0.190479 | 0.696137 | 0.573191 | 0.505417 | 0.626049 |
| 2 | 0.986353 | 0.547430 | 0.255769 | 0.165918 | 0.242719 | 0.575260 |
| 3 | 0.039384 | 0.085479 | 0.284002 | 0.509064 | 0.112163 | 0.334849 |
| 4 | 0.566034 | 0.474804 | 0.258897 | 0.316417 | 0.433785 | 0.127601 |

[5 rows x 100 columns]

[15]:
```
# Pause this and reflect -- do you understand the list comprehension and how I␣
 ↪created this DataFrame? You
# don't have to use a list comprehension to do this, but you should be able to␣
 ↪read this and figure out how it
# works as this is a commonly used approach on web forums.
```

[16]:
```
# Ok, let's create a second dataframe
df2=pd.DataFrame([np.random.random(100) for x in range(100)])
```

[17]:
```
# Are these two DataFrames the same? Maybe a better question is, for a given␣
 ↪row inside of df1, is it the same
# as the row inside df2?

# Let's take a look. Let's say our critical value is 0.1, or and alpha of 10%.␣
 ↪And we're going to compare each
# column in df1 to the same numbered column in df2. And we'll report when the␣
 ↪p-value isn't less than 10%,
```

```
# which means that we have sufficient evidence to say that the columns are␣
 ↪different.

# Let's write this in a function called test_columns
def test_columns(alpha=0.1):
    # I want to keep track of how many differ
    num_diff=0
    # And now we can just iterate over the columns
    for col in df1.columns:
        # we can run out ttest_ind between the two dataframes
        teststat,pval=ttest_ind(df1[col],df2[col])
        # and we check the pvalue versus the alpha
        if pval<=alpha:
            # And now we'll just print out if they are different and increment␣
 ↪the num_diff
            print("Col {} is statistically significantly different at alpha={},␣
 ↪pval={}".format(col,alpha,pval))
            num_diff=num_diff+1
    # and let's print out some summary stats
    print("Total number different was {}, which is {}%".
 ↪format(num_diff,float(num_diff)/len(df1.columns)*100))

# And now lets actually run this
test_columns()
```

```
Col 4 is statistically significantly different at alpha=0.1,
pval=0.03171904508748956
Col 9 is statistically significantly different at alpha=0.1,
pval=0.05535264960034999
Col 13 is statistically significantly different at alpha=0.1,
pval=0.06411026546897207
Col 25 is statistically significantly different at alpha=0.1,
pval=0.06474837389971169
Col 46 is statistically significantly different at alpha=0.1,
pval=0.0766378775191977
Col 73 is statistically significantly different at alpha=0.1,
pval=0.021821747279110435
Col 76 is statistically significantly different at alpha=0.1,
pval=0.04067259983835964
Col 85 is statistically significantly different at alpha=0.1,
pval=0.0029530262143926777
Col 86 is statistically significantly different at alpha=0.1,
pval=0.010704932840894129
Total number different was 9, which is 9.0%
```

[18]:
```
# Interesting, so we see that there are a bunch of columns that are different!␣
 ↪In fact, that number looks a
```

```
# lot like the alpha value we chose. So what's going on - shouldn't all of the␣
 ↪columns be the same? Remember
# that all the ttest does is check if two sets are similar given some level of␣
 ↪confidence, in our case, 10%.
# The more random comparisons you do, the more will just happen to be the same␣
 ↪by chance. In this example, we
# checked 100 columns, so we would expect there to be roughly 10 of them if our␣
 ↪alpha was 0.1.

# We can test some other alpha values as well
test_columns(0.05)
```

```
Col 4 is statistically significantly different at alpha=0.05,
pval=0.03171904508748956
Col 73 is statistically significantly different at alpha=0.05,
pval=0.021821747279110435
Col 76 is statistically significantly different at alpha=0.05,
pval=0.04067259983835964
Col 85 is statistically significantly different at alpha=0.05,
pval=0.0029530262143926777
Col 86 is statistically significantly different at alpha=0.05,
pval=0.010704932840894129
Total number different was 5, which is 5.0%
```

[19]:
```
# So, keep this in mind when you are doing statistical tests like the t-test␣
 ↪which has a p-value. Understand
# that this p-value isn't magic, that it's a threshold for you when reporting␣
 ↪results and trying to answer
# your hypothesis. What's a reasonable threshold? Depends on your question, and␣
 ↪you need to engage domain
# experts to better understand what they would consider significant.

# Just for fun, lets recreate that second dataframe using a non-normal␣
 ↪distribution, I'll arbitrarily chose
# chi squared
df2=pd.DataFrame([np.random.chisquare(df=1,size=100) for x in range(100)])
test_columns()
```

```
Col 0 is statistically significantly different at alpha=0.1,
pval=0.0006359131132062404
Col 1 is statistically significantly different at alpha=0.1,
pval=1.6254596067473314e-05
Col 2 is statistically significantly different at alpha=0.1,
pval=0.032780760709076824
Col 3 is statistically significantly different at alpha=0.1,
pval=0.0032178974546995965
```

```
Col 4 is statistically significantly different at alpha=0.1,
pval=0.0433143837061582
Col 5 is statistically significantly different at alpha=0.1,
pval=7.931565914508593e-05
Col 6 is statistically significantly different at alpha=0.1,
pval=0.0017013488226864455
Col 7 is statistically significantly different at alpha=0.1,
pval=0.00035664966978204754
Col 8 is statistically significantly different at alpha=0.1,
pval=0.0008750262277182237
Col 9 is statistically significantly different at alpha=0.1,
pval=3.9005870973813224e-05
Col 10 is statistically significantly different at alpha=0.1,
pval=0.0001340016519264235
Col 11 is statistically significantly different at alpha=0.1,
pval=2.9387732963990273e-06
Col 12 is statistically significantly different at alpha=0.1,
pval=0.0002593817469470156
Col 13 is statistically significantly different at alpha=0.1,
pval=0.0001639697026470583
Col 14 is statistically significantly different at alpha=0.1,
pval=0.00015526373271627754
Col 15 is statistically significantly different at alpha=0.1,
pval=4.1837854872688554e-05
Col 16 is statistically significantly different at alpha=0.1,
pval=0.00139904683631816
Col 17 is statistically significantly different at alpha=0.1,
pval=0.008340935008253801
Col 18 is statistically significantly different at alpha=0.1,
pval=0.0005255784024108116
Col 19 is statistically significantly different at alpha=0.1,
pval=0.0010508383648962462
Col 20 is statistically significantly different at alpha=0.1,
pval=9.250384193688497e-05
Col 21 is statistically significantly different at alpha=0.1,
pval=0.0004139078346825869
Col 22 is statistically significantly different at alpha=0.1,
pval=0.000239886090894122
Col 23 is statistically significantly different at alpha=0.1,
pval=0.005114973931480402
Col 24 is statistically significantly different at alpha=0.1,
pval=3.9612287309026586e-06
Col 25 is statistically significantly different at alpha=0.1,
pval=0.001183662110645545
Col 26 is statistically significantly different at alpha=0.1,
pval=0.0013250467524673893
Col 27 is statistically significantly different at alpha=0.1,
pval=0.0071413662391291675
```

```
Col 28 is statistically significantly different at alpha=0.1,
pval=0.0018050525585905151
Col 29 is statistically significantly different at alpha=0.1,
pval=1.8443168650897388e-07
Col 30 is statistically significantly different at alpha=0.1,
pval=0.00018610780116814143
Col 31 is statistically significantly different at alpha=0.1,
pval=0.003288231435508567
Col 32 is statistically significantly different at alpha=0.1,
pval=0.00022788735453261414
Col 33 is statistically significantly different at alpha=0.1,
pval=0.0005420802253194416
Col 34 is statistically significantly different at alpha=0.1,
pval=0.00043337835781774326
Col 35 is statistically significantly different at alpha=0.1,
pval=0.0009296626942443332
Col 36 is statistically significantly different at alpha=0.1,
pval=0.015486530171592511
Col 37 is statistically significantly different at alpha=0.1,
pval=0.08968808146980235
Col 38 is statistically significantly different at alpha=0.1,
pval=0.0003984738782135047
Col 39 is statistically significantly different at alpha=0.1,
pval=0.00029812833313821313
Col 40 is statistically significantly different at alpha=0.1,
pval=0.00020580570179961344
Col 41 is statistically significantly different at alpha=0.1,
pval=0.016524830474479658
Col 42 is statistically significantly different at alpha=0.1,
pval=0.004414785565347055
Col 43 is statistically significantly different at alpha=0.1,
pval=0.0006514221896553882
Col 44 is statistically significantly different at alpha=0.1,
pval=0.0009246080966774699
Col 45 is statistically significantly different at alpha=0.1,
pval=3.947719385039122e-05
Col 46 is statistically significantly different at alpha=0.1,
pval=0.003936803652126271
Col 47 is statistically significantly different at alpha=0.1,
pval=0.00021471514569468722
Col 48 is statistically significantly different at alpha=0.1,
pval=0.00021533428873488688
Col 49 is statistically significantly different at alpha=0.1,
pval=0.08198381088074698
Col 50 is statistically significantly different at alpha=0.1,
pval=0.0005833899282268582
Col 51 is statistically significantly different at alpha=0.1,
pval=0.0008976227172661998
```

```
Col 52 is statistically significantly different at alpha=0.1,
pval=0.002183481627605226
Col 53 is statistically significantly different at alpha=0.1,
pval=0.00011517331466172489
Col 54 is statistically significantly different at alpha=0.1,
pval=0.0004539298416324909
Col 55 is statistically significantly different at alpha=0.1,
pval=0.0005972354479418133
Col 56 is statistically significantly different at alpha=0.1,
pval=0.00039537371324602574
Col 57 is statistically significantly different at alpha=0.1,
pval=1.5992684064779836e-05
Col 58 is statistically significantly different at alpha=0.1,
pval=0.0035581119921681737
Col 59 is statistically significantly different at alpha=0.1,
pval=0.0009069465410284678
Col 60 is statistically significantly different at alpha=0.1,
pval=0.001490617627902622
Col 61 is statistically significantly different at alpha=0.1,
pval=9.982665694645061e-05
Col 62 is statistically significantly different at alpha=0.1,
pval=0.0001855201502830867
Col 63 is statistically significantly different at alpha=0.1,
pval=0.0717892199552874
Col 64 is statistically significantly different at alpha=0.1,
pval=0.00011111316914952978
Col 65 is statistically significantly different at alpha=0.1,
pval=0.0007326247241706332
Col 66 is statistically significantly different at alpha=0.1,
pval=0.0006985304647035251
Col 67 is statistically significantly different at alpha=0.1,
pval=4.043055942025079e-05
Col 68 is statistically significantly different at alpha=0.1,
pval=0.0030620341209672865
Col 69 is statistically significantly different at alpha=0.1,
pval=0.0002566485047083988
Col 70 is statistically significantly different at alpha=0.1,
pval=1.5265122338601147e-05
Col 71 is statistically significantly different at alpha=0.1,
pval=0.0025159633542264875
Col 72 is statistically significantly different at alpha=0.1,
pval=0.006683482780335058
Col 73 is statistically significantly different at alpha=0.1,
pval=0.0004996694763506213
Col 74 is statistically significantly different at alpha=0.1,
pval=0.00024394144210986108
Col 75 is statistically significantly different at alpha=0.1,
pval=7.077435724052419e-05
```

```
Col 76 is statistically significantly different at alpha=0.1,
pval=0.0074093508129584405
Col 77 is statistically significantly different at alpha=0.1,
pval=0.006900999886679228
Col 78 is statistically significantly different at alpha=0.1,
pval=0.0010484064032203217
Col 79 is statistically significantly different at alpha=0.1,
pval=0.0023846343634806146
Col 80 is statistically significantly different at alpha=0.1,
pval=8.180792170754975e-05
Col 81 is statistically significantly different at alpha=0.1,
pval=5.49471394287242e-05
Col 82 is statistically significantly different at alpha=0.1,
pval=0.00033420295079797965
Col 83 is statistically significantly different at alpha=0.1,
pval=0.003652590090290096
Col 84 is statistically significantly different at alpha=0.1,
pval=0.0005087442921717129
Col 85 is statistically significantly different at alpha=0.1,
pval=0.010877718451872447
Col 86 is statistically significantly different at alpha=0.1,
pval=0.006487775025921745
Col 87 is statistically significantly different at alpha=0.1,
pval=0.00012983278069724697
Col 88 is statistically significantly different at alpha=0.1,
pval=0.0003217680289398193
Col 89 is statistically significantly different at alpha=0.1,
pval=0.0007595005579977705
Col 90 is statistically significantly different at alpha=0.1,
pval=0.0009007314132232909
Col 91 is statistically significantly different at alpha=0.1,
pval=0.0041773223441325755
Col 92 is statistically significantly different at alpha=0.1,
pval=0.00013412252503388242
Col 93 is statistically significantly different at alpha=0.1,
pval=0.0038256024065565237
Col 94 is statistically significantly different at alpha=0.1,
pval=0.0025884060593381543
Col 95 is statistically significantly different at alpha=0.1,
pval=0.0008645653623897333
Col 96 is statistically significantly different at alpha=0.1,
pval=0.00033563002443083584
Col 97 is statistically significantly different at alpha=0.1,
pval=9.000223928968477e-06
Col 98 is statistically significantly different at alpha=0.1,
pval=0.007777985999968882
Col 99 is statistically significantly different at alpha=0.1,
pval=0.00024422963411248024
```

```
Total number different was 100, which is 100.0%
```

[20]:
```
# Now we see that all or most columns test to be statistically significant at␣
↪the 10% level.
```

In this lecture, we've discussed just some of the basics of hypothesis testing in Python. I introduced you to the SciPy library, which you can use for the students t test. We've discussed some of the practical issues which arise from looking for statistical significance. There's much more to learn about hypothesis testing, for instance, there are different tests used, depending on the shape of your data and different ways to report results instead of just p-values such as confidence intervals or bayesian analyses. But this should give you a basic idea of where to start when comparing two populations for differences, which is a common task for data scientists.