BasicStatisticalTesting

July 15, 2023

1 Basic Statistical Testing

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
```

```python
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())
```

```
74.94728457024304
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=1.3223540853721596, pvalue=0.18618101101713855)

[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  0.784561  0.143093  0.994432  0.830963  0.877276  0.416970  0.750634
1  0.255321  0.047881  0.398758  0.242838  0.205288  0.091025  0.977364
2  0.123759  0.808137  0.465623  0.972418  0.863892  0.693695  0.661254
3  0.576343  0.979884  0.740361  0.388672  0.376778  0.978228  0.032805
4  0.142473  0.443957  0.172624  0.480366  0.515050  0.347695  0.692685

          7         8         9   …         90        91        92        93  \
0  0.300715  0.855492  0.370564  …   0.640426  0.385308  0.941685  0.093030
1  0.498815  0.396510  0.861702  …   0.581837  0.010701  0.165911  0.945249
2  0.463712  0.811478  0.032888  …   0.558607  0.903286  0.321288  0.966404
3  0.982540  0.214757  0.953151  …   0.699600  0.994941  0.273427  0.979412
4  0.015737  0.678974  0.843999  …   0.569366  0.366748  0.851031  0.889310

         94        95        96        97        98        99
0  0.903792  0.749085  0.071989  0.276487  0.899951  0.505959
1  0.121085  0.964968  0.940653  0.742255  0.272952  0.755611
2  0.684333  0.730186  0.930317  0.915076  0.261474  0.900475
3  0.142267  0.802328  0.104293  0.839022  0.113747  0.712073
4  0.913491  0.661496  0.728779  0.339710  0.702350  0.845452

[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]:

```python
# 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 12 is statistically significantly different at alpha=0.1,
pval=0.06767925839957789
Col 18 is statistically significantly different at alpha=0.1,
pval=0.060961953802938985
Col 27 is statistically significantly different at alpha=0.1,
pval=0.06785099795714658
Col 31 is statistically significantly different at alpha=0.1,
pval=0.008332106458114195
Col 37 is statistically significantly different at alpha=0.1,
pval=0.04513440949945866
Col 51 is statistically significantly different at alpha=0.1,
pval=0.029030061070060724
Col 58 is statistically significantly different at alpha=0.1,
pval=0.054612215179054284
```

```
Col 59 is statistically significantly different at alpha=0.1,
pval=0.08156601001865037
Col 80 is statistically significantly different at alpha=0.1,
pval=0.09064527663555055
Col 83 is statistically significantly different at alpha=0.1,
pval=0.03506928859661154
Col 99 is statistically significantly different at alpha=0.1,
pval=0.07237191304184411
Total number different was 11, which is 11.0%
```

[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 31 is statistically significantly different at alpha=0.05,
pval=0.008332106458114195
Col 37 is statistically significantly different at alpha=0.05,
pval=0.04513440949945866
Col 51 is statistically significantly different at alpha=0.05,
pval=0.029030061070060724
Col 83 is statistically significantly different at alpha=0.05,
pval=0.03506928859661154
Total number different was 4, which is 4.0%
```

[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=2.1702732819726844e-05
Col 1 is statistically significantly different at alpha=0.1,
pval=8.250540866540605e-05
Col 2 is statistically significantly different at alpha=0.1,
pval=5.05078396883184e-05
Col 3 is statistically significantly different at alpha=0.1,
pval=0.04708795425138086
Col 4 is statistically significantly different at alpha=0.1,
pval=0.0002500110110347758
Col 5 is statistically significantly different at alpha=0.1,
pval=0.0006359196885136039
Col 6 is statistically significantly different at alpha=0.1,
pval=0.0038464386970038554
Col 7 is statistically significantly different at alpha=0.1,
pval=0.0009271598405014023
Col 8 is statistically significantly different at alpha=0.1,
pval=0.003785768232109207
Col 9 is statistically significantly different at alpha=0.1,
pval=0.005111219372425077
Col 10 is statistically significantly different at alpha=0.1,
pval=0.020876689987083436
Col 11 is statistically significantly different at alpha=0.1,
pval=0.013578021998385984
Col 12 is statistically significantly different at alpha=0.1,
pval=0.0020178876891349035
Col 13 is statistically significantly different at alpha=0.1,
pval=0.002870703683097223
Col 14 is statistically significantly different at alpha=0.1,
pval=0.003670569509860025
Col 15 is statistically significantly different at alpha=0.1,
pval=3.960698984421419e-06
Col 16 is statistically significantly different at alpha=0.1,
pval=0.00037707734164105396
Col 17 is statistically significantly different at alpha=0.1,
pval=5.6961121133150486e-05
Col 18 is statistically significantly different at alpha=0.1,
pval=0.020226868409573568
Col 19 is statistically significantly different at alpha=0.1,
pval=5.060001094907332e-05
Col 20 is statistically significantly different at alpha=0.1,
pval=3.550231713588656e-05
Col 21 is statistically significantly different at alpha=0.1,
pval=0.0026044902846320732
Col 22 is statistically significantly different at alpha=0.1,
pval=0.0003622823014905168
Col 23 is statistically significantly different at alpha=0.1,
pval=1.4928515031554456e-05
```

Col 24 is statistically significantly different at alpha=0.1,
pval=0.013366551145320288
Col 25 is statistically significantly different at alpha=0.1,
pval=0.000203094771099088
Col 26 is statistically significantly different at alpha=0.1,
pval=0.0026504714863639124
Col 27 is statistically significantly different at alpha=0.1,
pval=0.006032171036893771
Col 28 is statistically significantly different at alpha=0.1,
pval=0.008285555343904415
Col 29 is statistically significantly different at alpha=0.1,
pval=9.15647302583541e-05
Col 30 is statistically significantly different at alpha=0.1,
pval=6.316069414127049e-05
Col 31 is statistically significantly different at alpha=0.1,
pval=2.2585849838335167e-07
Col 32 is statistically significantly different at alpha=0.1,
pval=0.0003603517679068996
Col 33 is statistically significantly different at alpha=0.1,
pval=0.02821439062076009
Col 34 is statistically significantly different at alpha=0.1,
pval=0.006601119535177393
Col 35 is statistically significantly different at alpha=0.1,
pval=0.04794075255549269
Col 36 is statistically significantly different at alpha=0.1,
pval=0.00014279672792659565
Col 37 is statistically significantly different at alpha=0.1,
pval=0.03369168979754231
Col 38 is statistically significantly different at alpha=0.1,
pval=6.513054934893659e-05
Col 39 is statistically significantly different at alpha=0.1,
pval=0.008307454368482467
Col 40 is statistically significantly different at alpha=0.1,
pval=0.0011696729028944153
Col 41 is statistically significantly different at alpha=0.1,
pval=0.00025915928884403866
Col 42 is statistically significantly different at alpha=0.1,
pval=4.191804573681507e-06
Col 43 is statistically significantly different at alpha=0.1,
pval=0.018910097760053122
Col 44 is statistically significantly different at alpha=0.1,
pval=0.0022844499011564425
Col 45 is statistically significantly different at alpha=0.1,
pval=3.278096594236466e-06
Col 46 is statistically significantly different at alpha=0.1,
pval=0.00038546502672858623
Col 47 is statistically significantly different at alpha=0.1,
pval=0.0009297793125300256

```
Col 48 is statistically significantly different at alpha=0.1,
pval=0.007317371152646734
Col 49 is statistically significantly different at alpha=0.1,
pval=3.866829724151193e-05
Col 50 is statistically significantly different at alpha=0.1,
pval=0.01338413428376254
Col 51 is statistically significantly different at alpha=0.1,
pval=0.0005213254289293319
Col 52 is statistically significantly different at alpha=0.1,
pval=0.00040322793311739795
Col 53 is statistically significantly different at alpha=0.1,
pval=0.0033593501926800634
Col 54 is statistically significantly different at alpha=0.1,
pval=0.00019766158387444772
Col 55 is statistically significantly different at alpha=0.1,
pval=0.005785338804988192
Col 56 is statistically significantly different at alpha=0.1,
pval=5.024278076272807e-06
Col 57 is statistically significantly different at alpha=0.1,
pval=0.0025023877157328515
Col 58 is statistically significantly different at alpha=0.1,
pval=2.6962525556029266e-05
Col 59 is statistically significantly different at alpha=0.1,
pval=0.0019216539695931267
Col 60 is statistically significantly different at alpha=0.1,
pval=0.00033438396887247606
Col 61 is statistically significantly different at alpha=0.1,
pval=0.0008242570459537771
Col 62 is statistically significantly different at alpha=0.1,
pval=0.0004852887998258157
Col 63 is statistically significantly different at alpha=0.1,
pval=0.0016088089838123066
Col 64 is statistically significantly different at alpha=0.1,
pval=0.0001334545242192891
Col 65 is statistically significantly different at alpha=0.1,
pval=4.995392068468293e-05
Col 66 is statistically significantly different at alpha=0.1,
pval=0.0030688899118606642
Col 67 is statistically significantly different at alpha=0.1,
pval=9.018033999159097e-05
Col 68 is statistically significantly different at alpha=0.1,
pval=2.9533282550547202e-05
Col 69 is statistically significantly different at alpha=0.1,
pval=0.00012029740322040263
Col 70 is statistically significantly different at alpha=0.1,
pval=6.769335337126961e-05
Col 71 is statistically significantly different at alpha=0.1,
pval=0.05101543875512745
```

Col 72 is statistically significantly different at alpha=0.1,
pval=0.004033753896968596
Col 73 is statistically significantly different at alpha=0.1,
pval=0.0002907950396716467
Col 74 is statistically significantly different at alpha=0.1,
pval=0.0025378015241352275
Col 75 is statistically significantly different at alpha=0.1,
pval=0.0011707847380308444
Col 76 is statistically significantly different at alpha=0.1,
pval=2.8109258600491036e-06
Col 77 is statistically significantly different at alpha=0.1,
pval=0.006320832809171463
Col 78 is statistically significantly different at alpha=0.1,
pval=0.00016394580744759945
Col 79 is statistically significantly different at alpha=0.1,
pval=2.177646054694756e-05
Col 80 is statistically significantly different at alpha=0.1,
pval=0.0002589969070099564
Col 81 is statistically significantly different at alpha=0.1,
pval=0.01705508991901146
Col 82 is statistically significantly different at alpha=0.1,
pval=0.0022586607942925276
Col 83 is statistically significantly different at alpha=0.1,
pval=0.0041378438094041075
Col 84 is statistically significantly different at alpha=0.1,
pval=0.00014420418210924393
Col 85 is statistically significantly different at alpha=0.1,
pval=0.0004689923604392861
Col 86 is statistically significantly different at alpha=0.1,
pval=2.7785089511587993e-06
Col 87 is statistically significantly different at alpha=0.1,
pval=0.0008304786395584455
Col 88 is statistically significantly different at alpha=0.1,
pval=0.01635969650367854
Col 89 is statistically significantly different at alpha=0.1,
pval=0.00024764907556402494
Col 90 is statistically significantly different at alpha=0.1,
pval=0.00018697358107784835
Col 91 is statistically significantly different at alpha=0.1,
pval=2.3380525624090842e-05
Col 92 is statistically significantly different at alpha=0.1,
pval=0.001141107188424047
Col 93 is statistically significantly different at alpha=0.1,
pval=4.824351859728791e-07
Col 94 is statistically significantly different at alpha=0.1,
pval=0.00035519133515647975
Col 95 is statistically significantly different at alpha=0.1,
pval=0.0635327469365356

```
Col 96 is statistically significantly different at alpha=0.1,
pval=2.0813257431878083e-05
Col 97 is statistically significantly different at alpha=0.1,
pval=0.0028639977759847886
Col 98 is statistically significantly different at alpha=0.1,
pval=0.0003180121762895864
Col 99 is statistically significantly different at alpha=0.1,
pval=0.0005045744280018815
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.