

▼ COMPARING TWO SAMPLES

by Dr Juan H Klopper

- Research Fellow
- School for Data Science and Computational Thinking
- Stellenbosch University



▼ INTRODUCTION

In this notebook we build on our understanding of sampling distributions by investigating a few examples. Here, we only have the data from one study and, as is most often the case, we do not have access to the whole population.

We will still build sampling distributions of test statistics under a null hypothesis to put the test statistic of our data in perspective.

▼ PACKAGES USED IN THIS NOTEBOOK

```
1 # Data table configuration for Colab
2 %load_ext google.colab.data_table
```

```
1 %config InlineBackend.figure_format = "retina" # For Retina type displays
```

```
1 from google.colab import drive # For connecting to our Google Drive
```

```
1 import pandas as pd # Data import and manipulation
2 import numpy as np # Numerical computing
3 from scipy import stats # Statistics module
```

```
1 # Data visualisation
```

```
2 import plotly.graph_objects as go
3 import plotly.express as px
4 import plotly.io as pio
5 pio.templates.default = 'plotly_white'
```

▼ DATA IMPORT

We follow the now familiar process of importing data from our Google Drive, stored in a sub directory to the one in which this notebook resides.

```
1 # Log on and list files in the DATA directory of your Google Drive
2 drive.mount('/gdrive')
3 %cd '/gdrive/My Drive/Stellenbosch University/School for Data Science and Computat

Mounted at /gdrive
/gdrive/My Drive/Stellenbosch University/School for Data Science and Computat
```

```
1 df = pd.read_csv('data.csv') # Importing the CSV files
```

```
1 df # Printing the DataFrame to the screen
```

1 to 25 of 200 entries Filter ?

index	Name	DOB	Age	Vocation	Smoke	HR	sBP	CholesterolBefore	TAG	Survey
0	Dylan Patton	1981-10-07	43	Energy manager	0	47	145	1.2	1.2	1
1	Sandra Howard	1993-01-27	53	Tax adviser	0	51	115	1.2	0.6	3
2	Samantha Williams	1973-12-21	33	IT consultant	0	54	120	2.0	1.3	3
3	Ashley Hensley	1981-12-01	43	Nurse, children's	0	54	103	2.1	1.6	4
4	Robert Wilson	1964-06-23	46	Clinical embryologist	0	61	138	2.8	2.1	5
5	Leslie Diaz	1994-08-25	48	Politician's assistant	0	59	122	2.8	1.4	4
6	Frank Zimmerman	1981-03-04	54	Police officer	0	60	129	2.9	2.4	1
7	Aaron Harris	1948-01-10	58	Nurse, children's	0	61	131	3.1	2.2	1
8	William Smith	1998-11-20	44	Scientific laboratory technician	0	58	111	3.1	2.4	1
9	Andrea Fletcher	1955-12-23	31	Lexicographer	0	59	122	3.2	1.7	5
10	James Wells	1998-08-09	45	Charity fundraiser	0	62	121	3.2	1.7	4

```
1 df.shape # Number of patients and variables
(200, 13)
```

There are 200 observations and 13 variables.

COMPARING THE DISTRIBUTION OF A NUMERICAL VARIABLE BETWEEN TWO INDEPENDENT GROUPS

Comparing the distribution of data values for a continuous numerical variable between two groups, requires splitting the data along the sample space of one of the categorical variables (or then a numerical variable from which bins have been created). We consider an appropriate research question to investigate the comparison of two distributions.

RESEARCH QUESTION: *Is there a difference in heart rate values (the `HR` variable) between the Active group and the Control group (the `Group` variable)?*

Note how the groups are formed by the sample space elements of a nominal categorical variable. The two sample space elements, `Active` and `Control` are also independent of each other.

In this example we are going to make use of numpy arrays to hold the heart rate values for each group.


```
1 # Array for heart rates of control participants
2 hr_control = df[df.Group == 'Control']['HR'].to_numpy()
3
4 # Array for heart rate of active participants
5 hr_active = df[df.Group == 'Active']['HR'].to_numpy()
```

Our analysis is, as always, preceded by the use of descriptive statistics and visualisation.

▼ DESCRIPTIVE STATISTICS

We can group by the `Group` variable and then use the `describe` method to return summary statistics of the `HR` variable. Remember that the `groupby` function is used to generate groups according to the sample space elements of a variable.

```
1 df.groupby('Group')['HR'].describe()
```

1 to 2 of 2 entries 

Group	count	mean	std	min	25%	50%	75%	max
Active	100.0	76.88	11.767340524957323	47.0	67.0	80.5	87.0	104.0
Control	100.0	72.43	12.21710590801589	24.0	65.0	69.0	80.25	99.0

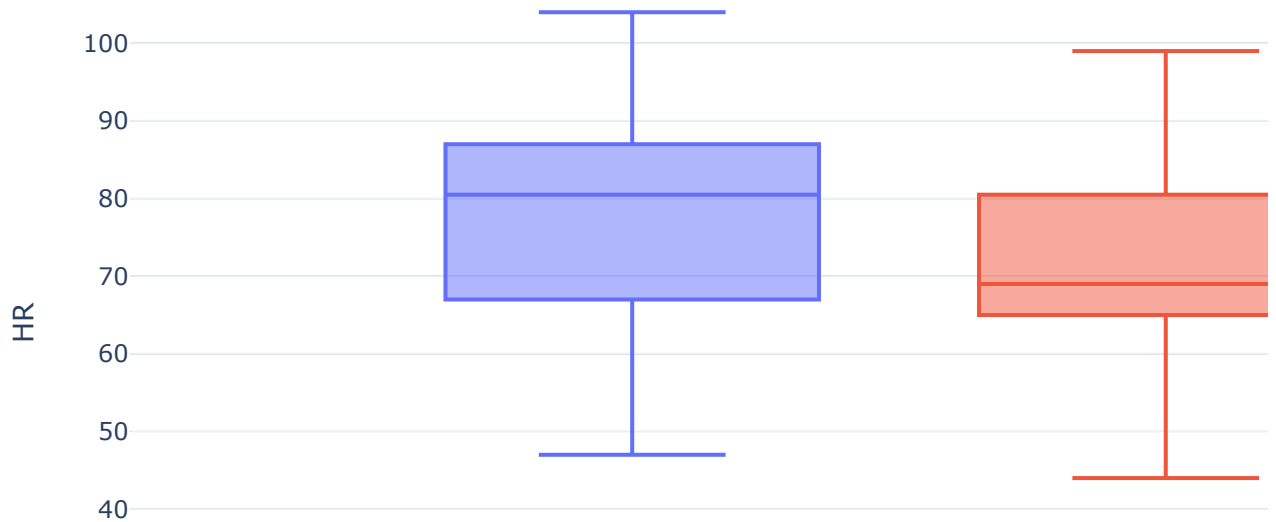
Show per page

▼ VISUALIZATION

Since we have a continuous numerical variable, a box plot will provide a good visual summary of the data.

```
1 px.box(df,
2       y = 'HR',
3       color = 'Group',
4       title='Heart rate distribution among treatment groups')
```

Heart rate distribution among treatment groups



▼ COMPARING THE TWO SAMPLES

Our **null hypothesis** states that there is no difference in the distribution of heart rate between the two groups (or that it is higher in the `Control` group). It can be denoted by

$$H_0 : HR_{\text{Active}} = HR_{\text{Control}}$$

Our **alternatine hypothesis** states that the heart rate in the `Active` group is different from the heart rate in the `Control` group. It can be denoted by $H_\alpha = HR_{\text{Active}} \neq HR_{\text{Control}}$.

A good test statistic to compare the difference in this continuous numerical variable is the mean, or then the difference in means.

```
1 # Our test statistic
2 np.mean(hr_active - hr_control)

4.45
```

Remember that this could also be -4.45 .

In the previous notebook, we used formal statistical tests to compare these means. Here, we expand our understanding of hypotheses testing using a more intuitive approach than statistical tests. We will use a statistical test at the end to verify our results.

In our next step, we have to calculate the distribution of our test statistic under the null hypothesis. In practical terms, this states a random reallocation of group status. There are 200

samples in the dataset, with 100 cases in each group. We randomly reassign a group to each heart rate. Doing this repeatedly and collecting all the means (and then differences in means) will yield a distribution of sampling mean differences (our test statistic). This is done 10000 below, populating a list object, `mean_stat` with 10000 mean differences.

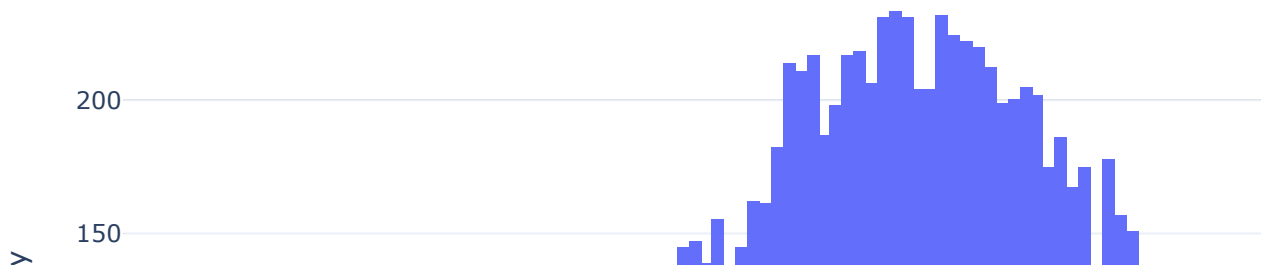
We can do this because our assumption on which we build is that the values are equal in both groups.

```
1 mean_stat = []
2
3 for i in range(10000):
4     grouping = np.random.choice(df.HR, size=(100, 2), replace=False)
5     groupI = np.mean(grouping[0:100, 0])
6     groupII = np.mean(grouping[0:100, 1])
7     mean_stat.append(groupI - groupII)
```

We look at a histogram of the sampling distribution of means (difference in means) and our original difference.

```
1 go.Figure(
2     data=go.Histogram(
3         x=mean_stat,
4         name='Mean differences'
5     )
6 ).add_trace(go.Scatter(
7     x=[4.45, 4.45],
8     y=[0, 140],
9     mode='lines',
10    name='Original difference'
11 ).update_layout(
12    title='Distribution of difference in means',
13    xaxis={'title': 'Difference'},
14    yaxis={'title': 'Frequency'}
15 )
```

Distribution of difference in means



We note that the difference in means of our study in relation to the distribution of differences in means under the null hypothesis was a rare finding indeed. Since Python stores a `False` value as `0` and a `True` value as `1`, we can sum over the `True` and `False` values, expressing the number of cases in the `mean_stat` list above, i.e. that were larger than our data's value of `4.45`.

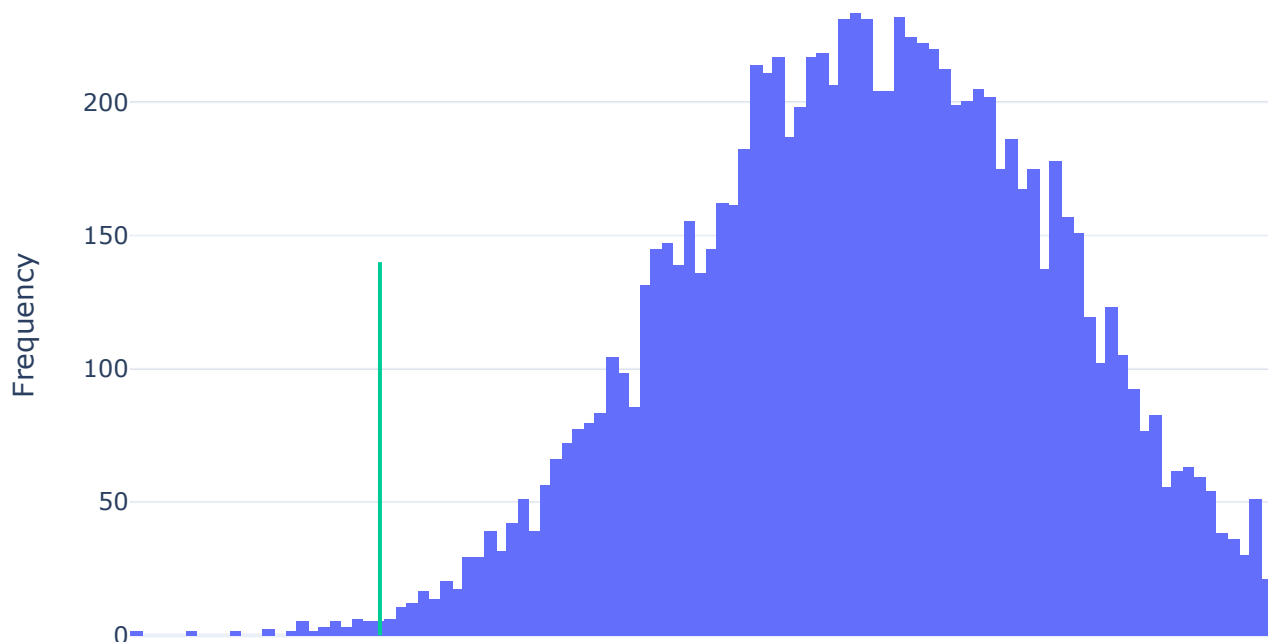
```
1 np.sum(np.array(mean_stat) > 4.45) / 10000
0.0052
```

-6 -4 -2 0 2

Since we could also have subtracted in a different order, we also need to consider all the fraction of sampling distribution values less than -4.45 , as we can see from the histogram below.

```
1 go.Figure(
2     data=go.Histogram(
3         x=mean_stat,
4         name='Mean differences'
5     )
6 ).add_trace(go.Scatter(
7     x=[4.45, 4.45],
8     y=[0, 140],
9     mode='lines',
10    name='Original difference'
11 ).add_trace(go.Scatter(
12    x=[-4.45, -4.45],
13    y=[0, 140],
14    mode='lines',
15    name='Reverse order subtraction'
16 ).update_layout(
17     title='Distribution of difference in means',
18     xaxis={'title': 'Difference'},
19     yaxis={'title': 'Frequency'}
20 )
```

Distribution of difference in means



We use the same method to calculate the fraction *below* as we did with the *above* calculation.

```
1 np.sum(np.array(mean_stat) < -4.45) / 10000
0.0034
```

We add these values to get $0.0044 + 0.0041 = 0.0085 \approx 0.009$.

This is a simulated p value for our study. It is much smaller than a chosen α value of 0.05 and we can reject the null hypothesis.

Let's recap. Since we don't have access to the complete population we used a technique of reassignment to our known data under the null hypothesis that there is no difference between the groups. From this we built a sampling distribution. We looked at how many times the sampling distribution values was more and less than our finding (with subtraction in either order).

Just to confirm, we also use Student's t to calculate a p value.

▼ COMPARING MEANS WITH STUDENT'S t TEST

The t test for independent groups compares the mean values of the variable in each group. The t distribution uses the degrees of freedom parameter (as we saw in the previous notebook).

The `ttest_ind` function from the `stats` module in the `scipy` package can perform this test. It returns the t statistic and a (two-tailed) p value. We have to divide this p value by 2 to get the one-tailed p values.

```
1 t_stat, p_val = stats.ttest_ind(hr_active,
2                               hr_control)
3 p_val

0.009382957236919214
```

This is very close to the p value that we calculated above.

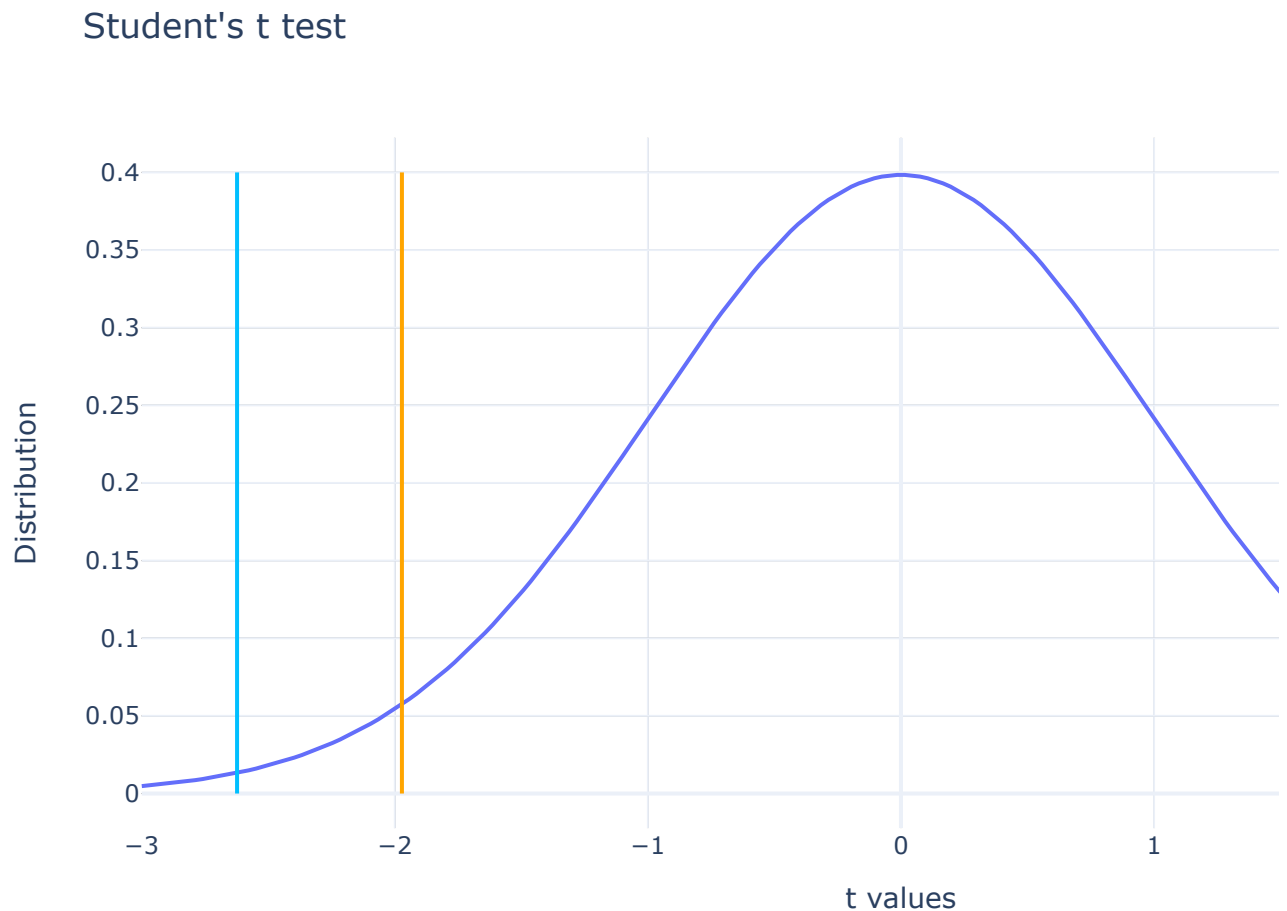
We see a t statistic and a p value for this t statistic. A visual representation is given below, where the critical t statistic (representing 2.5% (below) and 2.5% (above) of the total area under the curve) is in orange and the t statistic for our data is in blue.

```
1 t_stat

2.623426872064485

1 t_vals = np.linspace(-3, 3, 200) # Generating some values for the x-axis
2 t_pdf_vals = stats.t.pdf(t_vals, 198) # Calculating the PDF value for each of t
3
4 t_dist_fig = go.Figure()
5
6 t_dist_fig.add_trace(go.Scatter(x=t_vals,
7                                y=t_pdf_vals,
8                                mode='lines',
9                                name='t distribution'))
10
11 t_dist_fig.update_layout(title="Student's t test",
12                           xaxis=dict(title='t values'),
13                           yaxis=dict(title='Distribution'))
14
15 t_dist_fig.add_trace(go.Scatter(
16     x=[t_stat, t_stat],
17     y=[0,0.4],
18     name='t statistic',
19     mode='lines',
20     marker=dict({'color':'deepskyblue'}))
21 ))
22
23 t_dist_fig.add_trace(go.Scatter(
24     x=[-t_stat, -t_stat],
25     y=[0,0.4],
```

```
26     name='t statistic',
27     mode='lines',
28     marker=dict({'color':'deepskyblue'})
29 ))
30
31 t_crit = stats.t.ppf(0.975, 198)
32
33 t_dist_fig.add_trace(go.Scatter(
34     x=[t_crit, t_crit],
35     y=[0,0.4],
36     name='critical t statistic',
37     mode='lines',
38     marker=dict({'color':'orange'})
39 ))
40
41 t_dist_fig.add_trace(go.Scatter(
42     x=[-t_crit, -t_crit],
43     y=[0,0.4],
44     name='critical t statistic',
45     mode='lines',
46     marker=dict({'color':'orange'})
47 ))
48
49 t_dist_fig.show()
```



For a chosen α value of 0.05 (which is above and below the orange lines), we reject the null hypothesis and accept the alternative hypothesis and state that the heart rate in the active group is significantly different from the heart rate in the control group.

COMPARING THE MEANS OF SYSTOLIC BLOOD PRESSURE BETWEEN AGE GROUPS

Here we consider the difference in systolic blood pressure (`sBP` variable) between younger and older patients. Our null hypothesis is that there is no difference in the systolic blood pressure between the groups. Our alternative hypothesis is that there is a difference. Hypothesis testing is therefore two-sided.

In this example, we review how to work with data and create two groups of age by binning the data using a conditional. We will let every participant younger than 65 be in age group I and every participants 65 and older be in age group II.

```
1 # Creating a new variable using the where function
2 df['AgeGroup'] = np.where(df.Age < 65, 'I', 'II')
```

DESCRIPTIVE STATISTICS


The result is an unbalanced variable, where we have an over-representation of the younger participants.

```
1 df.AgeGroup.value_counts()

I      152
II     48
Name: AgeGroup, dtype: int64
```

The `groupby` method is used to describe the systolic blood pressure in both groups.

```
1 df.groupby('AgeGroup').sBP.describe()
```

1 to 2 of 2 entries 

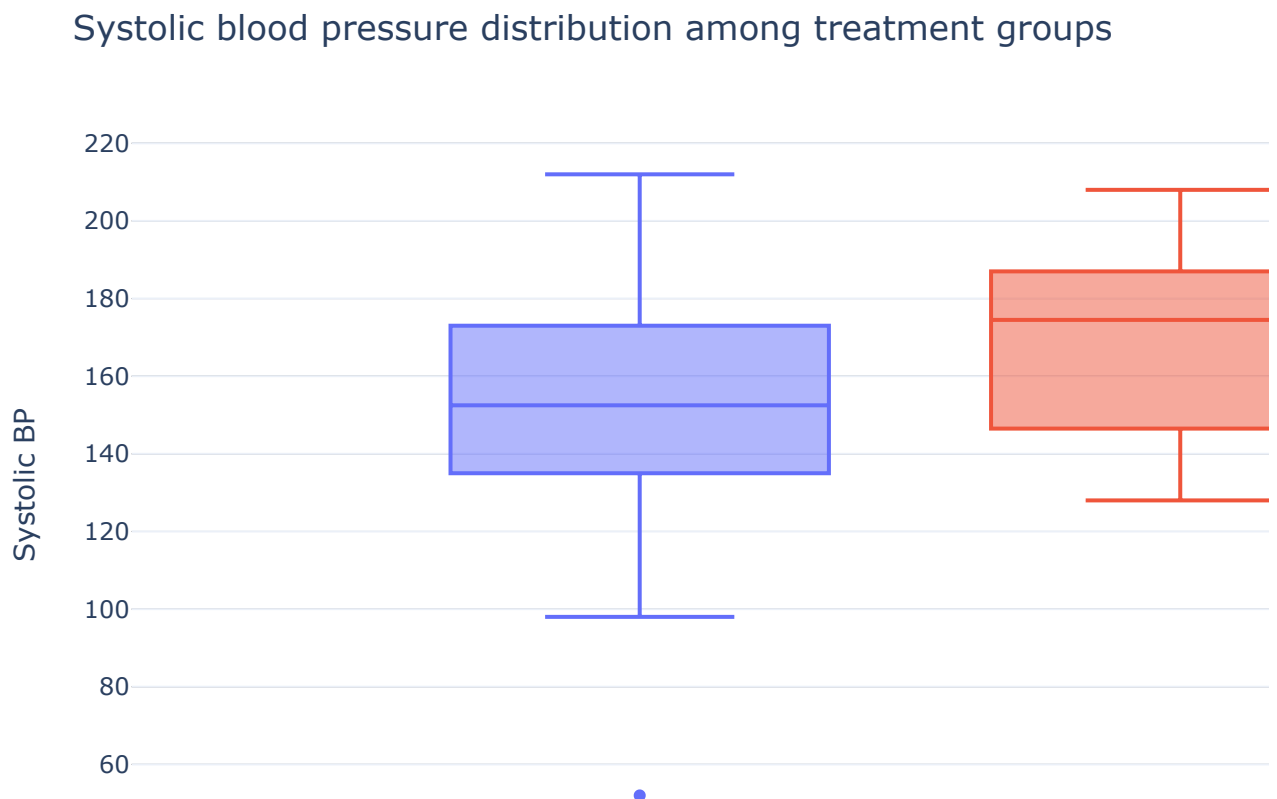
AgeGroup	count	mean	std	min	25%	50%	75%	max
I	152.0	153.66447368421052	25.351553129517992	52.0	135.0	152.5	173.0	212.0
II	48.0	168.47916666666666	23.68587835258244	128.0	146.75	174.5	187.0	208.0

Show per page

Younger participants have a lower mean blood pressure. Below, we visualise this difference.

▼ VISUALISATION

```
1 px.box(df,  
2     y = 'sBP',  
3     color = 'AgeGroup',  
4     title='Systolic blood pressure distribution among treatment groups',  
5     labels={'sBP':'Systolic BP'})
```



We need to know how different the means are.

▼ COMPARING THE VARIABLE BETWEEN THE TWO GROUPS

As with our previous use of the null hypothesis, we assume that the systolic blood pressure (the `sBP` variable) is independent of the age group.

We can reassign the systolic blood pressure. Here, we accomplish the task by repeatedly shuffling the values in the `SBP` numpy array, using the `random.shuffle` numpy function. We have to be careful with pandas here. When we do the shuffle, we actually change the original dataframe, even though we extracted the column and saved it as a separate numpy array. We therefore create an independent copy of the dataframe to work with

```
1 # Make an independent copy of the dataframe
2 df_copy = df.copy(deep=True)

1 mean_stat = []
2
3 for i in range(10000):
4     sBP = df_copy.SBP.to_numpy() # Reset the original array
5     np.random.shuffle(sBP) # Reshuffle the array randomly
6     groupI = np.mean(sBP[0:152]) # Select the first 152 observations
7     groupII = np.mean(sBP[152:201]) # Select the last 48 observations
8     mean_stat.append(groupI - groupII) # Calculate and store the difference in mean
```

Once again we view the sampling distribution of the mean difference test statistic. First, though, we store the difference in means for the sample data.

```
1 # Creating separate numpy arrays
2 younger_sBP = df.loc[df.AgeGroup == 'I'].SBP.to_numpy()
3 older_sBP = df.loc[df.AgeGroup == 'II'].SBP.to_numpy()
4
5 # Difference in means
6 mean_diff = np.mean(younger_sBP) - np.mean(older_sBP)
7 mean_diff

-14.814692982456137
```

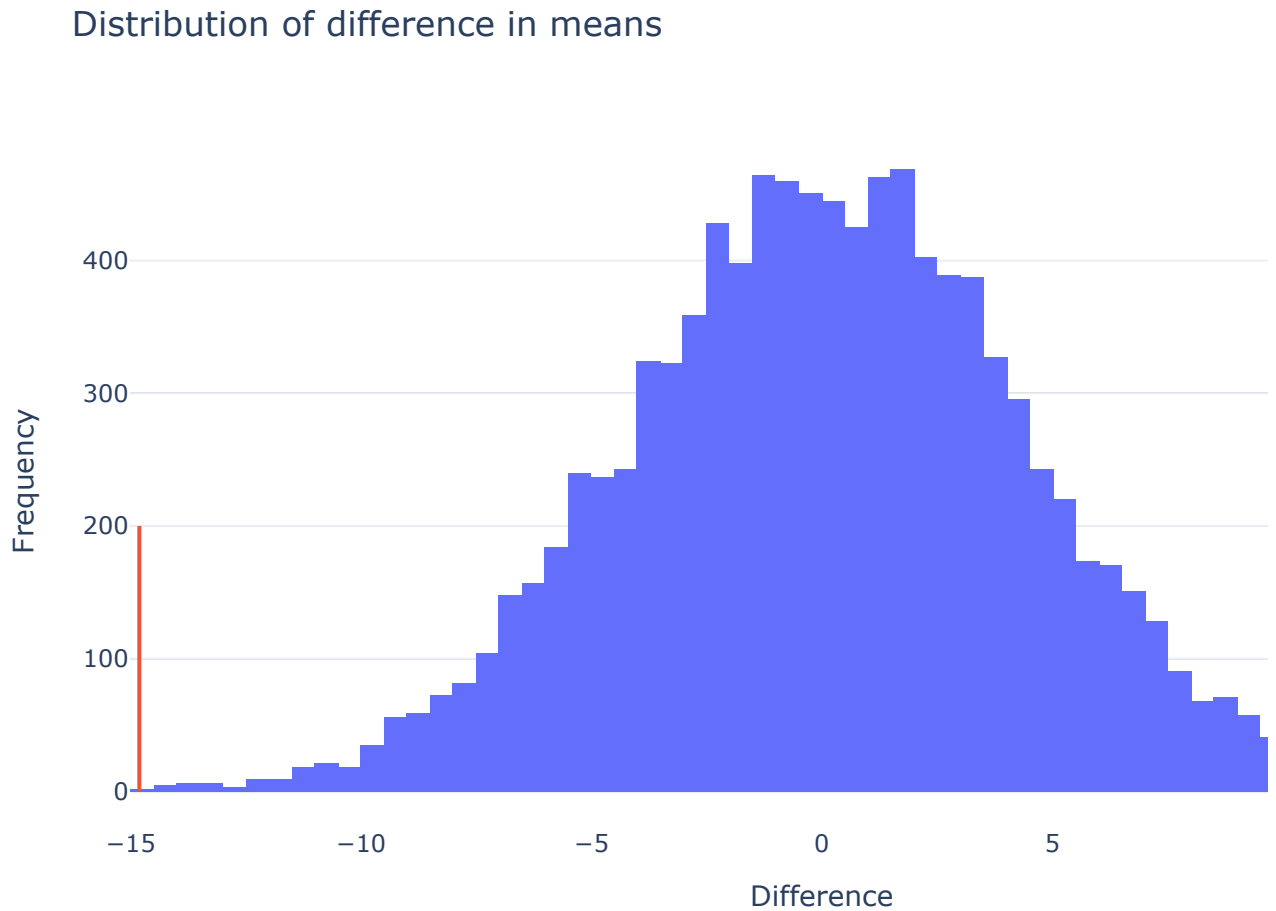
Since this is a two-tailed hypothesis, we need to reflect this difference. Below, we create a histogram of the mean difference sample distribution and the two mean differences from the data.

```
1 go.Figure(
2     data=go.Histogram(
3         x=mean_stat,
4         name='Mean differences'
5     )
6 ).add_trace(go.Scatter(
7     x=[mean_diff, mean_diff],
8     y=[0, 200],
9     mode='lines',
10    name='Original difference'
11 ).add_trace(go.Scatter(
12    x=[-mean_diff, -mean_diff],
13    y=[0, 200],
14    mode='lines',
15    name='Reflected original difference'
```

```

16 ).update_layout(
17     title='Distribution of difference in means',
18     xaxis={'title': 'Difference'},
19     yaxis={'title': 'Frequency'}
20 )

```



Below, we view both the fractions below and above our mean difference.

```
1 np.sum(np.array(mean_stat) < mean_diff) / 10000
```

```
0.0
```

```
1 np.sum(np.array(mean_stat) > -mean_diff) / 10000
```

```
0.0
```

Combined, we have a very small fraction of values more extreme than our original difference. We can verify this again with a t test.

▼ STUDENT'S t TEST

```

1 stats.ttest_ind(
2     younger_sBP,
3     older_sBP
4 )

```

```
Ttest_indResult(statistic=-3.5839931075832485, pvalue=0.00042605478175169987)
```

We are usually only interested in two decimal places, so in both cases we would have $p < 0.01$.

For the sake of interest we look at one more t test that we use when the variances in our continuous numerical variable is different between two groups.

▼ COMPARING MEANS WITH UNEQUAL VARIANCES

Student's t test assumes that the data are from populations in which the variances of the variable are equal. This can be verified using Levene's test. The Levene test null hypothesis states that the variances are indeed equal and the alternative hypothesis is that they are not.

We will consider if there is a difference in age between two randomly created groups.

```

1 # Creating two numpy arrays to hold the age values with different variances
2 np.random.seed(12)
3 age_I = np.random.normal(loc=100, scale=10, size=100)
4 age_II = np.random.normal(loc=100, scale=12.1, size=100)

```

We generate two arrays with the same mean and size. One is taken from a normal distribution with a standard deviation of 10 and the other being 12.1. Is this a significant difference.

Below we use the `levене` function from the `stats` module of the `scipy` library. The two arrays are used as arguments.

```

1 stats.levене(age_I, age_II)

LeveneResult(statistic=4.200486278483402, pvalue=0.04173087591445651)

```

We note that we reject the null hypothesis. We now use the t test for unequal variances, termed the **Welch test**.

▼ t TEST FOR UNEQUAL VARIANCES

This test is simple to perform and requires the addition of the `equal_var` argument to the

```
1 stats.ttest_ind(  
2     age_I,  
3     age_II,  
4     equal_var=False  
5 )
```

```
Ttest_indResult(statistic=0.7190917638181298, pvalue=0.4729566937881784)
```

Here, we fail to reject the null hypothesis.

▼ CONCLUSION

The t tests are commonly used in data science. They are termed **parametric tests** for the comparison of two means as they are calculated from theoretical distributions based on parameters, i.e. the t distribution is based on the parameter of degrees of freedom.

We have seen though that we can build sampling distributions from our original data under the null hypothesis that there is no difference and from this we can estimate the difference between groups.

1