

# Assumptions in hypothesis testing

HYPOTHESIS TESTING IN PYTHON



**James Chapman**

Content Developer, DataCamp

# Randomness

## Assumption

The samples are random subsets of larger populations

## Consequence

- Sample is not representative of population

## How to check this

- Understand how your data was collected
- Speak to the data collector/domain expert



<sup>1</sup> Sampling techniques are discussed in "Sampling in Python".

# Independence of observations

## Assumption

Each observation (row) in the dataset is independent

## Consequence

- Increased chance of false negative/positive error

## How to check this

- Understand how our data was collected

# Large sample size

## Assumption

The sample is big enough to mitigate uncertainty, so that the Central Limit Theorem applies

## Consequence

- Wider confidence intervals
- Increased chance of false negative/positive errors

## How to check this

- It depends on the test

# Large sample size: t-test

## One sample

- At least 30 observations in the sample

$$n \geq 30$$

$n$ : sample size

## Paired samples

- At least 30 pairs of observations across the samples

Number of rows in our data  $\geq 30$

## Two samples

- At least 30 observations in each sample

$$n_1 \geq 30, n_2 \geq 30$$

$n_i$ : sample size for group  $i$

## ANOVA

- At least 30 observations in each sample

$$n_i \geq 30 \text{ for all values of } i$$

# Large sample size: proportion tests

## One sample

- Number of successes in sample is greater than or equal to 10

$$n \times \hat{p} \geq 10$$

- Number of failures in sample is greater than or equal to 10

$$n \times (1 - \hat{p}) \geq 10$$

$n$ : sample size

$\hat{p}$ : proportion of successes in sample

## Two samples

- Number of successes in each sample is greater than or equal to 10

$$n_1 \times \hat{p}_1 \geq 10$$

$$n_2 \times \hat{p}_2 \geq 10$$

- Number of failures in each sample is greater than or equal to 10

$$n_1 \times (1 - \hat{p}_1) \geq 10$$

$$n_2 \times (1 - \hat{p}_2) \geq 10$$

# Large sample size: chi-square tests

- The number of successes in each group is greater than or equal to 5

$$n_i \times \hat{p}_i \geq 5 \text{ for all values of } i$$

- The number of failures in each group is greater than or equal to 5

$$n_i \times (1 - \hat{p}_i) \geq 5 \text{ for all values of } i$$

$n_i$ : sample size for group  $i$

$\hat{p}_i$ : proportion of successes in sample group  $i$

# Sanity check

If the bootstrap distribution doesn't look normal, assumptions likely aren't valid

- Revisit data collection to check for **randomness**, **independence**, and **sample size**



**Let's practice!**  
HYPOTHESIS TESTING IN PYTHON

# Non-parametric tests

HYPOTHESIS TESTING IN PYTHON



**James Chapman**

Content Developer, DataCamp

# Parametric tests

- z-test, t-test, and ANOVA are all **parametric** tests
- Assume a normal distribution
- Require sufficiently large sample sizes

# Smaller Republican votes data

```
print(repub_votes_small)
```

	state	county	repub_percent_08	repub_percent_12
80	Texas	Red River	68.507522	69.944817
84	Texas	Walker	60.707197	64.971903
33	Kentucky	Powell	57.059533	61.727293
81	Texas	Schleicher	74.386503	77.384464
93	West Virginia	Morgan	60.857614	64.068711

# Results with pingouin.ttest()

- 5 pairs is not enough to meet the sample size condition for the paired t-test:
  - At least 30 pairs of observations across the samples.

```
alpha = 0.01
import pingouin
pingouin.ttest(x=repub_votes_potus_08_12_small['repub_percent_08'],
               y=repub_votes_potus_08_12_small['repub_percent_12'],
               paired=True,
               alternative="less")
```

	T	dof	alternative	p-val	CI95%	cohen-d	BF10	power
T-test	-5.875753	4	less	0.002096	[-inf, -2.11]	0.500068	26.468	0.239034

# Non-parametric tests

- Non-parametric tests avoid the parametric assumptions and conditions
- Many non-parametric tests use *ranks* of the data

```
x = [1, 15, 3, 10, 6]
```

```
from scipy.stats import rankdata  
rankdata(x)
```

```
array([1., 5., 2., 4., 3.])
```

# Non-parametric tests

- Non-parametric tests are more reliable than parametric tests for **small sample sizes** and when data **isn't normally distributed**

# Non-parametric tests

- Non-parametric tests are more reliable than parametric tests for **small sample sizes** and when data **isn't normally distributed**

## Wilcoxon-signed rank test

- Developed by Frank Wilcoxon in 1945
- One of the first non-parametric procedures



# Wilcoxon-signed rank test (Step 1)

- Works on the ranked absolute differences between the pairs of data

```
repub_votes_small['diff'] = repub_votes_small['repub_percent_08'] -  
                             repub_votes_small['repub_percent_12']  
  
print(repub_votes_small)
```

	state	county	repub_percent_08	repub_percent_12	diff
80	Texas	Red River	68.507522	69.944817	-1.437295
84	Texas	Walker	60.707197	64.971903	-4.264705
33	Kentucky	Powell	57.059533	61.727293	-4.667760
81	Texas	Schleicher	74.386503	77.384464	-2.997961
93	West Virginia	Morgan	60.857614	64.068711	-3.211097

# Wilcoxon-signed rank test (Step 2)

- Works on the ranked absolute differences between the pairs of data

```
repub_votes_small['abs_diff'] = repub_votes_small['diff'].abs()  
print(repub_votes_small)
```

	state	county	repub_percent_08	repub_percent_12	diff	abs_diff
80	Texas	Red River	68.507522	69.944817	-1.437295	1.437295
84	Texas	Walker	60.707197	64.971903	-4.264705	4.264705
33	Kentucky	Powell	57.059533	61.727293	-4.667760	4.667760
81	Texas	Schleicher	74.386503	77.384464	-2.997961	2.997961
93	West Virginia	Morgan	60.857614	64.068711	-3.211097	3.211097

# Wilcoxon-signed rank test (Step 3)

- Works on the ranked absolute differences between the pairs of data

```
from scipy.stats import rankdata
repub_votes_small['rank_abs_diff'] = rankdata(repub_votes_small['abs_diff'])
print(repub_votes_small)
```

	state	county	repub_percent_08	repub_percent_12	diff	abs_diff	rank_abs_diff
80	Texas	Red River	68.507522	69.944817	-1.437295	1.437295	1.0
84	Texas	Walker	60.707197	64.971903	-4.264705	4.264705	4.0
33	Kentucky	Powell	57.059533	61.727293	-4.667760	4.667760	5.0
81	Texas	Schleicher	74.386503	77.384464	-2.997961	2.997961	2.0
93	West Virginia	Morgan	60.857614	64.068711	-3.211097	3.211097	3.0

# Wilcoxon-signed rank test (Step 4)

	state	county	repub_percent_08	repub_percent_12	diff	abs_diff	rank_abs_diff
80	Texas	Red River	68.507522	69.944817	-1.437295	1.437295	1.0
84	Texas	Walker	60.707197	64.971903	-4.264705	4.264705	4.0
33	Kentucky	Powell	57.059533	61.727293	-4.667760	4.667760	5.0
81	Texas	Schleicher	74.386503	77.384464	-2.997961	2.997961	2.0
93	West Virginia	Morgan	60.857614	64.068711	-3.211097	3.211097	3.0

- Incorporate the sum of the ranks for negative and positive differences

```
T_minus = 1 + 4 + 5 + 2 + 3
T_plus = 0
W = np.min([T_minus, T_plus])
```

```
0
```

# Implementation with pingouin.wilcoxon()

```
alpha = 0.01
pingouin.wilcoxon(x=repub_votes_potus_08_12_small['repub_percent_08'],
                  y=repub_votes_potus_08_12_small['repub_percent_12'],
                  alternative="less")
```

	W-val	alternative	p-val	RBC	CLES
Wilcoxon	0.0	less	0.03125	-1.0	0.72

Fail to reject  $H_0$ , since  $0.03125 > 0.01$

**Let's practice!**  
HYPOTHESIS TESTING IN PYTHON

# Non-parametric ANOVA and unpaired t-tests

HYPOTHESIS TESTING IN PYTHON



**James Chapman**  
Content Developer, DataCamp

# Wilcoxon-Mann-Whitney test

- Also known as the *Mann Whitney U test*
- A t-test on the ranks of the numeric input
- Works on unpaired data



# Wilcoxon-Mann-Whitney test setup

```
age_vs_comp = stack_overflow[['converted_comp', 'age_first_code_cut']]
```

```
age_vs_comp_wide = age_vs_comp.pivot(columns='age_first_code_cut',  
                                       values='converted_comp')
```

```
age_first_code_cut  adult  child  
0          77556.0    NaN  
1           NaN  74970.0  
2           NaN 594539.0  
...          ...      ...  
2258          NaN  97284.0  
2259          NaN  72000.0  
2260          NaN 180000.0
```

```
[2261 rows x 2 columns]
```

# Wilcoxon-Mann-Whitney test

```
alpha=0.01
```

```
import pingouin
pingouin.mwu(x=age_vs_comp_wide['child'],
             y=age_vs_comp_wide['adult'],
             alternative='greater')
```

	U-val	alternative	p-val	RBC	CLES
MWU	744365.5	greater	1.902723e-19	-0.222516	0.611258

# Kruskal-Wallis test

*Kruskal-Wallis test is to Wilcoxon-Mann-Whitney test as ANOVA is to t-test*

```
alpha=0.01
```

```
pingouin.kruskal(data=stack_overflow,  
                 dv='converted_comp',  
                 between='job_sat')
```

	Source	ddof1	H	p-unc
Kruskal	job_sat	4	72.814939	5.772915e-15

**Let's practice!**  
HYPOTHESIS TESTING IN PYTHON

# Congratulations!

HYPOTHESIS TESTING IN PYTHON



**James Chapman**

Content Developer, DataCamp

# Course recap

## Chapter 1

- Workflow for testing proportions vs. a hypothesized value
- False negative/false positive errors

## Chapter 2

- Testing differences in sample means between two groups using t-tests
- Extending this to more than two groups using ANOVA and pairwise t-tests

## Chapter 3

- Testing differences in sample proportions between two groups using proportion tests
- Using chi-square independence/goodness of fit tests

## Chapter 4

- Reviewing assumptions of parametric hypothesis tests
- Examined non-parametric alternatives when assumptions aren't valid

# More courses

## Inference

[Statistics Fundamentals with Python](#) skill track

## Bayesian statistics

[Bayesian Data Analysis in Python](#)

## Applications

[Customer Analytics and A/B Testing in Python](#)

# Congratulations!

HYPOTHESIS TESTING IN PYTHON