Type I errors

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Ways of being wrong

When we run a test:

	Real effect present	No real effect present
Effect found (positive : alternative hypothesis)	True Positive	**False Positive**
No effect found (negative: null hypothesis)	**False Negative**	True Negative

Type I error : find difference where none exists

Type II error : fail to find difference that does exist

Avoiding type I errors

Basis of tests

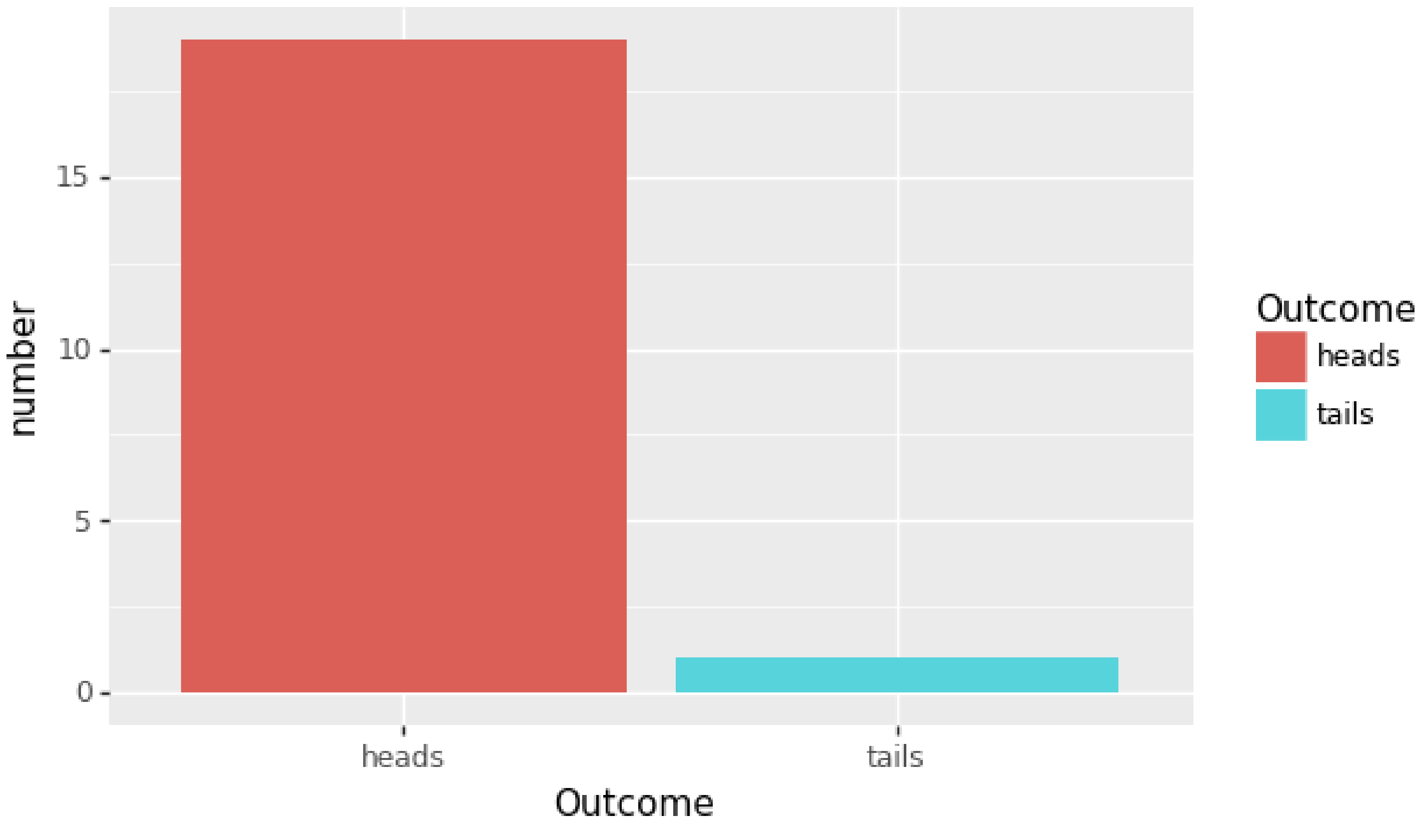
- Statistical tests are probabilistic
- Quantify likelihood of results under null hypothesis

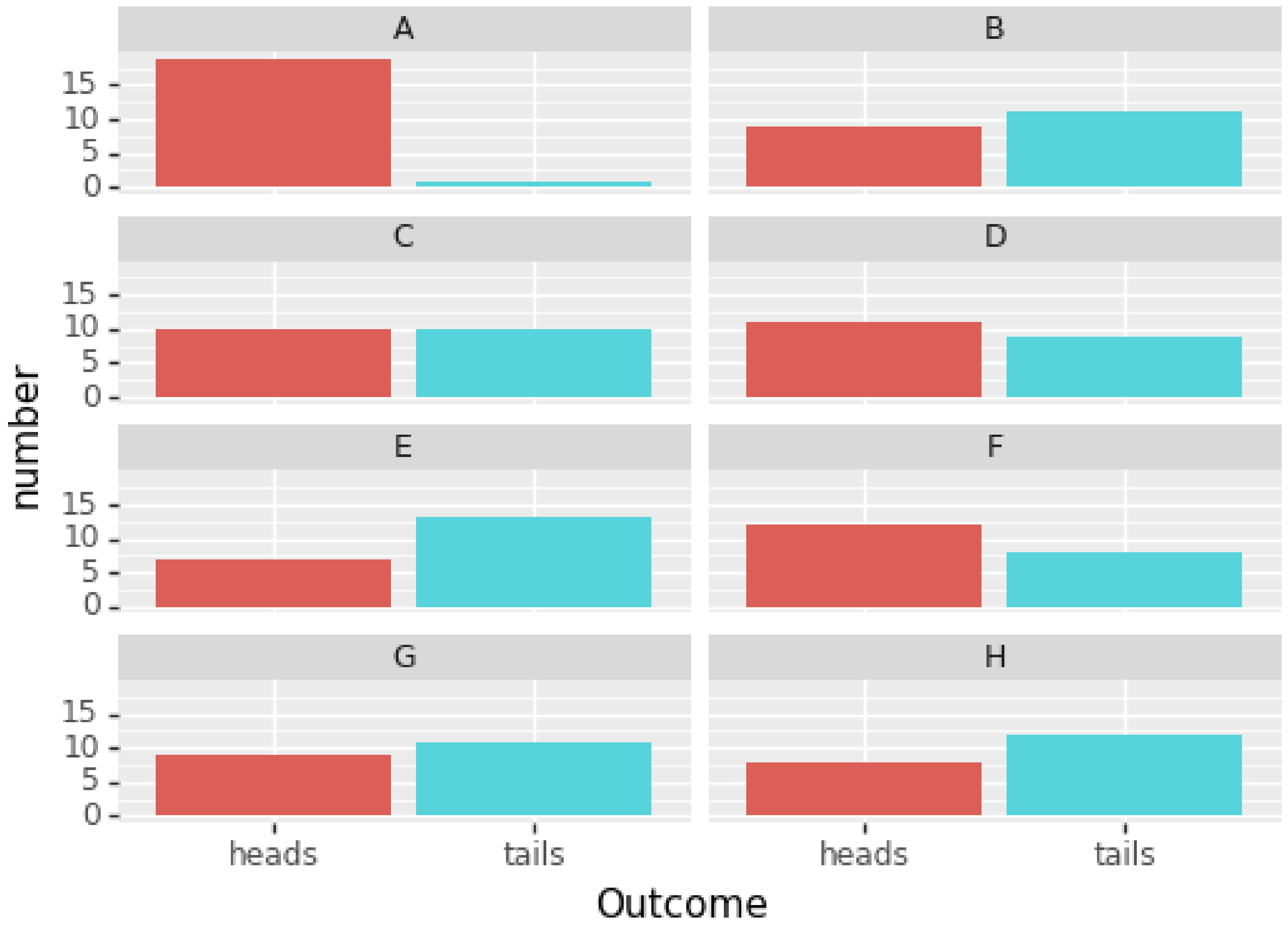
Consider:

- Significant results are *improbable*, not impossible under null hypothesis
- Still possible result are by chance

Picking a single result can be misleading

Example





Accounting for multiple tests

By design

- Avoid "p-value fishing"

By correction

- Correct p-values for presence of multiple tests

Correction methods

- Bonferroni and Šídák
- Choose method based on independence of tests

Bonferroni correction

- Conservative method
- Simple

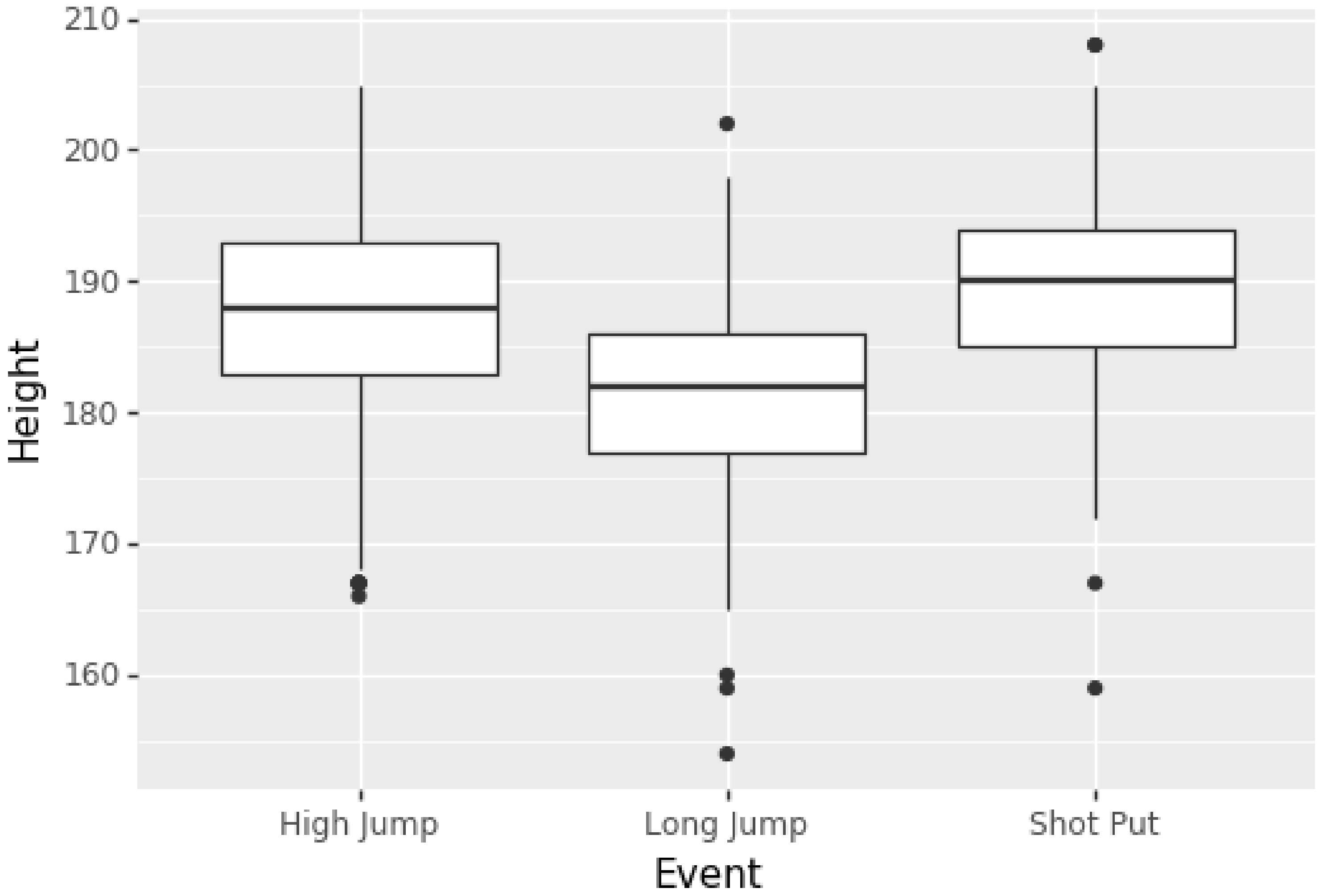
Use when

- Tests are not independent from each other

```
import statsmodels as sm
from scipy import stats

t_1= stats.ttest_ind(Array1, Array2)
t_2= stats.ttest_ind(Array2, Array3)
t_3= stats.ttest_ind(Array1, Array3)

pvals_array = [t_1[1],t_2[1],t_3[1]]

adjustedvalues=  sm.stats.multitest.multipletests(
pvals_array, alpha=0.05, method='b')
```

Bonferroni correction example

- Multiple non-independent t-tests



```python
from scipy import stats
import statsmodels as sm

t_result_1= stats.ttest_ind(HighJumpVals, LongJumpVals)
t_result_2= stats.ttest_ind(LongJumpVals, ShotPutVals)
t_result_3= stats.ttest_ind(HighJumpVals, HighJumpVals)

pvals_array = [t_result_1[1],t_result_2[1],t_result_3[1]]
adjustedvalues=  sm.stats.multitest.multipletests(pvals_array, alpha=0.05, method='b')
print(adjustedvalues)
```

```
(array([ True,  True, False]),
array([6.72030836e-63, 3.46967459e-97, 1.00000000e+00]),
0.016952427508441503, 0.016666666666666666)
```

Šídák correction

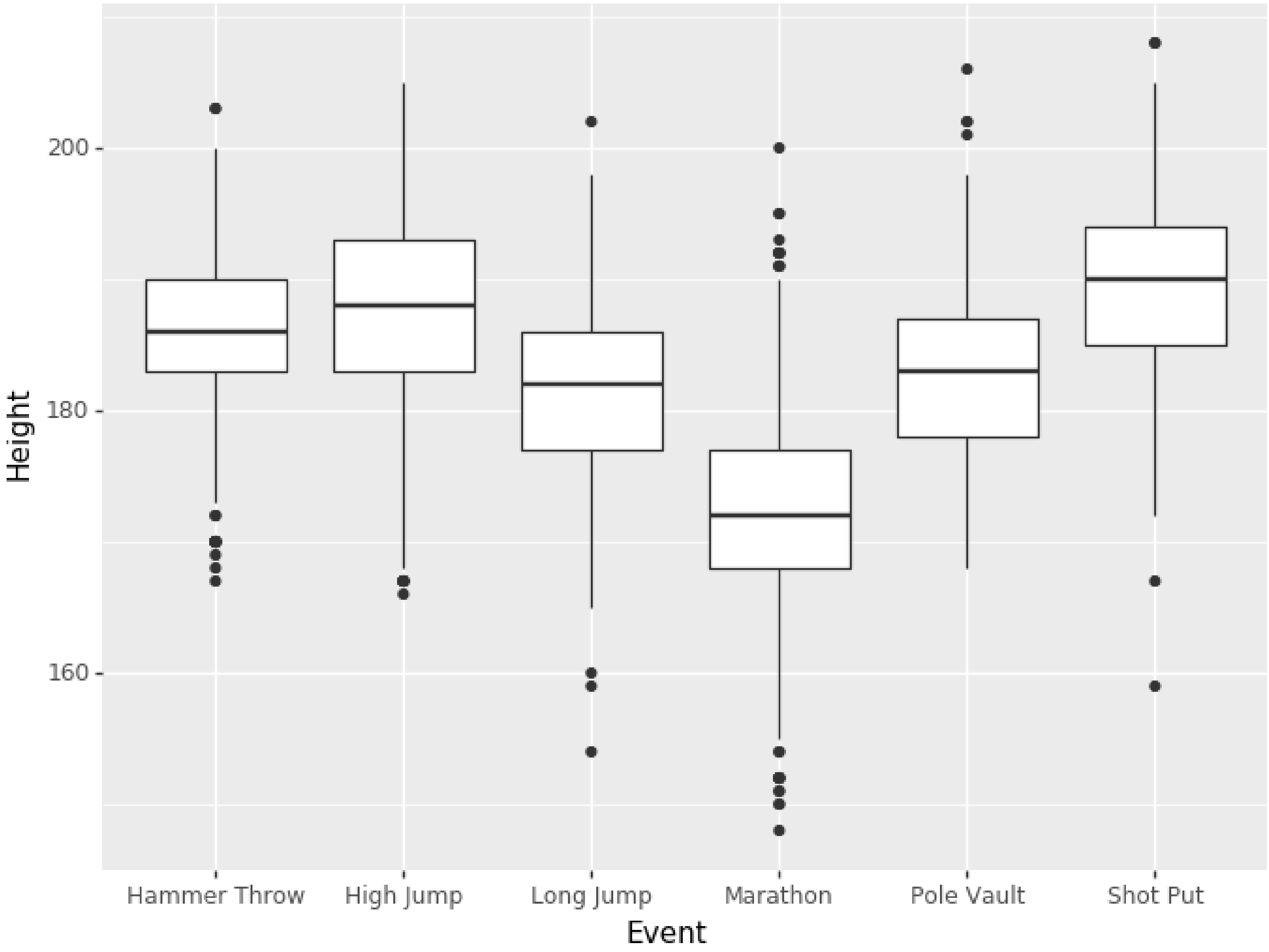
- Less conservative method

Use when

- Tests are independent from each other

```
import statsmodels as sm

t_1= stats.ttest_ind(Array1, Array2)
t_2= stats.ttest_ind(Array3, Array4)
t_3= stats.ttest_ind(Array5, Array6)

pvals_array = [t_1[1],t_2[1],t_3[1]]

adjustedvalues=  sm.stats.multitest.multipletests(
pvals_array, alpha=0.05, method='s')
```

Šídák correction example



```python
from scipy import stats
import statsmodels as sm

t_result_1 = stats.ttest_ind(HighJumpVals, LongJumpVals)
t_result_2 = stats.ttest_ind(ShotPutVals, HammerVals)
t_result_3 = stats.ttest_ind(MarathonVals, PoleVals)

pvals_array = [t_result_1[1],t_result_2[1],t_result_3[1]]
adjustedvaluesm = sm.stats.multitest.multipletests(pvals_array, alpha=0.05, method='s')
print(adjustedvalues)
```

```
(array([ True,  True,  True]), array([0., 0., 0.]),
0.016952427508441503, 0.016666666666666666)
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Sample size

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Type II errors & sample size

Definition

- False negative
- Fail to detect an effect that exists

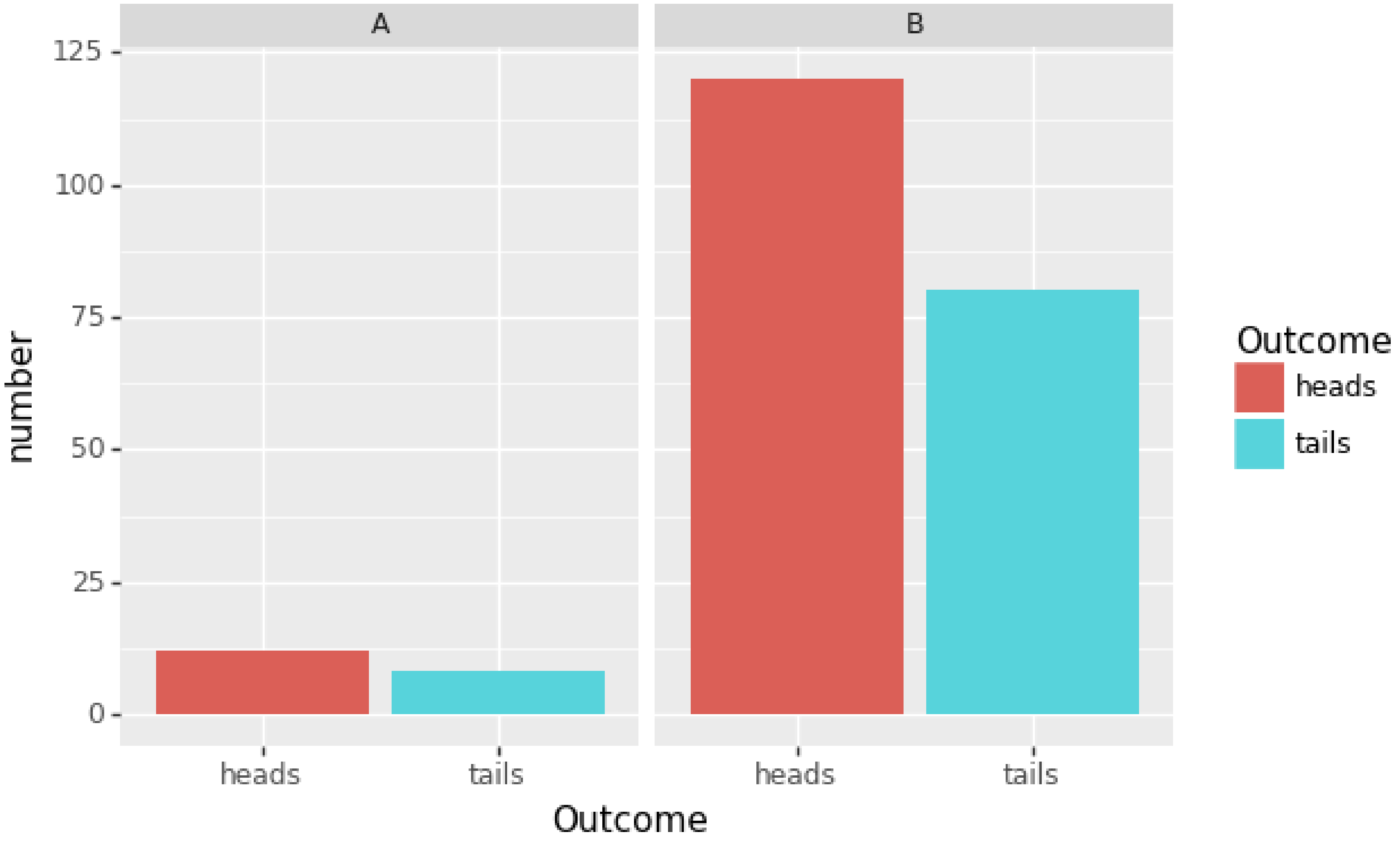
Caveat

- Can never be sure that no effect is present

Sample size

- Helps avoid false negatives
- Larger sample size = more sensitive methods

Importance of sample size



Other factors that affect sample size

Alpha

- Critical value of p at which to reject null hypothesis

Power

- Probability we correctly reject null hypothesis if alternative hypothesis is true

Effect size

- Departure from null hypothesis

Effects of other factors

Increase sample size:

- Increase statistical power
- Decrease usable alpha
- Smaller effect size detectable

What sample size do we need with `effect_size` = x, `power` = y, `alpha` = z?

Functions

- t-test: `TTestIndPower()`
- Other functions for other tests

Calculating sample size needed for t-test

Initialize analysis

`TTestIndPower()` for `ttest_ind()`

Values

`effect size` : standardized effect size

`power` : 0 - 1

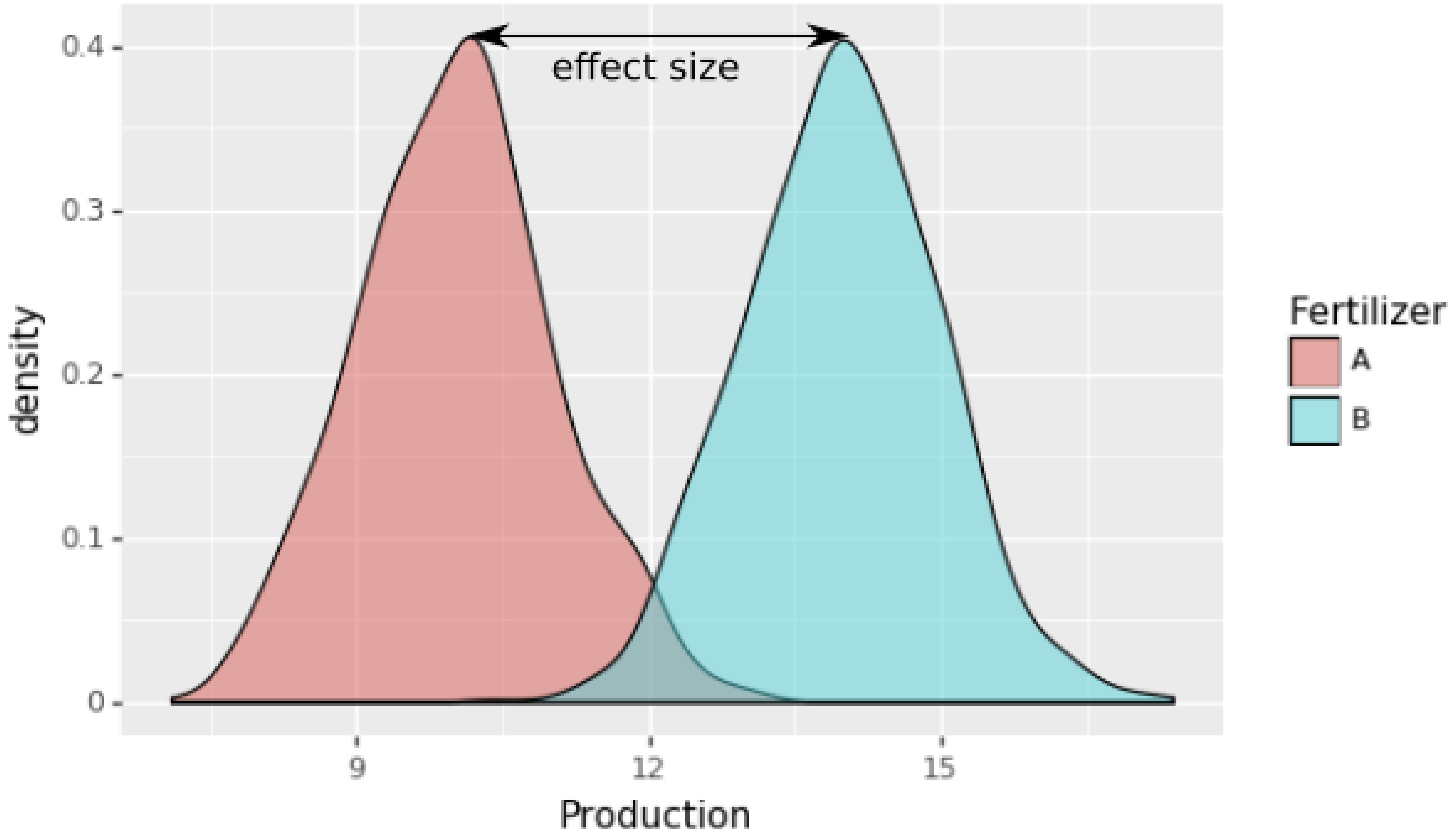
`alpha` : 0.05 standard

`ratio` : 1 if experiment balanced

`nob1s` : set to `None`

```
from statsmodels.stats import power as pwr

analysis = pwr.TTestIndPower()

ssresult = analysis.solve_power(
effect_size=effect_size,
power=power,
alpha=alpha,
ratio=1.0,
nobs1=None)
print(ssresult)
```

Sample size calculation example



Sample size calculation example

Assumptions

`effect_size` : 0.8 (large)

`power` : 0.8 (80% chance of detection)

`alpha` : 0.05 (standard)

`ratio` : (group 2 samples / group 1 samples)

```
effect_size = 0.8
power = 0.8
alpha = 0.05
ratio =
float(len(df[df.Fertilizer == "B"]) )/
len(df[df.Fertilizer == "A"])
```

Sample size calculation example

```
from statsmodels.stats import power as pwr

analysis = pwr.TTestIndPower()

ssresult = analysis.solve_power(
effect_size=effect_size,
power=power,
alpha=alpha,
ratio=ratio ,
nobs1=None)
print(ssresult)
```

```
25.5245725005
```

Let's practice!

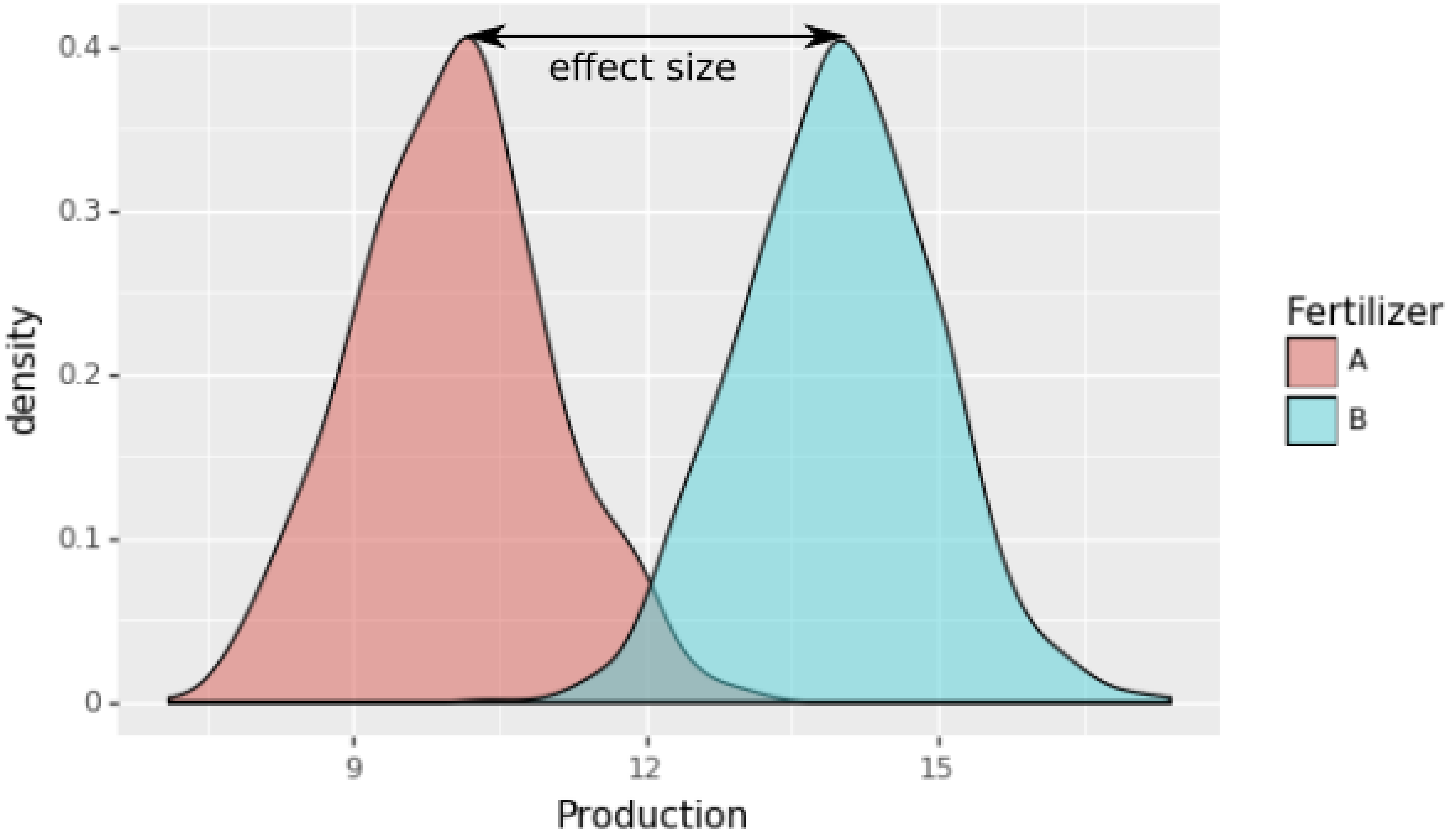
EXPERIMENTAL DESIGN IN PYTHON

Effect size

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

Defining effect size



Effect size vs. significance

Significance

- How sure we are that effect exists
- *X% confident that fertilizer A is better than fertilizer B*

Effect size

- How much difference that effect makes
- *Yields with fertilizer A are Y higher than yields with fertilizer B*

Measures of effect size

Cohen's d

- Continuous variables in relation to discrete variables
- Normalized differences between the means of two samples

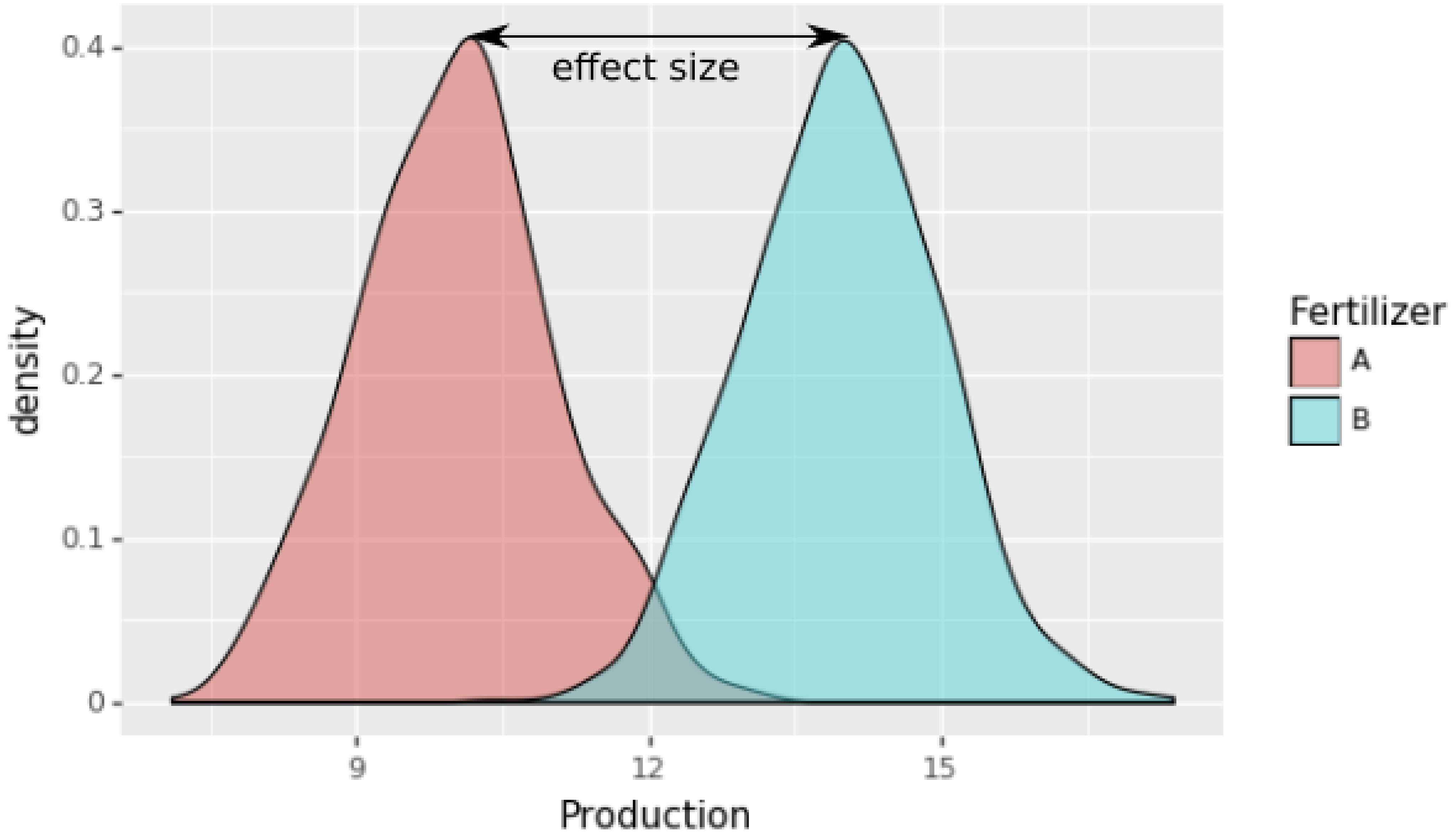
Odds ratio

- For discrete variables
- How much one event is associated with another

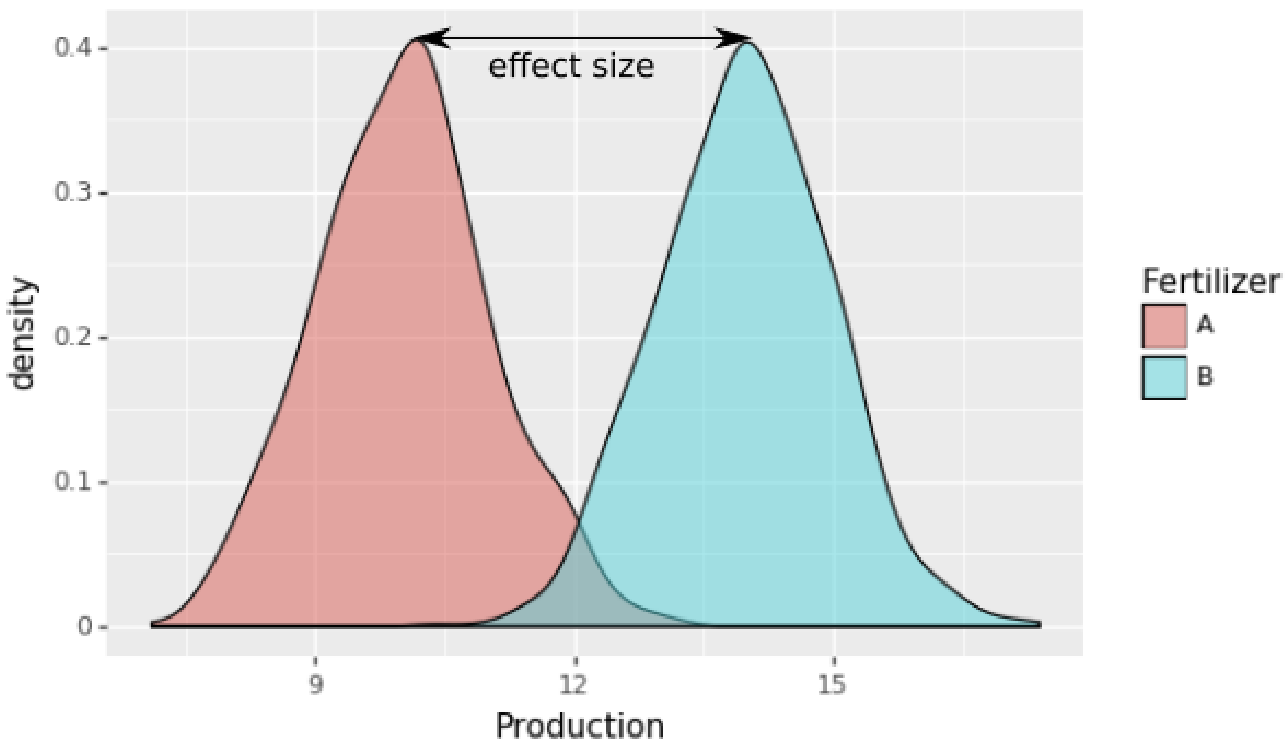
Correlation coefficients

- For continuous variables
- Measures correlation

Effect sizes for t-tests



Calculating Cohen's d



Cohen's d = (M2 - M1) ? SDpooled

```
import math as ma

sampleA = df[df.Fertilizer == "A"].Production
sampleB = df[df.Fertilizer == "B"].Production

diff = abs(sampleA.mean() - sampleB.mean() )
pooledstdev = ma.sqrt(
(sampleA.std()**2 + sampleB.std()**2)/2 )

cohend = diff / pooledstdev
print(cohend)
```

```
4.05052530265279
```

Calculating minimum detectable effect size

Assumptions

`effect_size` : None

`power` : 0.8 (80% chance of detection)

`alpha` : 0.05 (standard)

`ratio` : 1 (equal sample size per group)

`nobs1` : 100

```
from statsmodels.stats import power as pwr

analysis = pwr.TTestIndPower()

esresult = analysis.solve_power(
effect_size=None,
power=power,
alpha=alpha,
ratio=ratio ,
nobs1=nobs1 )
print(esresult)
```

```
0.398139117391
```

Effect size for Fisher exact test

Metric: Odds ratio

```python
import pandas as pd
print(pd.crosstab(df.Coin,df.Flip))
```

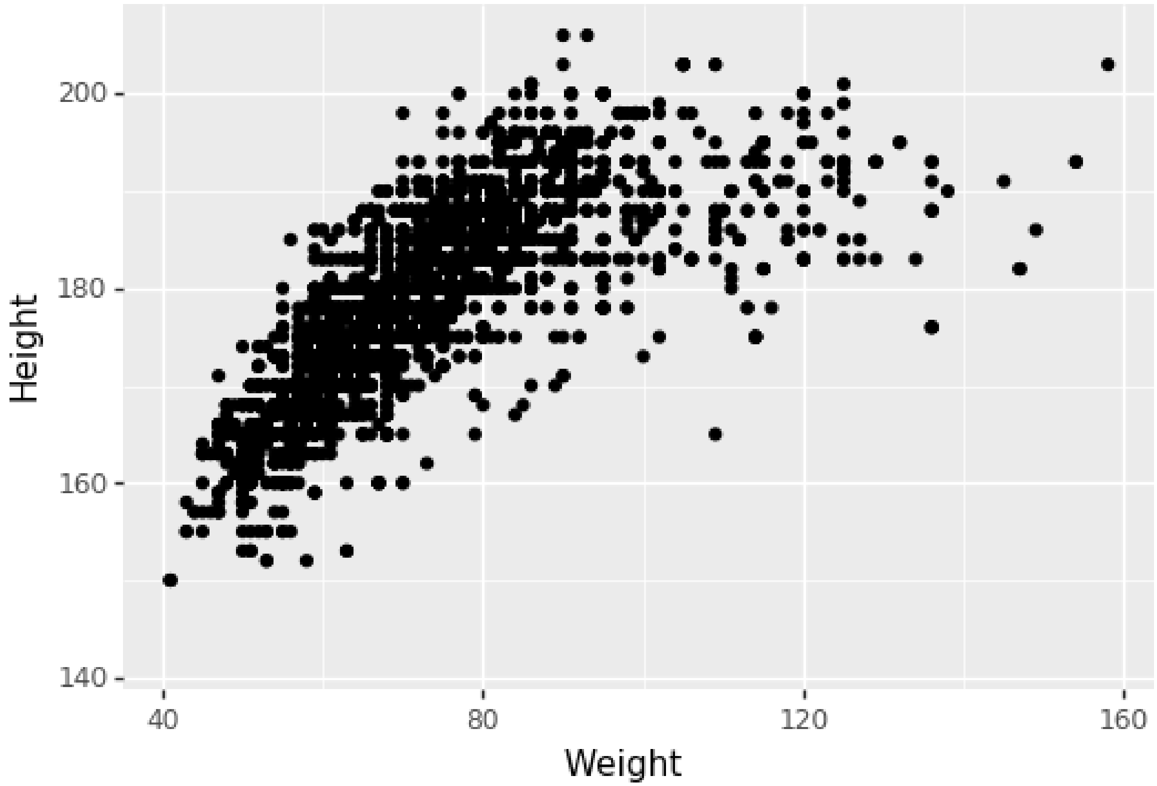
Flip	heads	tails
Coin		
1	22	8
2	17	13

```python
from scipy import stats
chi = stats.fisher_exact(
table, alternative='two-sided')
print(round(chi[0],1))
```

```
2.1
```

Effect size for Pearson correlation

Example



Metric: Pearson correlation coefficient (r)

- Perfect correlation at r = 1

```python
from scipy import stats

pearson = stats.pearsonr(
df.Weight, df.Height)

print(pearson[0])
```

```
0.7922545330545416
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Power

EXPERIMENTAL DESIGN IN PYTHON

Luke Hayden
Instructor

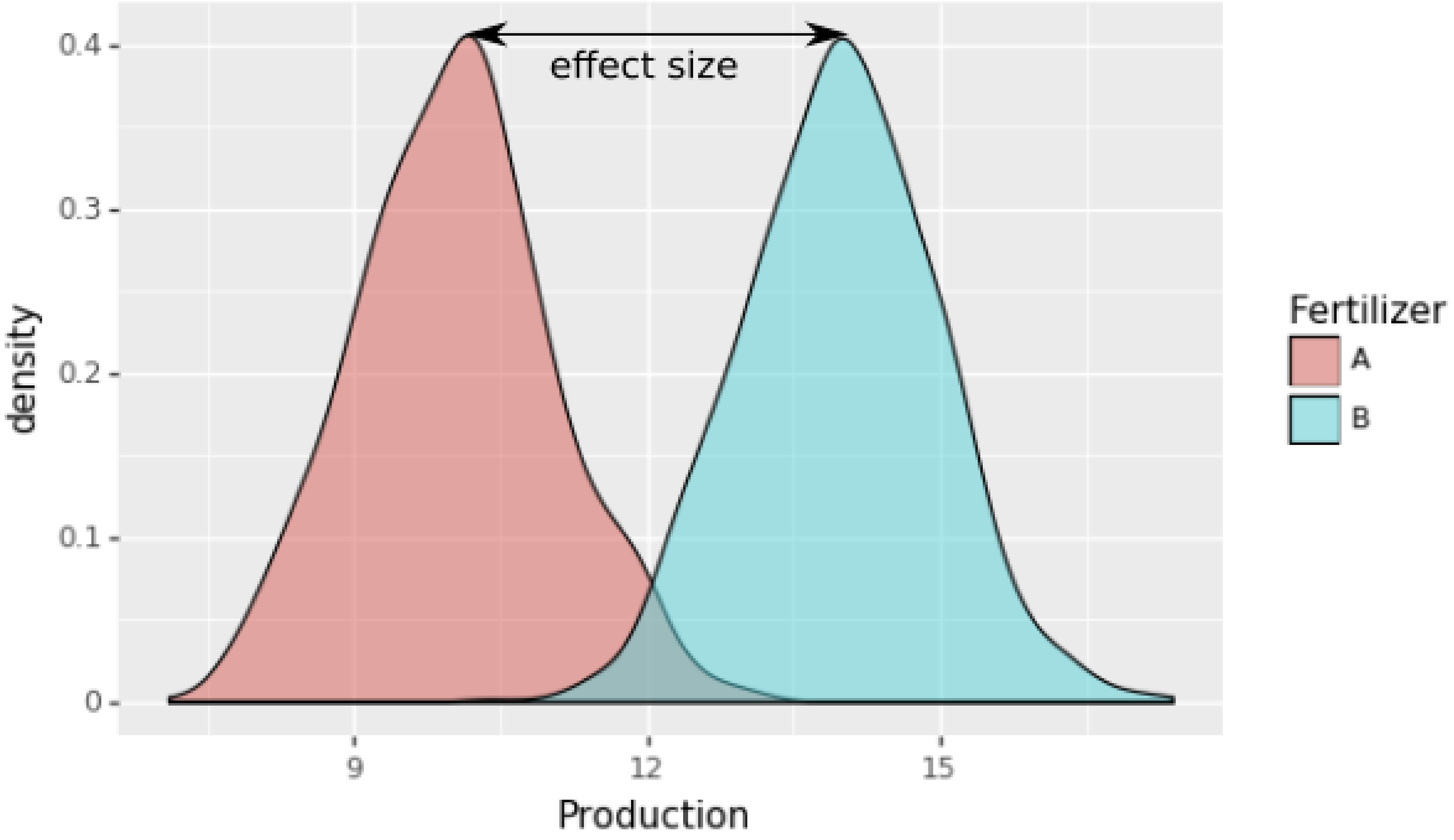
Defining statistical power

- Probability of detecting an effect
- Increase power, decrease chance of type II errors

Relationship to other factors

- Larger effect size, increase power
- Larger sample size, increase power

Calculating power



Calculating power

Assumptions

`effect_size` : 0.8 (large)

`power` : None

`alpha` : 0.05 (standard)

`ratio` : 1 (balanced design)

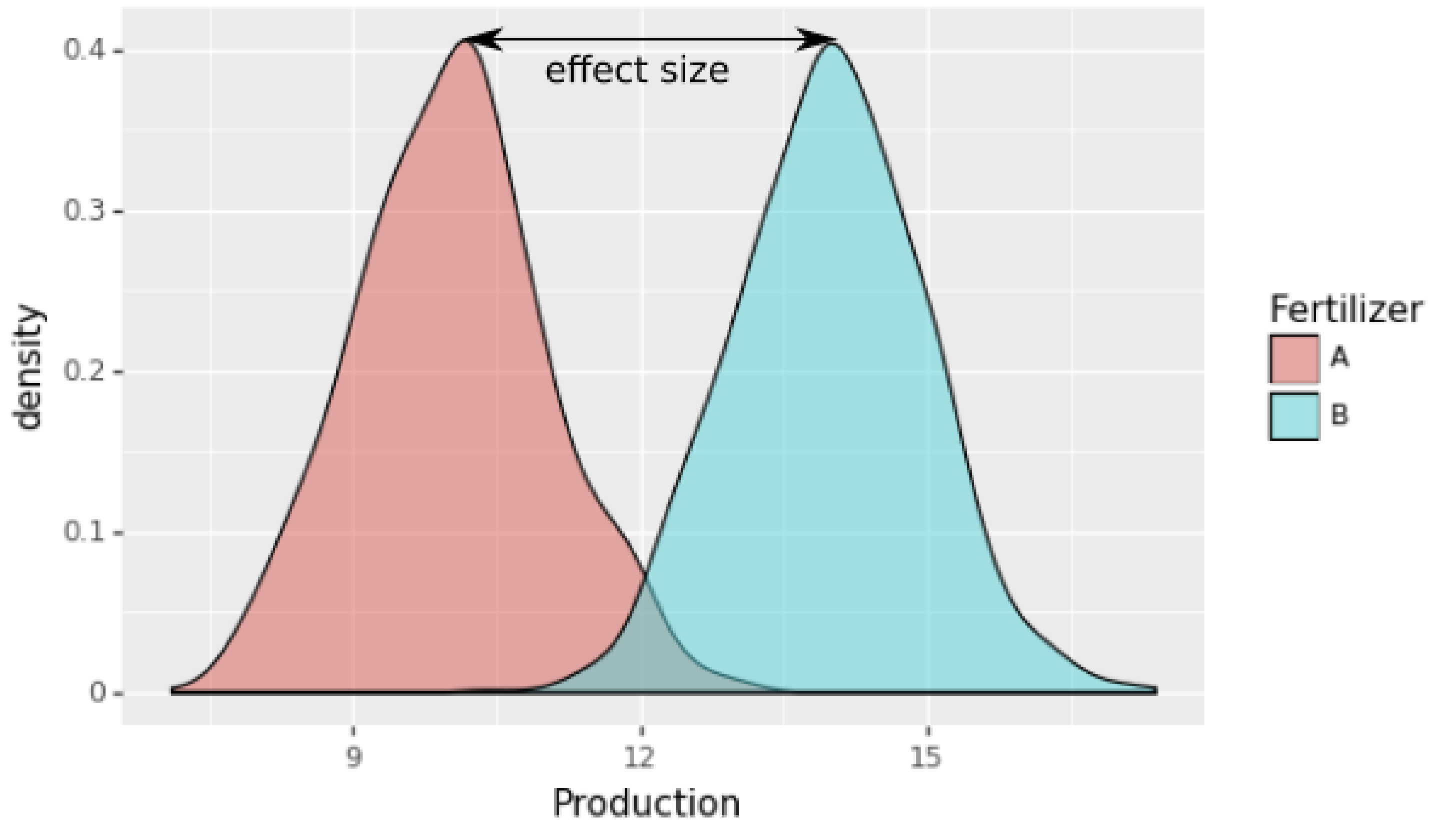
`nobs1` : 100

```
from statsmodels.stats import power as pwr

analysis = pwr.TTestIndPower()

pwresult = analysis.solve_power(
effect_size=effect_size,
power=None,
alpha=alpha,
ratio=ratio ,
nobs1=nobs1 )
print(pwresult)
```

```
0.9998783661018764
```

Calculating power

```
0.9998783661018764
```



Interpretation

- Almost certain of detecting an effect of this size

Dealing with uncertainty

Hypothesis tests

- Estimate likelihoods
- Can't give absolute certainty

Power analysis

- Estimates the strength of answers

Drawing conclusions

Interpreting tests

- In context of power analyses

Possibility of type II errors

- Negative test result & high power: *true negative*
- Negative test result & low power: *possible false negative*

Type I & II errors in context

Find balance

- More power: maybe more risk of type I errors

Domain knowledge

- Make reasonable assumptions

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON