pyAB Documentation

Release 0.0.1

Aditya Varma Kalidindi

Jun 17, 2020

Contents

pyAB is a Python package for Bayesian & Frequentist A/B Testing.

CHAPTER 1

Features:

Bayesian A/B Test:

- Conduct quick experiments to check for winning variant with additional prior information (Beta Distribution parameters).
- Try different evaluation metrics (Uplift Ratio, Uplift Difference & Uplift Percent Gain) & vary number of mcmc simulations.
- Visualize & inspect Uplift Density & Cumulative Density distributions.

Frequentist A/B Test:

- Conduct quick experiments to check for winning variant using two sample proportion test (Statistical significance).
- Estimate required sample size per variant to reach provided Type-II error rate.
- Visualize & inspect power curve for varying alternative proportions.

1.1 Installation

Best way to install pyAB is through pip

```
pip install pyAB
```

To install from source, use the following Github link

```
git clone https://github.com/AdiVarma27/pyAB.git
cd pyAB
python setup.py install
```

1.1.1 Dependencies

pyAB has the following dependencies:

- numpy
- matplotlib
- seaborn
- scipy
- statsmodels

1.2 Quick Start

1.2.1 Bayesian A/B Test

Let us assume we have two Banner Ads and want to run an AB Test to decide on the final version. We run the test and collect 1000 samples per version. We observe 100 and 120 clicks for version-A & Version-B respectively (**10 % & 12.5 % Click-through-rates**). From our previous experience, we know that the average Click-through-rate for our previous Ads was around 12 %.

We first need to import `ABTestBayesian` class and provide prior clicks `success_prior` and prior impressions `trials_prior`. Then, call the `conduct_experiment` method with successful clicks and impressions per version.

For `uplift_method`, there are three metrics to choose from are `'uplift_ratio'`, `'uplift_percent'` & `'uplift_difference'`. We also choose mcmc `num_simulations`, which samples from Uplift Probability Density function.

```
# import Bayesian class
from pyab.experiments import ABTestBayesian

# provide beta priors
ad_experiment_bayesian = ABTestBayesian(success_prior=120, trials_prior=1000)

# conduct experiment with two variants successes and trials, along with uplift method␣
↪and number of simulations
ad_experiment_bayesian.conduct_experiment(success_null=100, trials_null=1000,
                                          success_alt=125, trials_alt=1000,
                                          uplift_method='uplift_ratio', num_
↪simulations=1000)
```

Bayesian A/B test results can extremely useful to **understand & communicate test results** with other stakeholders and answers the main business question: **Which version works the best ?**

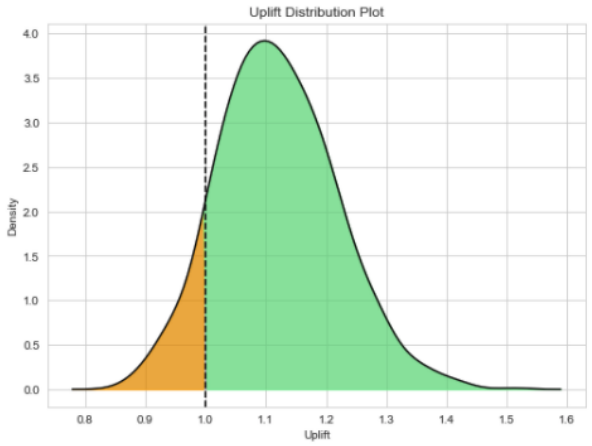
Output:

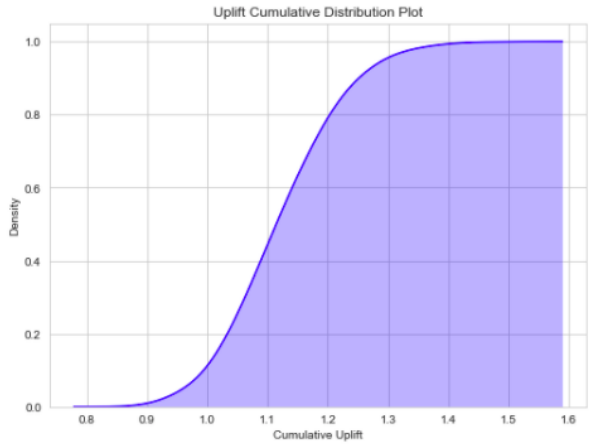
```
pyAB Summary
============

Test Parameters
_______________

Variant A: Successful Trials 100, Sample Size 1000
Variant B: Successful Trials 125, Sample Size 1000
Prior: Successful Trials 120, Sample Size 1000

Test Results
____________
```

(continues on next page)

(continued from previous page)

```
Evaluation Metric: uplift_ratio
Number of mcmc simulations: 1000

90.33 % simulations show Uplift Ratio above 1.
```





1.2.2 Frequentist A/B Test

Let us now run a Frequentist A/B Test and verify if there is a statistically significant difference between two proportions, provided the sample sizes and Type-I Error rate. From above, we know the performance of version-A & version-B **(10 % & 12.5 % Click-through-rates)**, for 1000 impressions of each version.

We first need to import `ABTestFrequentist` class and provide type of alternative hypothesis `alt_hypothesis`, `'one_tailed'` or `'two_tailed'` & Type-I error rate `alpha` (default = 0.05). Then, we call the `conduct_experiment` method with successful clicks and impressions per version.

This traditional methodology might be **slightly tricky to communicate**, and **Type-I & Type-II error rates** need to be accounted for, unlike Bayesian methods.

```
# import Frequentist class
from pyab.experiments import ABTestFrequentist

# provide significance rate and type of test
ad_experiment_freq = ABTestFrequentist(alpha=0.05, alt_hypothesis='one_tailed')

# conduct experiment with two variants successes and trials, returns stat & pvalue
stat, pvalue = ad_experiment_freq.conduct_experiment(success_null=100, trials_
↪null=1000,
                                     success_alt=125, trials_alt=1000)
```

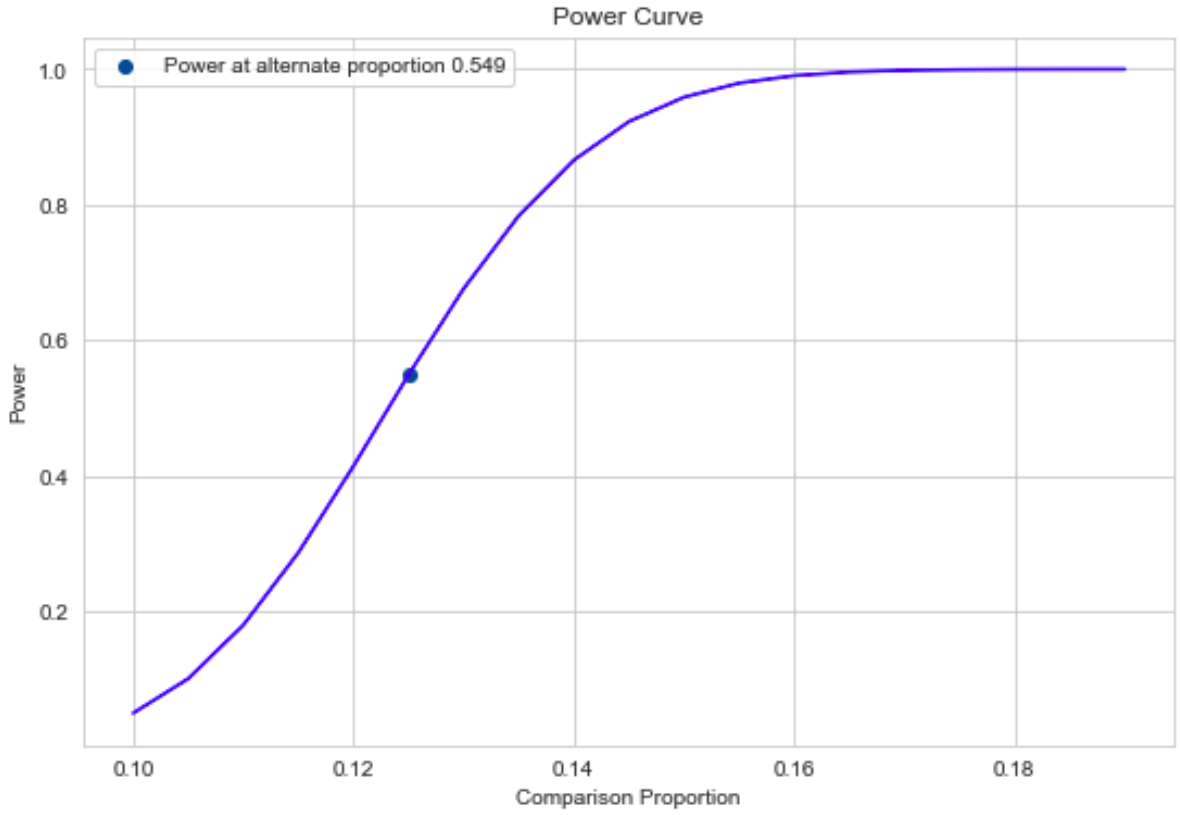
Output:

```
pyAB Summary
============


Test Parameters
_______________
```

(continues on next page)

(continued from previous page)

```
Variant A: Success Rate 0.1, Sample Size 1000
Variant B: Success Rate 0.125, Sample Size 1000
Type-I Error: 0.05, one_tailed test


Test Results
____________

Test Stat: 1.769
p-value: 0.038
Type-II Error: 0.451
Power: 0.549

There is a statistically significant difference in proportions of two variants.
```



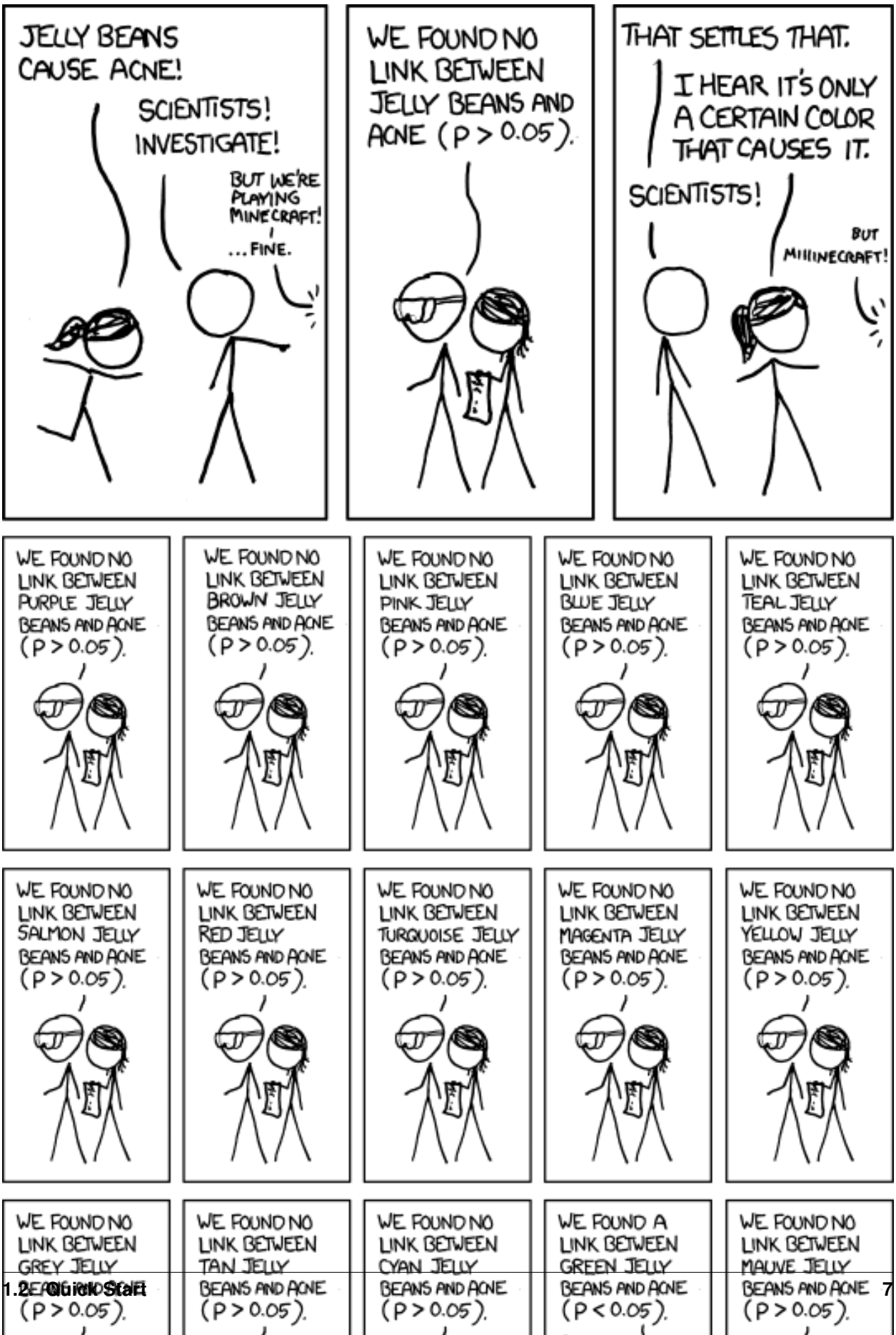
Given that the current Type-II error is 0.451 at 1000 samples per variant, we can find out **required sample size per variant** to reach Type-II error of 0.1.

```
# required sample size per variant for given beta
ad_experiment.get_sample_size(beta=0.1)
```

Output:

```
2729
```

Never misinterpret your Results !


JELLY BEANS CAUSE ACNE!
SCIENTISTS! INVESTIGATE!
BUT WE'RE PLAYING MINECRAFT!
... FINE.
WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE (P > 0.05).
THAT SETTLES THAT.
I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.
SCIENTISTS!
BUT MIIIINECRAFT!
WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN SALMON JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN RED JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TURQUOISE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN MAGENTA JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN YELLOW JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN GREY JELLY
(P > 0.05).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE (P < 0.05).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE (P > 0.05).

1.3 pyAB API

(Version 0.0.1)

1.3.1 Experiments

class pyab.experiments.**ABTestBayesian**(*success_prior*, *trials_prior*)
Bayesian A/B Testing.

Parameters

- **success_prior** (*int*) – Number of successful samples from prior.
- **trials_prior** (*int*) – Number of trials from prior.

calculate_uplift_area()
Calculate Uplift pdf & area beyond threshold.

Returns

- **uplift_distribution** (*ndarray*) – uplift distribution based on chosen uplift method.
- **uplift_area** (*float*) – percentage area above threshold.

conduct_experiment(*success_null*, *trials_null*, *success_alt*, *trials_alt*, *uplift_method='uplift_percent'*, *num_simulations=1000*)
Conduct experiment & generate uplift distributions.

Parameters

- **success_null** (*int*) – Number of successful samples for variant-a.
- **trials_null** (*int*) – Number of trials for variant-a.
- **success_alt** (*int*) – Number of successful samples for variant-b.
- **trials_alt** (*int*) – Number of trials for variant-b.
- **num_simulations** (*int*) – Number of mcmc simulations.
- **uplift_method** (*str, default = 'uplift_percent'*) – Uplift evaluation metric.
  - **'uplift_percent':** percent uplift gain from variant-a to variant-b
  - **'uplift_ratio':** uplift ratio of variant-b & variant-a
  - **'uplift_difference':** uplift difference between variant-b & variant-a

plot_uplift_distributions(*figsize=(18, 6)*)
Plot uplift pdf & cdf for provided experiment parameters.

Parameters **figsize** (*tuple, default = (18, 6)*) – matplotlib plot size.

print_bayesian_results()
Print Bayesian Experiment Results

class pyab.experiments.**ABTestFrequentist**(*alpha=0.05*, *alt_hypothesis='one_tailed'*)
Frequentist A/B Testing aka Two sample proportion test.

Parameters

- **alpha** (*float, default = 0.05*) – Significane level or Type 1 error rate.

- **alt_hypothesis** (*str, default = 'one_tailed'*) – One or two tailed hypothesis test.
  - **'one_tailed':** one tailed hypothesis test
  - **'two_tailed':** two tailed hypothesis test

calculate_power(*stat*)
Calculate power (1-beta) at given test statistics.

Parameters **stat** (*float*) – z or t test statistic.

Returns **1 - beta** – power at given test statistic.

Return type float

calculate_stat(*prop_alt*)
Calculate test statistic with current experiment parameters.

Parameters **prop_alt** (*float*) – alternate hypothesis proportion.

Returns **stat** – z or t statistic.

Return type float

conduct_experiment(*success_null*, *trials_null*, *success_alt*, *trials_alt*)
Conduct experiment & generate power curve with provided parameters.

Parameters

- **success_null** (*int*) – number of successful clicks or successful events (Version-A).
- **trials_null** (*int*) – number of impressions or events (Version-A).
- **success_alt** (*int*) – number of successful clicks or successful events (Version-B).
- **trials_alt** (*int*) – number of impressions or events (Version-B).

Returns

- **stat** (*float*) – z or t statistic.
- **pvalue** (*float*) – probability of obtaining results atleast as extreme as the results actually observed during the test.

get_sample_size(*beta=0.1*)
Calculate required sample size per group to obtain provided beta.

Parameters **beta** (*float*) – Type 2 error rate.

Returns **n** – sample size per group.

Return type int

plot_power_curve(*figsize=(9, 6)*)
Plot power curve for provided experiment parameters.

Parameters **figsize** (*tuple, default = (9,6)*) – matplotlib plot size.

print_freq_results()
Print Frequentist Experiment Results

1.4 Contributing to pyAB

Welcome! pyAB is a community project and your contribution is important to the packages usability and success.

1.4.1 Code of Conduct

Contributors and participants of pyAB are expected to follow guidelines provided by Python Community Code of Conduct.

1.4.2 General Guidelines

- For issues, please submit them to the issue tracker. If you can provide features, improvements or anything else the world of AB Testing has to offer, your contributions are highly appreciated.
- Code in master branch should reflect the latest version. Create a pull request and add your merge request to the `dev-pyab` branch.
- Please follow pep8 for coding conventions. For quick information about Git, visit https://rogerdudler.github.io/git-guide/.

1.5 Release History

1.5.1 Version 0.0.1

- ABTestBayesian, ABTestFrequentist classes in experiments module.
- Base utility functions, test cases.

CHAPTER 2

Usage:

CHAPTER 3

Bayesian A/B Test

Let us assume we have two Banner Ads and want to run an AB Test to decide on the final version. We run the test and collect 1000 samples per version. We observe 100 and 120 clicks for version-A & Version-B respectively **(10 % & 12.5 % Click-through-rates)**. From our previous experience, we know that the average Click-through-rate for our previous Ads was around 12 %.

We first need to import `ABTestBayesian` class and provide prior clicks `success_prior` and prior impressions `trials_prior`. Then, call the `conduct_experiment` method with successful clicks and impressions per version.

For `uplift_method`, there are three metrics to choose from are `'uplift_ratio'`, `'uplift_percent'` & `'uplift_difference'`. We also choose mcmc `num_simulations`, which samples from Uplift Probability Density function.

```
# import Bayesian class
from pyab.experiments import ABTestBayesian

# provide beta priors
ad_experiment_bayesian = ABTestBayesian(success_prior=120, trials_prior=1000)

# conduct experiment with two variants successes and trials, along with uplift method␣
↪and number of simulations
ad_experiment_bayesian.conduct_experiment(success_null=100, trials_null=1000,
                                          success_alt=125, trials_alt=1000,
                                          uplift_method='uplift_ratio', num_
↪simulations=1000)
```

Bayesian A/B test results can extremely useful to **understand & communicate test results** with other stakeholders and answers the main business question: **Which version works the best ?**

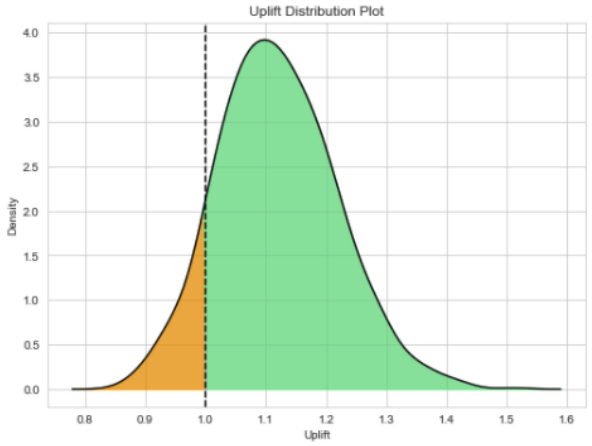
Output:

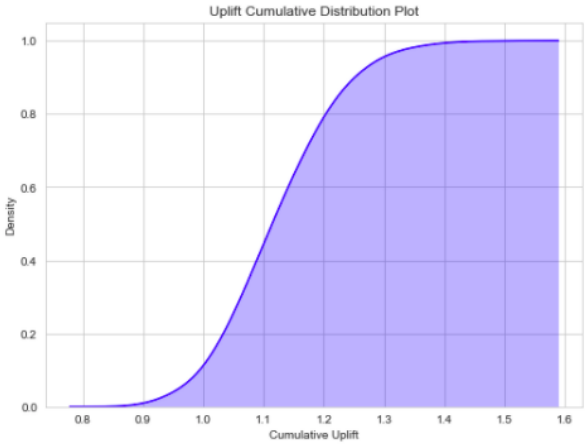
```
pyAB Summary
============

Test Parameters
```

(continues on next page)

(continued from previous page)

```
_________________

Variant A: Successful Trials 100, Sample Size 1000
Variant B: Successful Trials 125, Sample Size 1000
Prior: Successful Trials 120, Sample Size 1000

Test Results
_____________

Evaluation Metric: uplift_ratio
Number of mcmc simulations: 1000

90.33 % simulations show Uplift Ratio above 1.
```





CHAPTER 4

Frequentist A/B Test

Let us now run a Frequentist A/B Test and verify if there is a significant difference between two proportions provided the sample sizes and Type-I Error rate. From above, we know the performance of version-A & version-B **(10 % & 12.5 % Click-through-rates)**, for 1000 impressions of each version.

We first need to import `ABTestFrequentist` class and provide type of alternative hypothesis `alt_hypothesis`, `'one_tailed'` or `'two_tailed'` & Type-I error rate `alpha` (default = 0.05). Then, we call the `conduct_experiment` method with successful clicks and impressions per version.

This traditional methodology might be **slightly tricky to communicate**, and **Type-I & Type-II error rates** need to be accounted for, unlike Bayesian methods.

```
# import Frequentist class
from pyab.experiments import ABTestFrequentist

# provide significance rate and type of test
ad_experiment_freq = ABTestFrequentist(alpha=0.05, alt_hypothesis='one_tailed')

# conduct experiment with two variants successes and trials, returns stat & pvalue
stat, pvalue = ad_experiment_freq.conduct_experiment(success_null=100, trials_
↪null=1000,
                                    success_alt=125, trials_alt=1000)
```

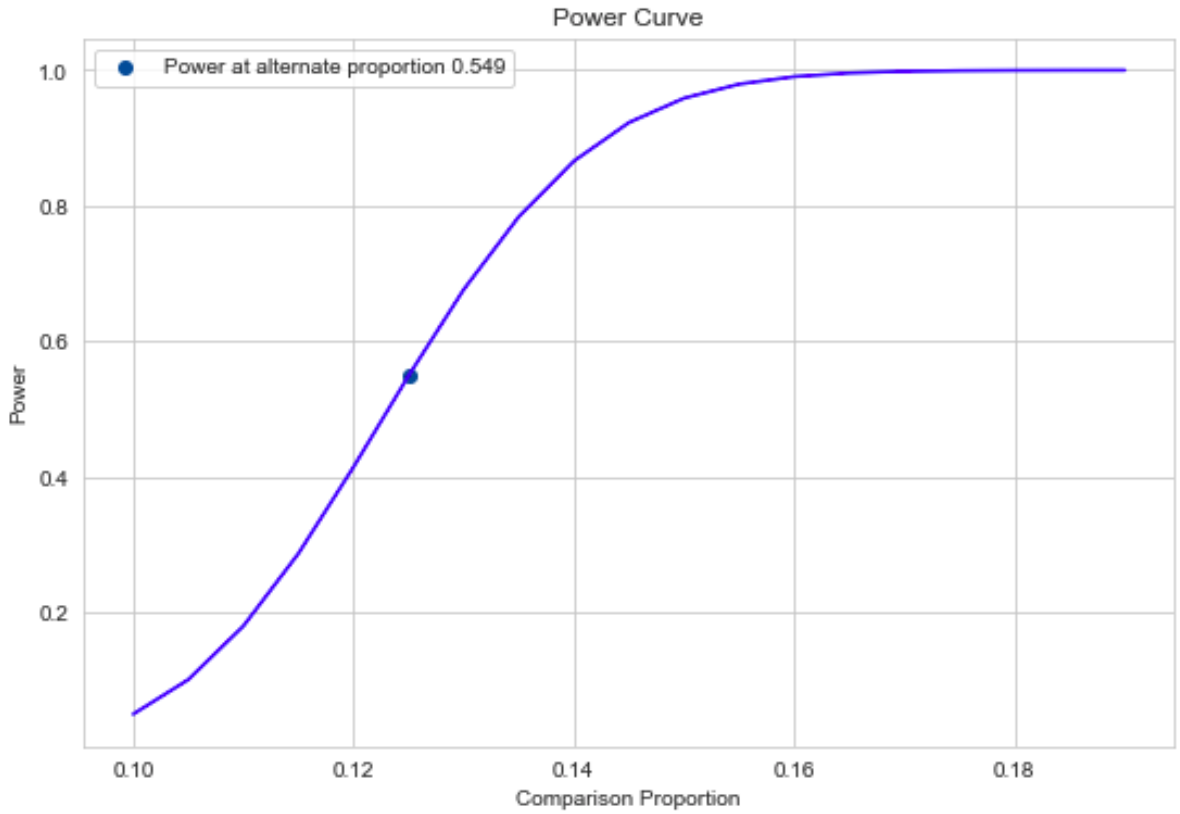
Output:

```
pyAB Summary
============


Test Parameters
_______________

Variant A: Success Rate 0.1, Sample Size 1000
Variant B: Success Rate 0.125, Sample Size 1000
Type-I Error: 0.05, one_tailed test
```

(continues on next page)

(continued from previous page)

```
Test Results
————————————

Test Stat: 1.769
p-value: 0.038
Type-II Error: 0.451
Power: 0.549

There is a statistically significant difference in proportions of two variants.
```



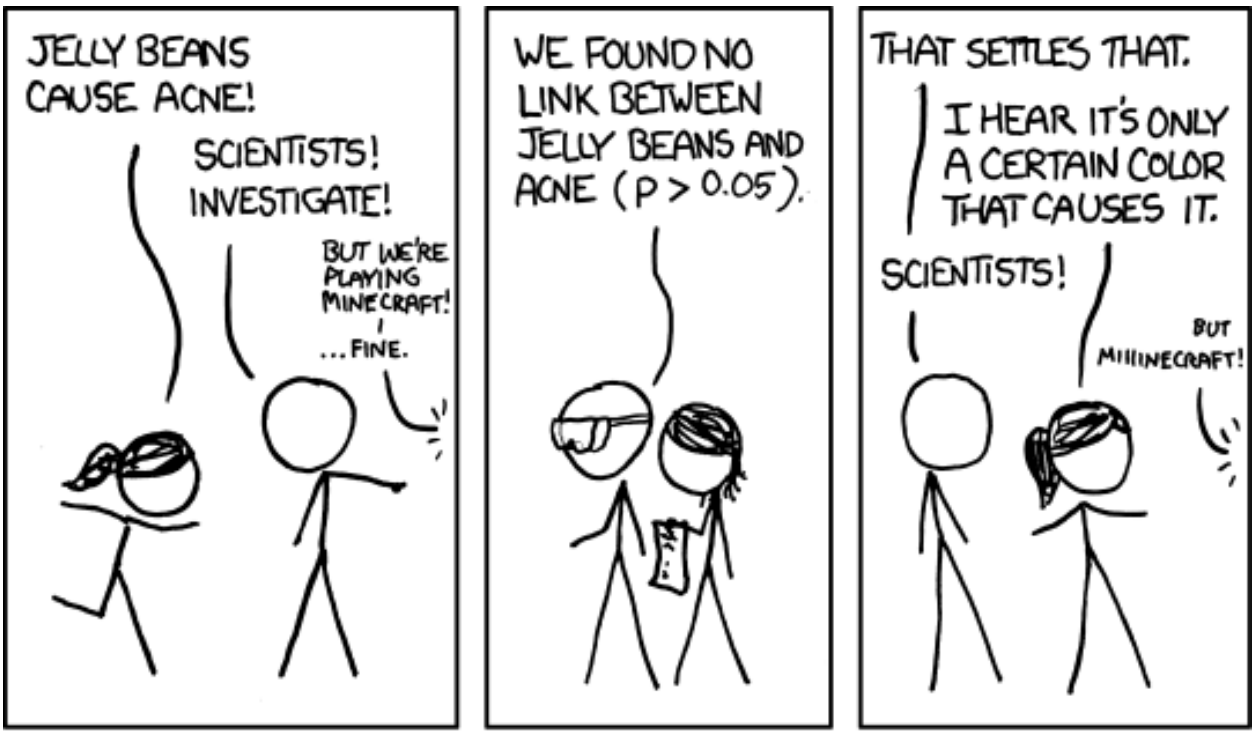
Given that the current Type-II error is 0.451 at 1000 samples per variant, we can find out **required sample size per variant** to reach Type-II error of 0.1.

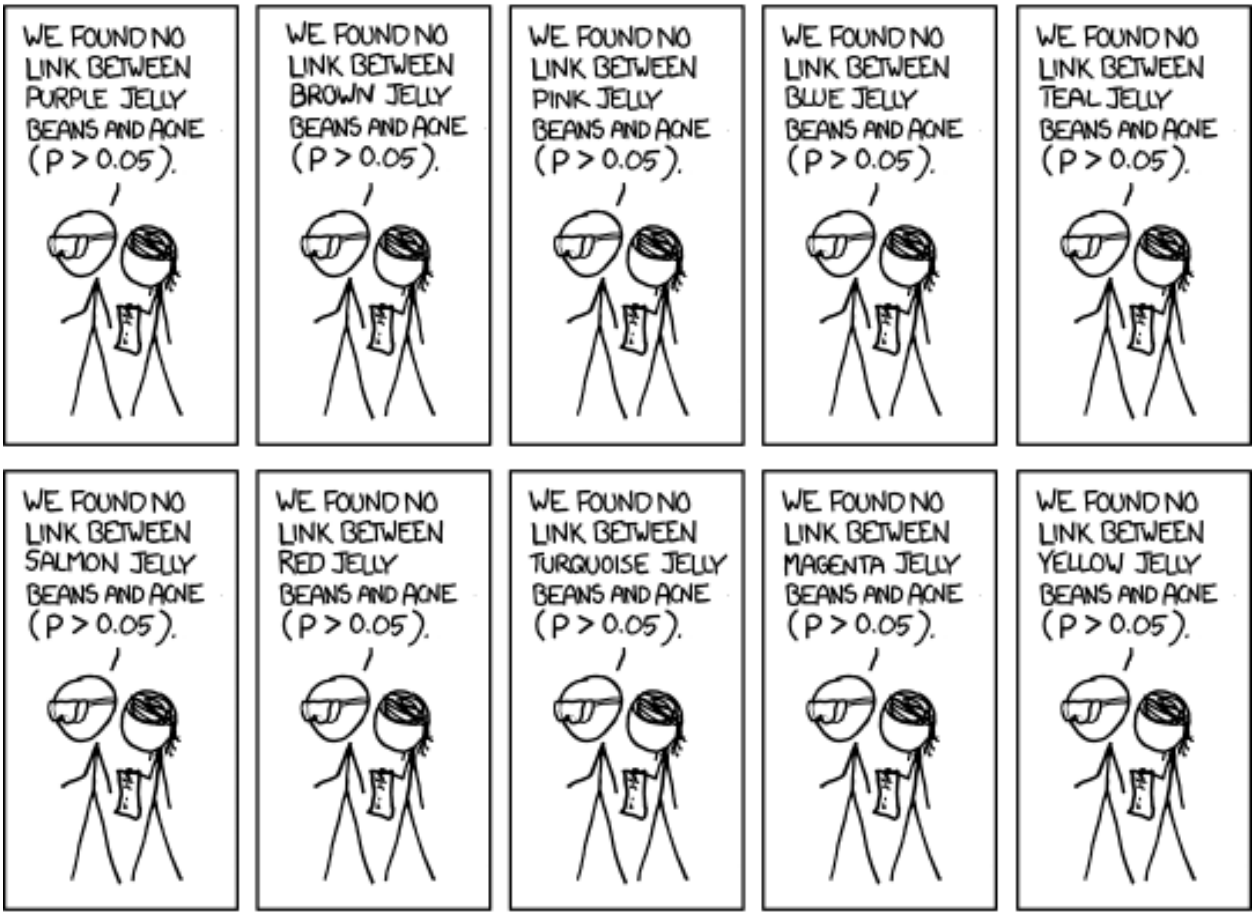
```
# required sample size per variant for given beta
ad_experiment.get_sample_size(beta=0.1)
```

Output:

```
2729
```

Never misinterpret your Results !


JELLY BEANS CAUSE ACNE!
SCIENTISTS! INVESTIGATE!
BUT WE'RE PLAYING MINECRAFT!
... FINE.
WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE (P > 0.05).
THAT SETTLES THAT.
I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.
SCIENTISTS!
BUT MIIIINECRAFT!


WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN SALMON JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN RED JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TURQUOISE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN MAGENTA JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN YELLOW JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN GREY JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE (P < 0.05).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE (P > 0.05).

Index

A

C

G

P