Analyzing the A/B test results

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

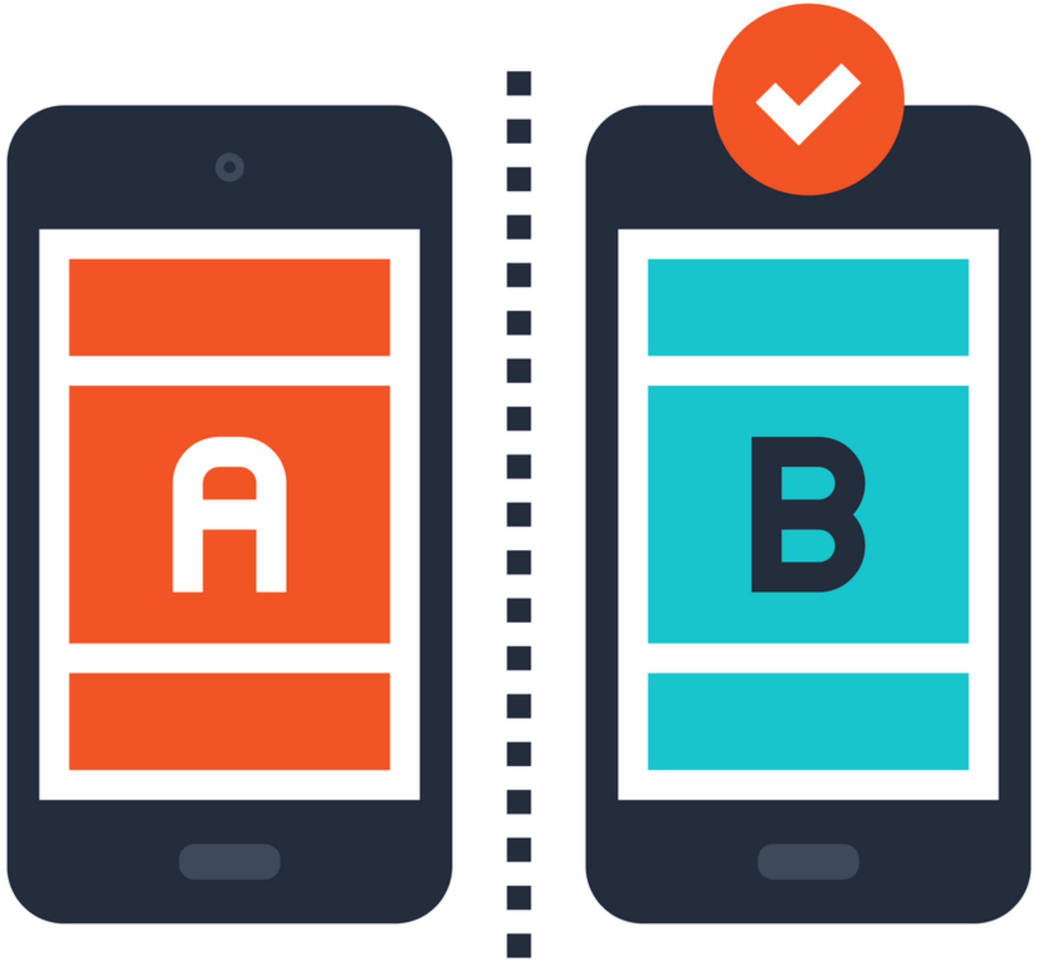
Ryan Grossman
Data Scientist, EDO

Analyzing A/B test results

- How to analyze an A/B test
- Further topics in A/B testing

Evaluating our paywall test

- **So far:** Run our test for the specified amount of time
- **Next:** Compare the two groups' purchase rates



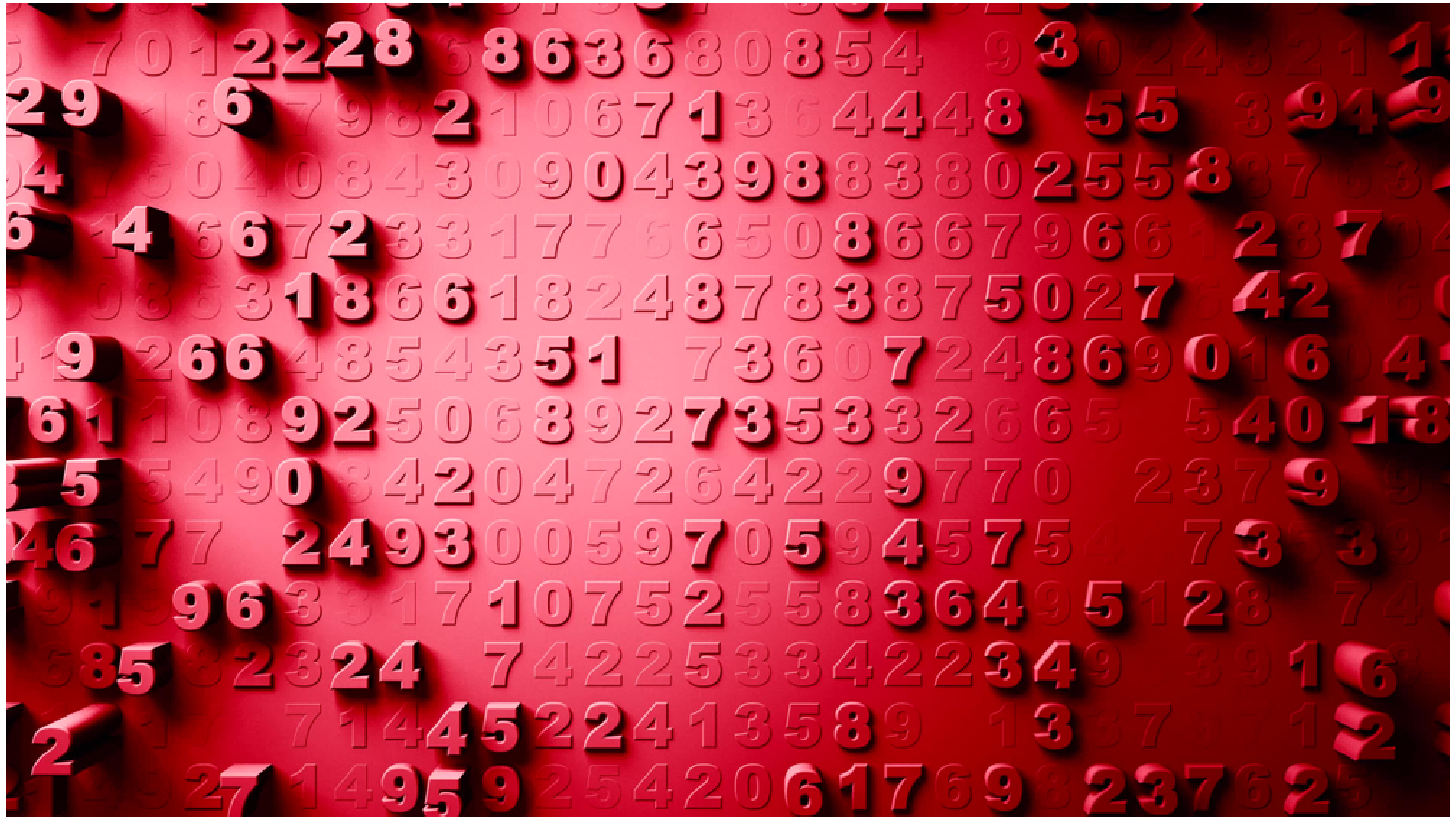
Test results data

```
# Demographic information for our test groups
test_demographics = pd.read_csv('test_demographics.csv`)


# results for our A/B test
# group column: 'c' for control | 'v' for variant
test_results = pd.read_csv('ab_test_results.csv')
test_results.head(n=5)
```

```
uid                  date     purchase    sku     price     group
90554036.0    2018-02-27 14:22:12     0     NaN      NaN      C
90554036.0    2018-02-28 08:58:13     0     NaN      NaN      C
90554036.0    2018-03-01 09:21:18     0     NaN      NaN      C
90554036.0    2018-03-02 10:14:30     0     NaN      NaN      C
90554036.0    2018-03-03 13:29:45     0     NaN      NaN      C
```

Confirming our test results

- Crucial to validate your test data
  - Does the data look reasonable?
  - Ensure you have a random sample



Are our groups the same size?

```
# Group our data by test vs. control
test_results_grpd = test_results.groupby(
    by=['group'], as_index=False)

# Count the unique users in each group
test_results_grpd.uid.count()
```

```
    group    uid
0     C       48236
1     V       49867
```

Do our groups have similar demographics?

```
# Group our test data by demographic breakout
test_results_demo = test_results.merge(
    test_demo, how='inner', on='uid')
test_results_grpd = test_results_demo.groupby(
    by= ['country','gender', 'device', 'group' ],
    as_index=False)
test_results_grpd.uid.count()
```

```
country    gender    device    group    uid
BRA        F         and       C        5070
BRA        F         and       V        4136
BRA        F         iOS       C        3359
BRA        F         iOS       V        2817
...
```

Test & control group conversion rates

```
# Find the count of payawall viewer and purchases in each group
test_results_summary = test_results_demo.groupby(
    by=['group'], as_index=False
).agg({'purchase': ['count', 'sum']})

# Calculate our paywall conversion rate by group
test_results_summary['conv'] = (test_results_summary.purchase['sum'] /
    test_results_summary.purchase['count'])
test_results_summary
```

```
    group    purchase    conv
count    sum
0    C    48236    1657    0.034351
1    V    49867   2094     0.041984
```

Is the result statistically significant?

- **Statistical Significance:** Are the conversion rates different enough?
  - If *yes* then we reject the null hypothesis
  - Conclude that the paywall's have different effects
  - If *no* then it may just be randomness

p-values

- probability if the Null Hypothesis is true...
- of observing a value as or more extreme...
- than the one we observed
- **Low p-values**
  - represent potentially significant results
  - the observation is unlikely to have happened due to randomness

Interpreting p-values

- Controversial concept in some ways
- **Typically:** accept or reject hypothesis based on the p-value
- Below table shows the general rules of thumb:

p-value	Conclusion
< 0.01	very strong evidence against the Null Hypothesis
0.01 - 0.05	strong evidence against the Null Hypothesis
0.05 - 0.10	very weak evidence against the Null Hypothesis
> 0.1	small to no evidence against the Null Hypothesis

Next steps

1. Confirm our results
2. Explore how to provide useful context for them

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

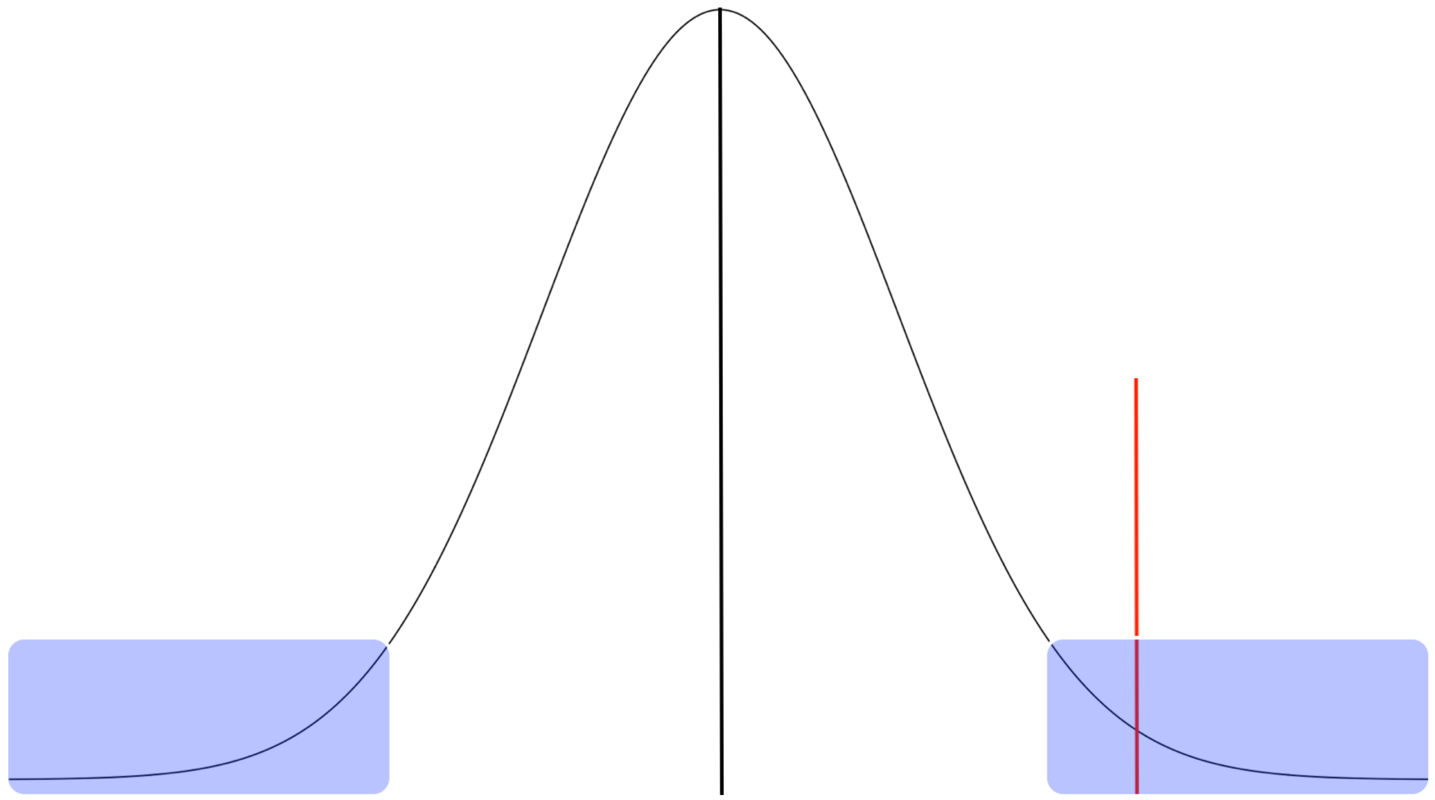
Understanding statistical significance

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Revisiting statistical significance

- Distribution of expected difference between control and test groups *if* the Null Hypothesis true
- **Red line:** The observed difference in conversion rates from our test
- **p-value:** Probability of being as or more extreme than the red line on either side of the distribution



p-value Function

```python
# calculate the p-value from our
# group conversion rates and  group sizes
def get_pvalue(con_conv, test_conv,con_size,  test_size,):
    lift =  - abs(test_conv - con_conv)
    scale_one = con_conv * (1 - con_conv) * (1 / con_size)
    scale_two = test_conv * (1 - test_conv) * (1 / test_size)
    scale_val = (scale_one + scale_two)**0.5
    p_value = 2 * stats.norm.cdf(lift, loc = 0, scale = scale_val )

    return p_value
```

Calculating our p-value

- Observe a small p-value and statistically significant results
- Achieved lift is relatively large

```
# previously calculated quantities
con_conv  = 0.034351 # control group conversion rate
test_conv = 0.041984 # test group conversion rate
con_size  = 48236 # control group size
test_size = 49867 # test group size


# calculate the test p-value
p_value = get_pvalue(_conv, con_size, test_size)
print(p_value)
```

```
4.2572974855869089e-10
```

Finding the power of our test

```
# Calculate our test's power
get_power(test_size, con_conv, test_conv, 0.95)
```

```
0.99999259413722819
```

What is a confidence interval

- Range of values for our estimation rather than single number
- Provides context for our estimation process
- Series of repeated experiments...
  - the calculated intervals will contain the true parameter X% of the time
- The true conversion rate is a fixed quantity, our estimation and the interval are variable

Confidence interval calculation

Confidence Interval Formula

$$\mu \pm \Phi\left(\alpha + \frac{1-\alpha}{2}\right) \times \sigma$$

- Estimated parameter (difference in conversion rates) follows Normal Distribution
- Can estimate the:
  - standard deviation (σ) and...
  - mean (μ) of this distribution
- α: Desired confidence interval width
- Bounds containing X% of the probability around the mean (e.g. 95%) of that distribution

Confidence interval function

```
# Calculate the confidence interval
from scipy import stats
def get_ci(test_conv, con_conv,
    test_size, con_size, ci):

    sd = ((test_conv * (1 - test_conv)) / test_size +
        (con_conv * (1 - con_conv)) / con_size)**0.5
    lift = test_conv - con_conv

    val = stats.norm.isf((1 - ci) / 2)
    lwr_bnd = lift - val * sd
      upr_bnd = lift + val * sd

    return((lwr_bnd, upr_bnd))
```

Calculating confidence intervals

- `test_conv` : test group conversion rate
- `con_conv` : control group conversion rate
- `test_size` : test group observations
- `con_size` : control group observations

```
# Calcualte the conversion rate
get_ci(
    test_conv, con_conv,
    test_size, con_size,
    0.95
)
```

```
(0.00523, 0.0100)
```

- Provides additional context about our results

Next steps

- Adding context to our test results
- Communicating the data through visualizations

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Interpreting your test results

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

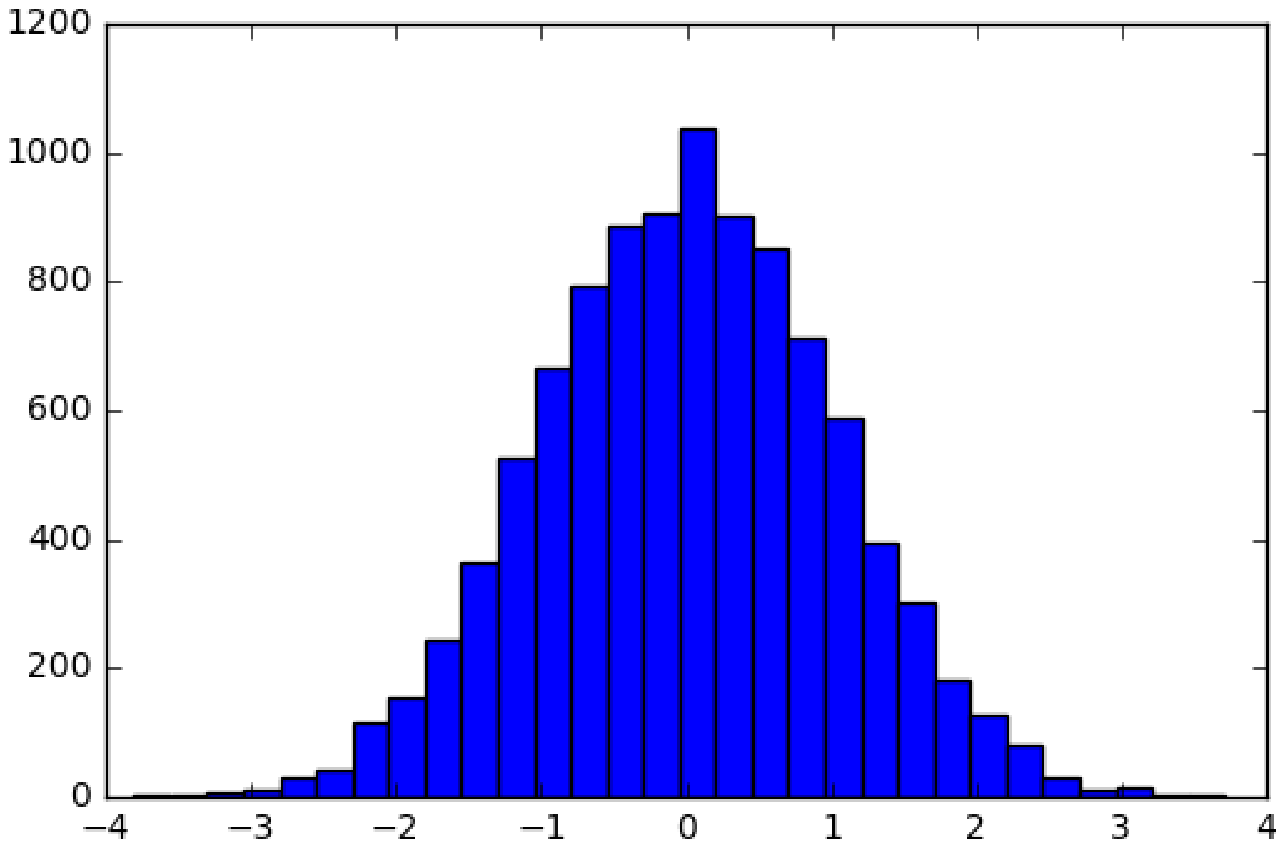
Factors to communicate

	Test Group	Control Group
Sample Size	7030	6970
Run Time	2 Weeks	2 Weeks
Mean	3.12	2.69
Variance	3.20	2.64
Estimated Lift: 0.56 *		
Confidence Intervel 0.56 ± 0.4		

** Significant at the 0.05 Level*

Visualizing your results

- **Histogram:** Bucketed counts of observations across values
- Histogram of centered and scaled conversion rates for users
    - `(conv_rate - mean) / sd`



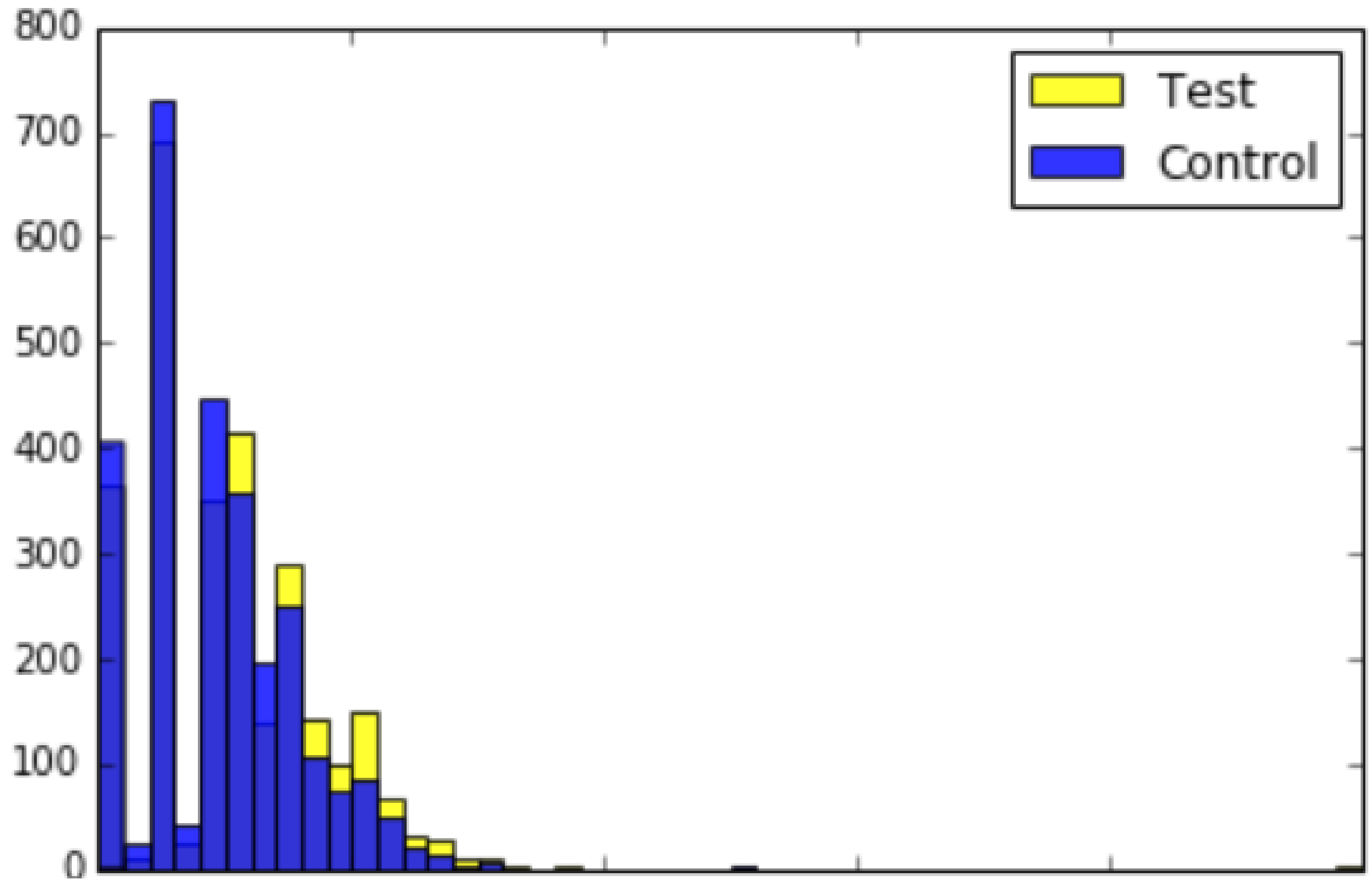
Generating a histogram

```
# Purchase rate grouped by user and test group
results.head(n=10)
```

```
     uid      group    purchase
11128497.0    V          0.000000
11145206.0    V          0.050000
11163353.0    C          0.150000
11215368.0    C          0.000000
11248473.0    C          0.157895
11258429.0    V          0.086957
11271484.0    C          0.071429
11298958.0    V          0.157895
11325422.0    C          0.045455
11340821.0    C          0.040000
```

Generating a histogram

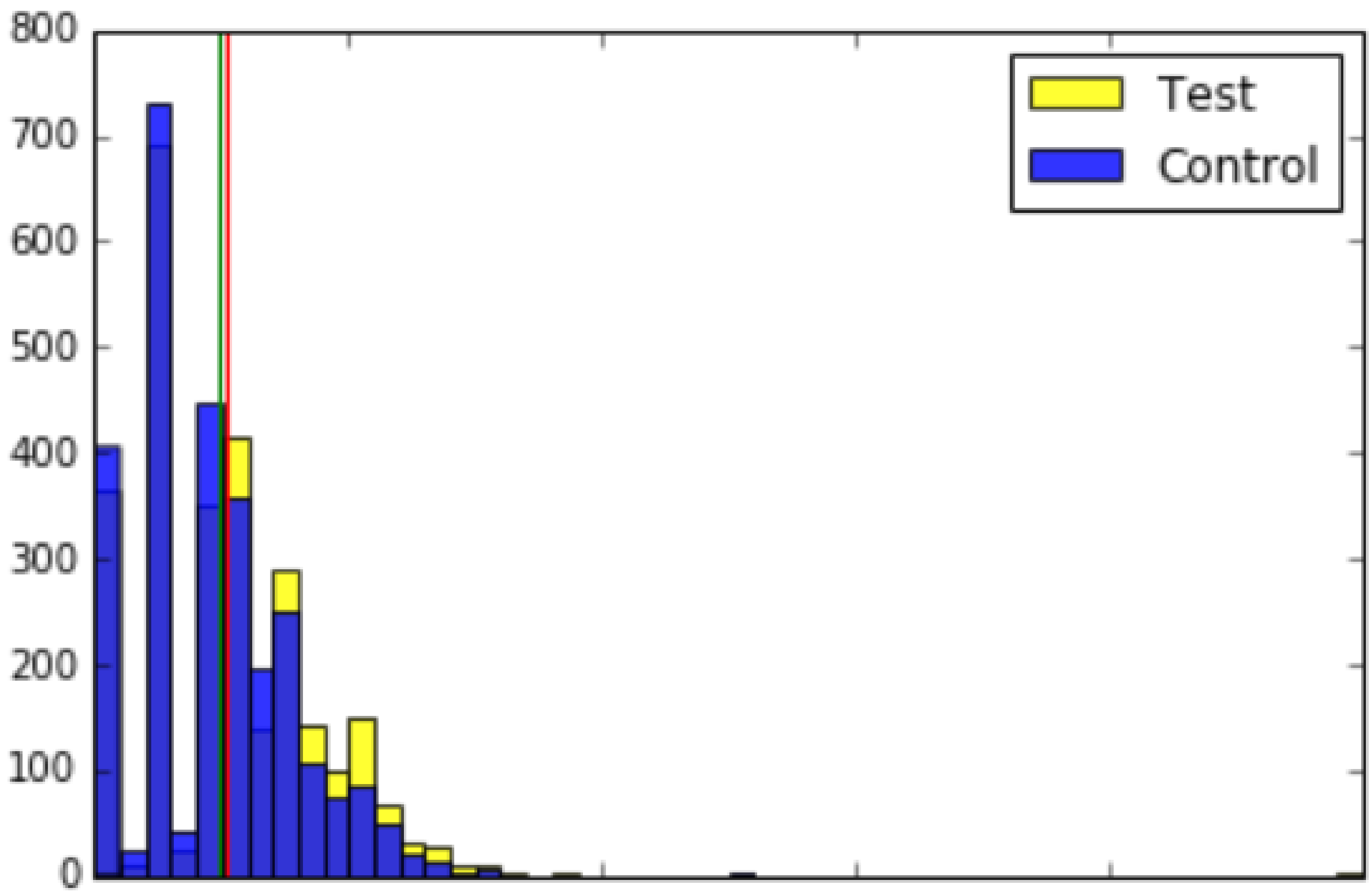
```
# Break out our user groups
var = results[results.group == 'V']
con = results[results.group == 'C']


# plot our conversion rate data for each group
plt.hist(var['purchase'],color = 'yellow',
    alpha = 0.8, bins =50, label = 'Test')
plt.hist(con['purchase'],  color = 'blue',
    alpha = 0.8, bins = 50, label = 'Control')
plt.legend(loc='upper right')
```



Annotating our plot

- `plt.axvline()` : Draw a vertical line of the specified color

```
# Draw annotation lines at the mean values
# for each group
plt.axvline(x = np.mean(results.purchase),
    color = 'red')
plt.axvline(x= np.mean(results.purchase),
    color = 'green')
plt.show()
```



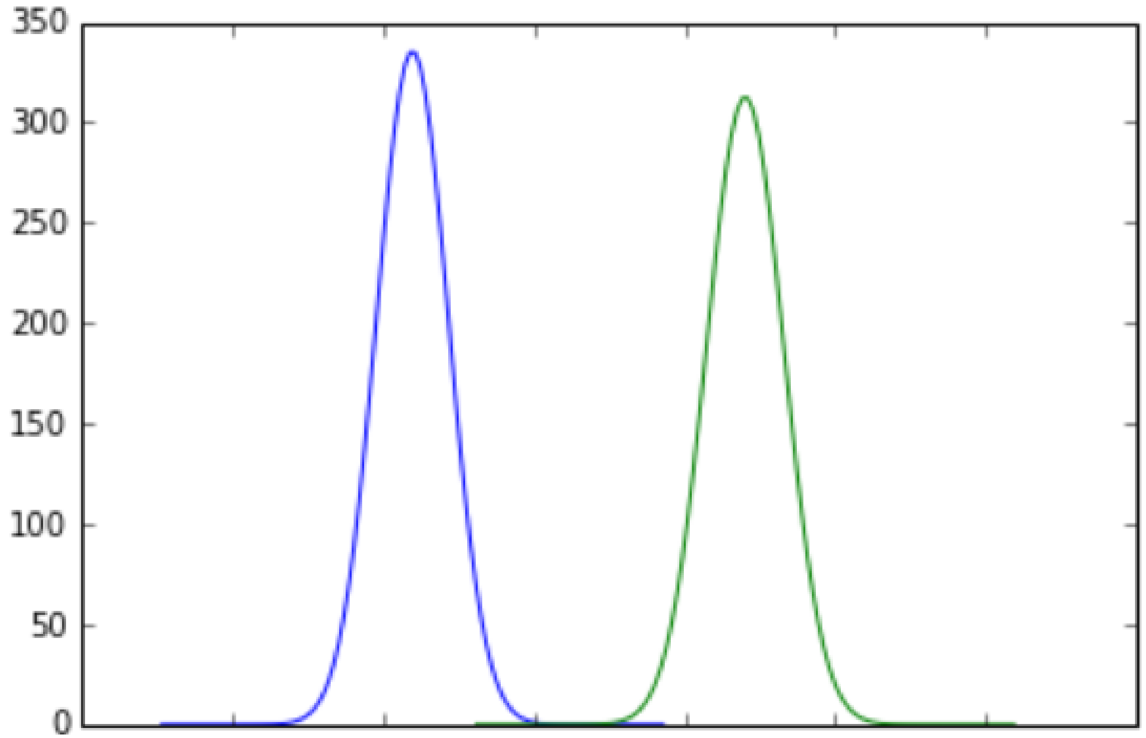
Plotting a distribution

```
# Use our mean values to calculate the variance
mean_con = 0.090965
mean_test = 0.102005
var_con = (mean_con * (1 - mean_con)) / 58583
var_test = (mean_test * (1 - mean_test)) / 56350


# Generate a range of values across the
# distribution from +/- 3 sd around the mean
con_line = np.linspace(-3 * var_con**0.5 +
    mean_con, 3 * var_con**0.5 + mean_con, 100)
test_line = np.linspace(-3 * var_test**0.5 +
    mean_test, 3 * var_test**0.5 + mean_test, 100)
```

Plotting a distribution

```
import mlab from matplotlib

# Plot the probabilities across the
distributioin of conversion rates
plt.plot(con_line,mlab.normpdf(
    con_line, mean_con,var_con**0.5)
)
plt.plot(test_line,mlab.normpdf(
    test_line, mean_test, var_test**0.5)
)
plt.show()
```

- `mlab.normpdf()` : Converts values to probabilities from Normal distribution



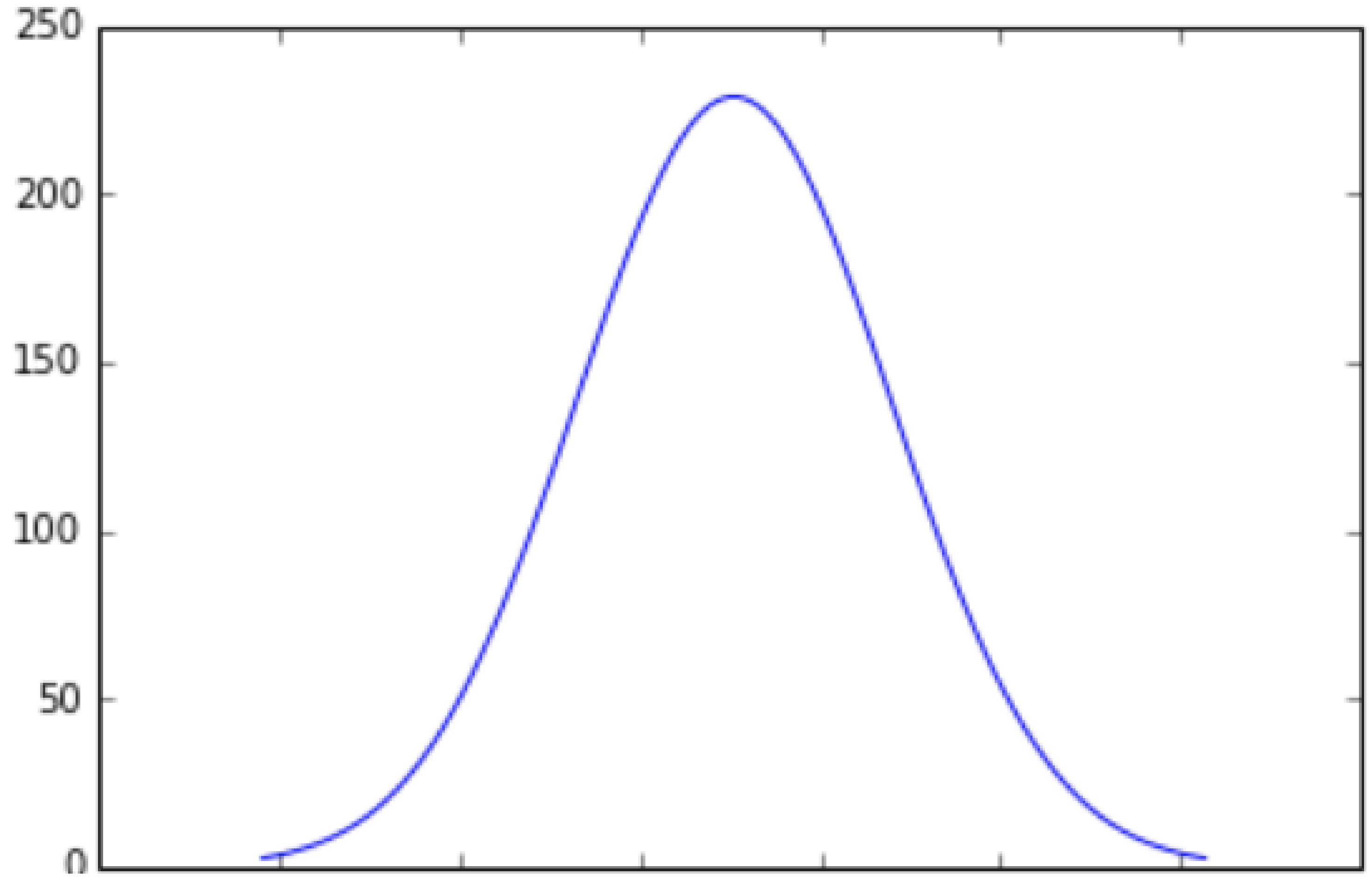
Plotting the difference of conversion rates

- The difference of Normal Distributions is a Normal Distribution
  - **Mean**: Difference of the means
  - **Variance**: Sum of the variances

```
lift = mean_test - mean_control

var = var_test + var_control
```

Plotting the difference of conversion rates

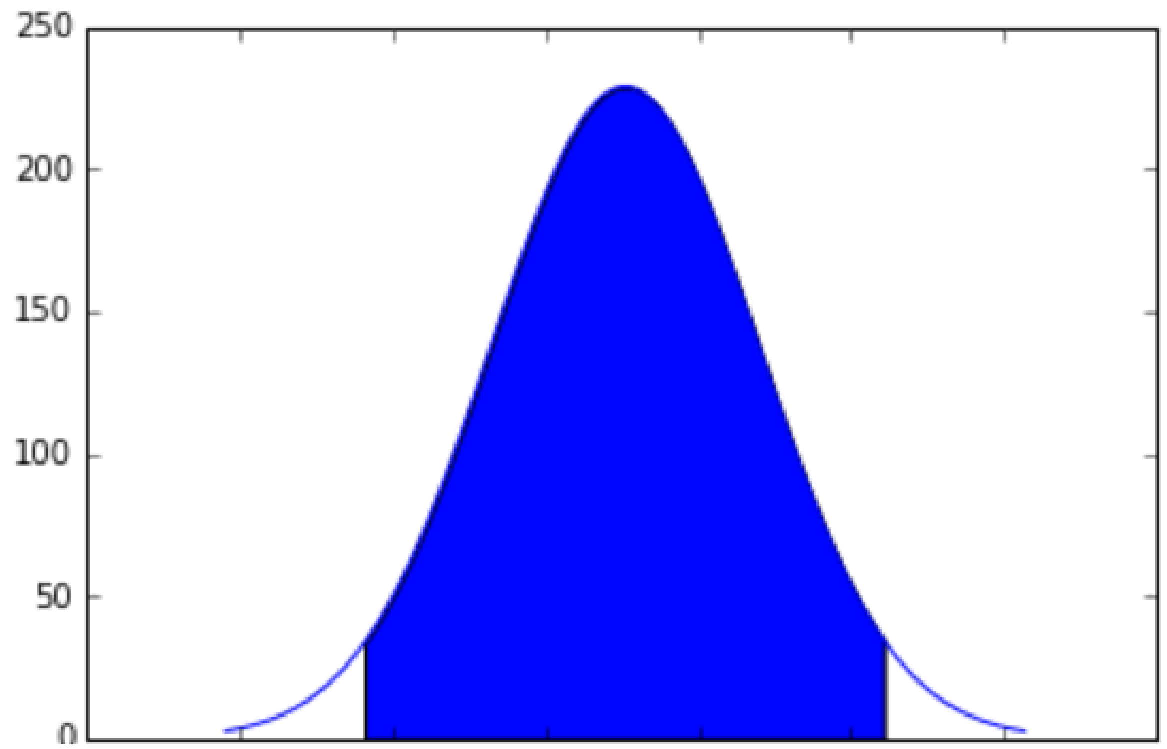
```
# Plot our difference in conversion rates
# as a distribution
diff_line = np.linspace(-3 * var**0.5 + lift,
    3 * var**0.5 + lift, 100
)
plt.plot(diff_line,mlab.normpdf(
    diff_line, lift, var**0.5)
)
plt.show()
```



Plotting the confidence interval

```
# Find values over our confidence interval
section = np.arange(0.007624, 0.01445 , 1/10000)

# Fill in between those boundaries
plt.fill_between(
    section,
    mlab.normpdf(section,lift, var**0.5)
)

# Plot the difference with the confidence int.
plt.plot(
    diff_line,
    mlab.normpdf(diff_line, lift, var**0.5)
)
plt.show()
```

- `np.arrange()` : Generate points in an interval
- `plt.fill_between()` : Fill in an interval



Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Finale

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON