Building functions to automate analysis

ANALYZING MARKETING CAMPAIGNS WITH PANDAS

Jill Rosok
Data Scientist

Why build a function?

```
# Count the subs by referring channel and day
retention_total = marketing.groupby(['date_subscribed',
                                     'subscribing_channel'])\
                                     ['user_id'].nunique()

retention_subs = marketing[marketing['is_retained'] == True]\
                           .groupby(['date_subscribed',
                                          'subscribing_channel'])\
                            ['user_id'].nunique()

# Calculate the retention rate
daily_retention_rate = retention_subs/retention_total
daily_retention_rate = pd.DataFrame(
    daily_retention_rate.unstack(level=1)
)
```

```python
print(daily_retention_rate)
```

```
subscribing_channel  Email  Facebook  House Ads  Instagram      Push
date_subscribed
2018-01-01            1.00  0.875000   0.687500   0.750000  1.000000
2018-01-02            0.75  1.000000   0.588235   0.625000  1.000000
2018-01-03             NaN  0.800000   0.647059   0.909091  0.666667
2018-01-04            1.00  0.666667   0.466667   0.500000       NaN
2018-01-05            1.00  0.571429   0.500000   0.636364  1.000000
```

Building a retention function

```python
def retention_rate(dataframe, column_names):
    # Group by column_names and calculate retention
    retained = dataframe[dataframe['is_retained'] == True]\
                    .groupby(column_names)['user_id'].nunique()

    # Group by column_names and calculate conversion
    converted = dataframe[dataframe['converted'] == True]\
                     .groupby(column_names)['user_id'].nunique()

    retention_rate = retained/converted

    return retention_rate
```

Retention rate by channel

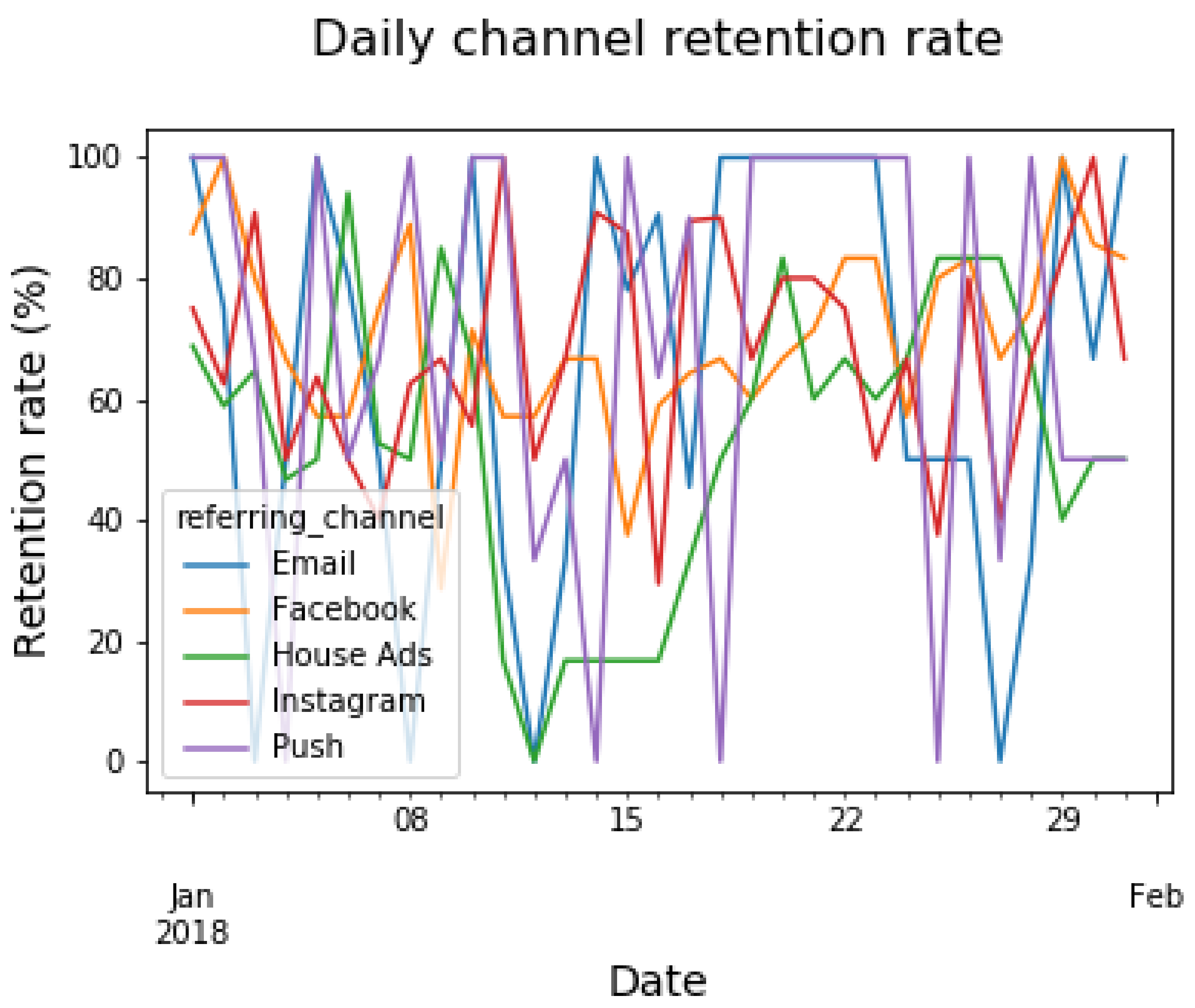
```python
daily_retention = retention_rate(marketing,
                                 ['date_subscribed',
                                  'subscribing_channel'])


daily_retention = pd.DataFrame(
    daily_retention.unstack(level=1)
)
print(daily_retention.head())
```

```
subscribing_channel  Email  Facebook  House Ads  Instagram      Push
date_subscribed
2018-01-01            1.00  0.875000   0.687500   0.750000  1.000000
2018-01-02            0.75  1.000000   0.588235   0.625000  1.000000
2018-01-03             NaN  0.800000   0.647059   0.909091  0.666667
2018-01-04            1.00  0.666667   0.466667   0.500000       NaN
2018-01-05            1.00  0.571429   0.500000   0.636364  1.000000
```

Plotting daily retention by channel

```
daily_retention.plot(date_subscribed, conversion_rate)
plt.title('Daily channel retention rate\n', size = 16)
plt.ylabel('Retention rate (%)', size = 14)
plt.xlabel('Date', size = 14)
plt.show()
```

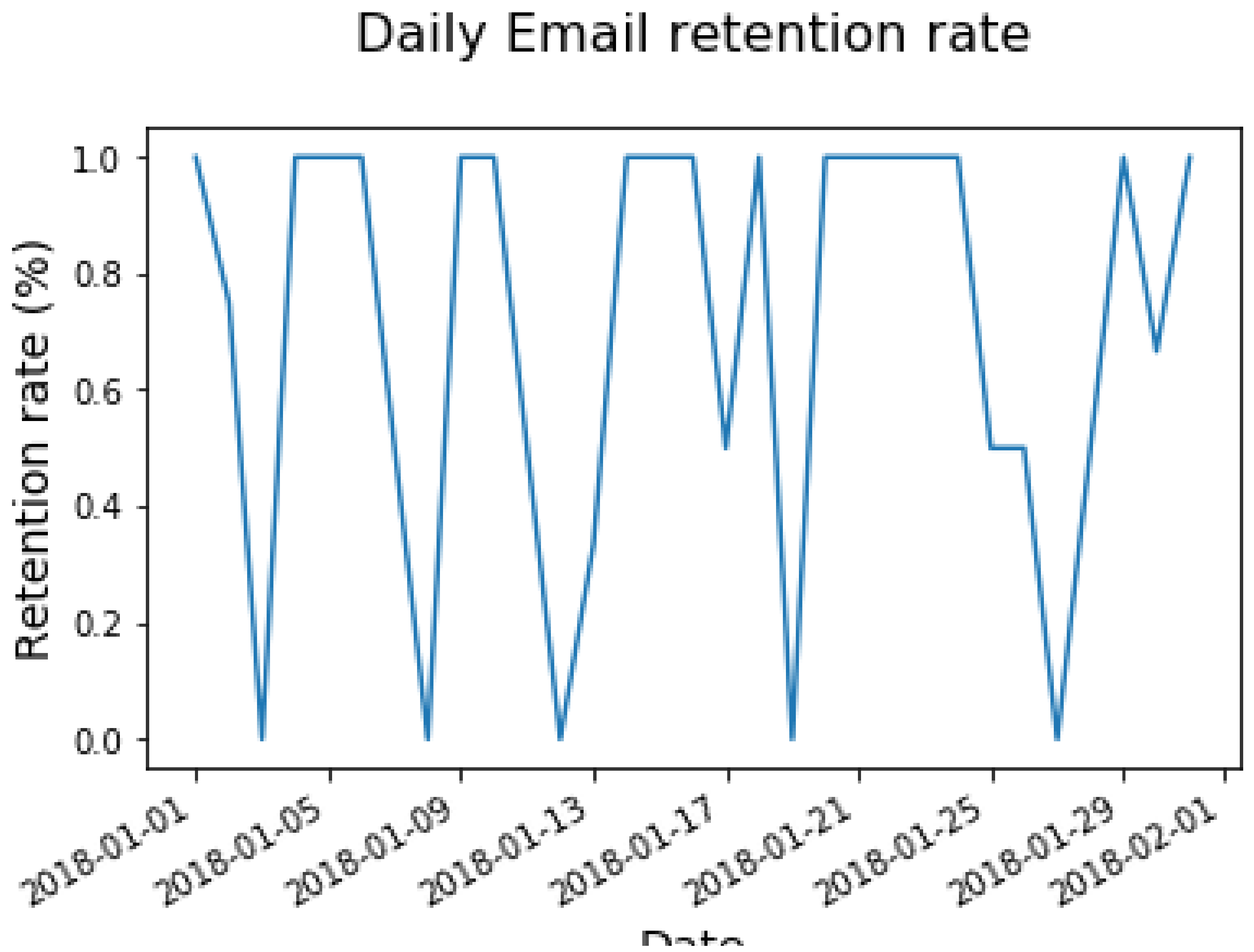


Daily channel retention rate
Retention rate (%)
100
80
60
40
20
0
referring_channel
Email
Facebook
House Ads
Instagram
Push
08
15
22
29
Jan
2018
Feb
Date

Plotting function

```
def plotting(dataframe):
    for column in dataframe:
        plt.plot(dataframe.index, dataframe[column])
        plt.title('Daily ' + column + ' retention rate\n',
                  size = 16)
        plt.ylabel('Retention rate (%)', size = 14)
        plt.xlabel('Date', size = 14)
        plt.show()


plotting(daily_channel_retention)
```


Daily Email retention rate
Retention rate (%)
1.0
0.8
0.6
0.4
0.2
0.0
2018-01-01
2018-01-05
2018-01-09
2018-01-13
2018-01-17
2018-01-21
2018-01-25
2018-01-29
2018-02-01
Date

Let's practice!

ANALYZING MARKETING CAMPAIGNS WITH PANDAS

Identifying inconsistencies

ANALYZING MARKETING CAMPAIGNS WITH PANDAS

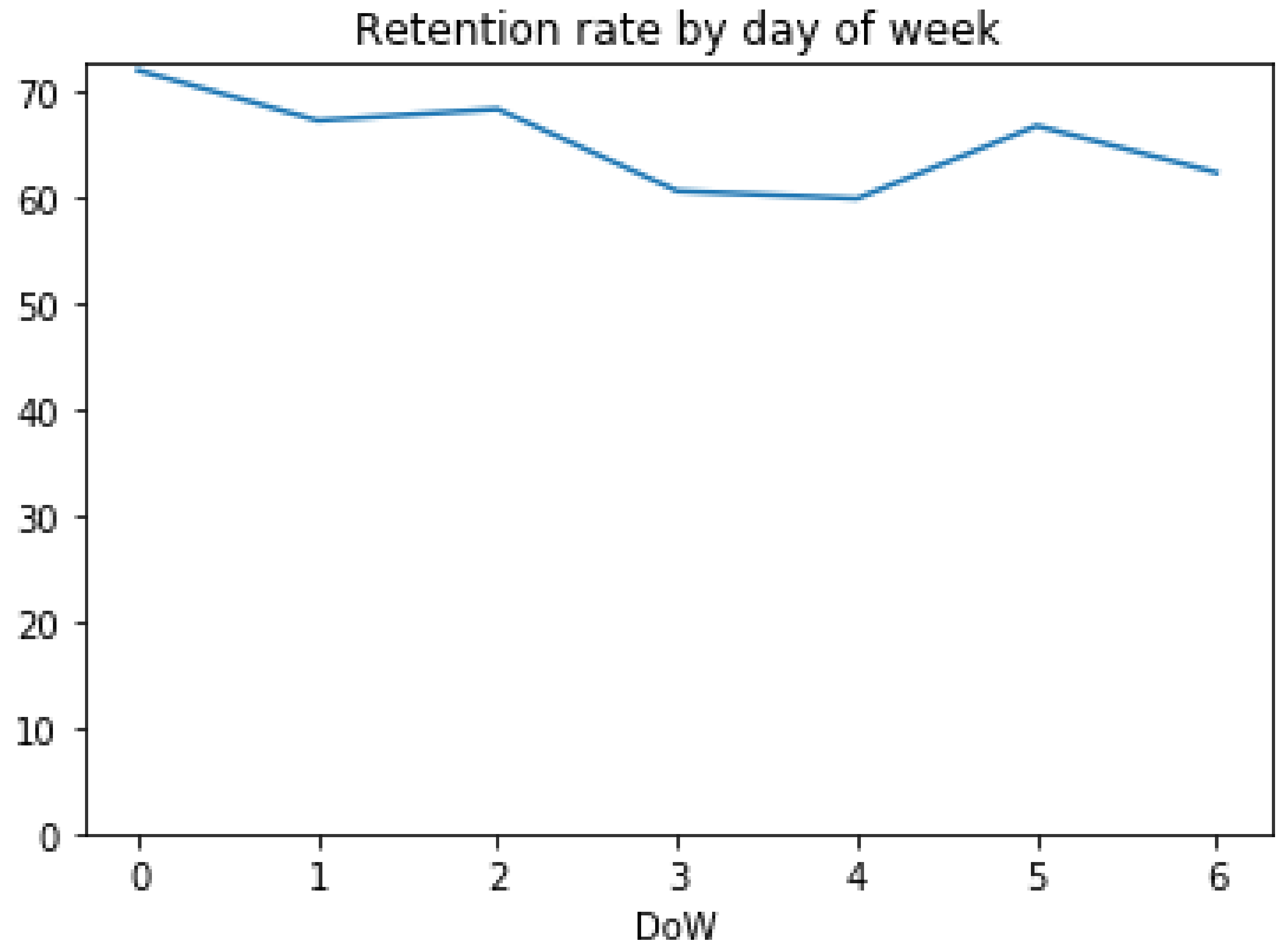
Jill Rosok
Data Scientist

Day of week trends

```
DoW_retention = retention_rate(marketing, ['DoW'])
```

Plotting the results

```
# Plot retention by day of week
DoW_retention.plot()
plt.title('Retention rate by day of week')
plt.ylim(0)
plt.show()
```


Retention rate by day of week
70
60
50
40
30
20
10
0
0
1
2
3
4
5
6
DoW

Real data can be messy and confusing

Let's practice!

ANALYZING MARKETING CAMPAIGNS WITH PANDAS

Resolving inconsistencies

ANALYZING MARKETING CAMPAIGNS WITH PANDAS

Jill Rosok
Data Scientist

Assessing impact

```
# Calculate pre-error conversion rate
# Bug arose sometime around '2018-01-11'
house_ads_no_bug = house_ads[house_ads['date_served'] < '2018-01-11']
lang_conv = conversion_rate(house_ads_no_bug,
                            ['language_displayed'])
```

Assessing impact

```python
# Index other language conversion rate against English
spanish_index = lang_conv['Spanish']/lang_conv['English']
arabic_index = lang_conv['Arabic']/lang_conv['English']
german_index = lang_conv['German']/lang_conv['English']
```

Interpreting Indexes

```python
print("Spanish index:", spanish_index)
print("Arabic index:", arabic_index)
print("German index:", german_index)
```

```
Spanish index: 1.6819248826291078
Arabic index: 5.045774647887324
German index: 4.485133020344288
```

Daily conversion

```python
# Create actual conversion DataFrame
language_conversion = house_ads.groupby(['date_served', \
                                          'language_preferred'])\
                                   .agg({'user_id':'nunique',\
                                         'converted':'sum'})
```

Daily conversion

```
expected_conversion = pd.DataFrame(
    language_conversion.unstack(level=1)
)
```

```
                  user_id                          converted
language_preferred  Arabic English German Spanish     Arabic English German Spanish
date_served
2018-01-01             2.0   171.0    5.0    11.0          2      13      1       0
2018-01-02             3.0   200.0    5.0    10.0          0      14      3       0
2018-01-03             2.0   179.0    3.0     8.0          0      15      1       1
2018-01-04             2.0   149.0    2.0    14.0          0      12      0       3
2018-01-05             NaN   143.0    1.0    14.0        NaN      17  False       3
...
...
```

Create English conversion rate column

```
# Create English conversion rate column for affected period
language_conversion['actual_english_conversions'] = \
                         language_conversion.loc\
                         ['2018-01-11':'2018-01-31']\
                         [('converted','English')]
```

Calculating daily expected conversion rate

```python
# Create expected conversion rates for each language
language_conversion['expected_spanish_rate'] = \
    language_conversion['actual_english_rate']*spanish_index

language_conversion['expected_arabic_rate'] = \
    language_conversion['actual_english_rate']*arabic_index

language_conversion['expected_german_rate'] = \
    language_conversion['actual_english_rate']*german_index
```

Calculating daily expected conversions

```python
# Multiply total ads served by expected conversion rate
language_conversion['expected_spanish_conversions'] = \
    language_conversion['expected_spanish_rate']/100
    *language_conversion[('user_id','Spanish')]

language_conversion['expected_arabic_conversions'] = \
    language_conversion['expected_arabic_rate']/100
    *language_conversion[('user_id','Arabic')]

language_conversion['expected_german_conversions'] = \
    language_conversion['expected_german_rate']/100
    *language_conversion[('user_id','German')]
```

Determining the number of lost subscribers

```
bug_period = language_conversion.loc['2018-01-11':'2018-01-31']

# Sum expected subscribers for each language
expected_subs = bug_period['expected_spanish_conv_rate'].agg('sum') + \
                bug_period['expected_arabic_conv_rate'].agg('sum') + \
                bug_period['expected_german_conv_rate'].agg('sum')

# Calculate how many subscribers we actually got
actual_subs = bug_period[('converted','Spanish')].sum() + \
              bug_period[('converted','Arabic')].agg('sum') + \
              bug_period[('converted','German')].agg('sum')



lost_subs = expected_subs - actual_subs
print(lost_subs)
```

```
32.14414319248826
```

Let's practice!

ANALYZING MARKETING CAMPAIGNS WITH PANDAS