CUSTOMER SEGMENTATION IN PYTHON

Data pre-processing for k-means clustering

Karolis Urbonas
Head of Data Science, Amazon

Advantages of k-means clustering

- One of the most popular unsupervised learning method
- Simple and fast
- Works well*

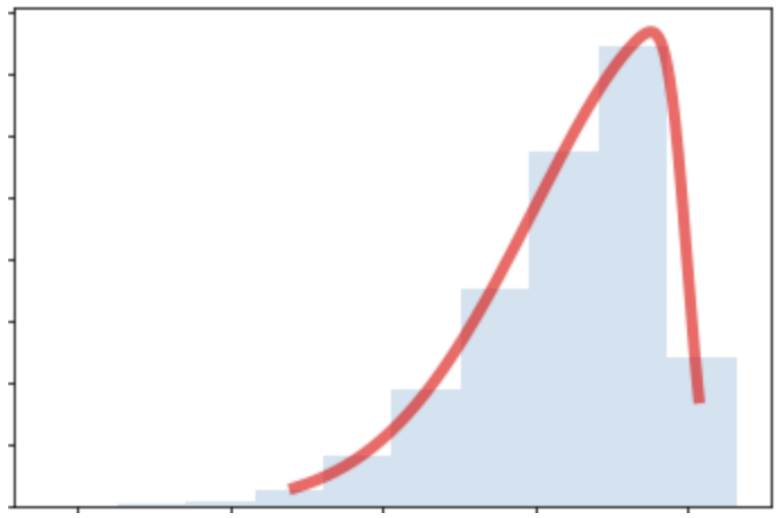
** with certain assumptions about the data*

Key k-means assumptions

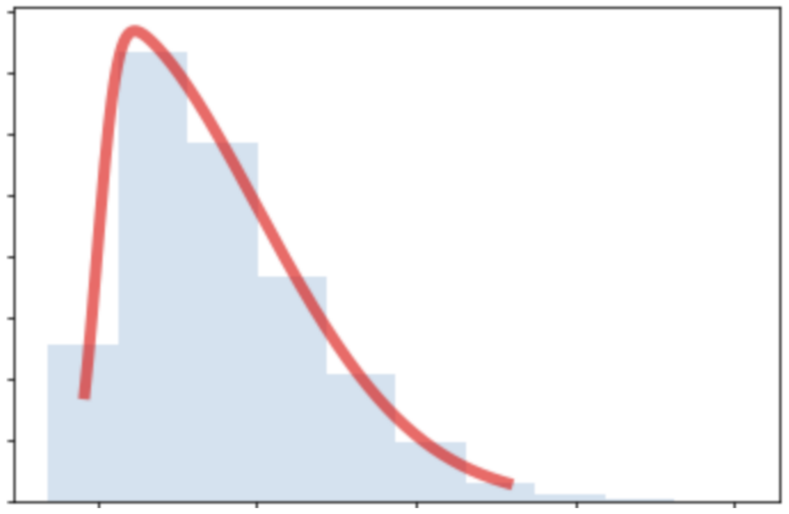
- Symmetric distribution of variables (not skewed)
- Variables with same average values
- Variables with same variance

Skewed variables

- Left-skewed

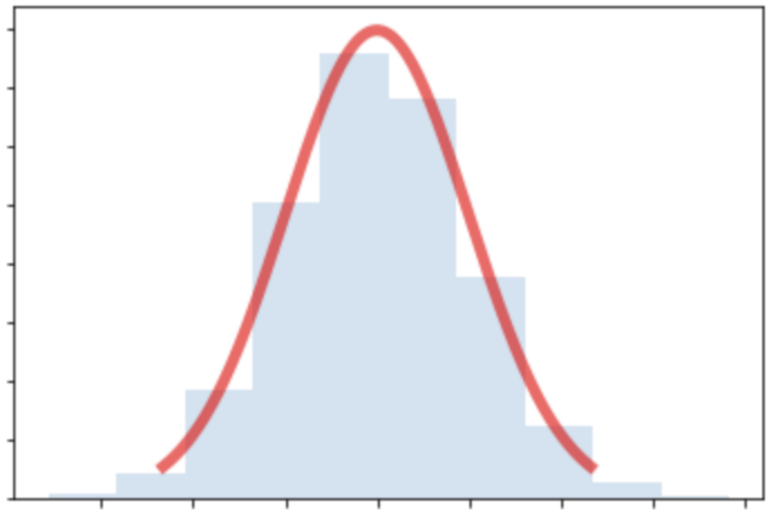


- Right-skewed



Skewed variables

- Skew removed with logarithmic transformation



Variables on the same scale

- K-means assumes equal mean
- And equal variance
- It's not the case with RFM data

```
datamart_rfm.describe()
```

	Recency	Frequency	MonetaryValue
count	3643.00000	3643.000000	3643.000000
mean	90.43563	18.714247	370.694387
std	94.44651	43.754468	1347.443451
min	1.00000	1.000000	0.650000
25%	19.00000	4.000000	58.705000
50%	51.00000	9.000000	136.370000
75%	139.00000	21.000000	334.350000
max	365.00000	1497.000000	48060.350000

CUSTOMER SEGMENTATION IN PYTHON

Let's review the concepts

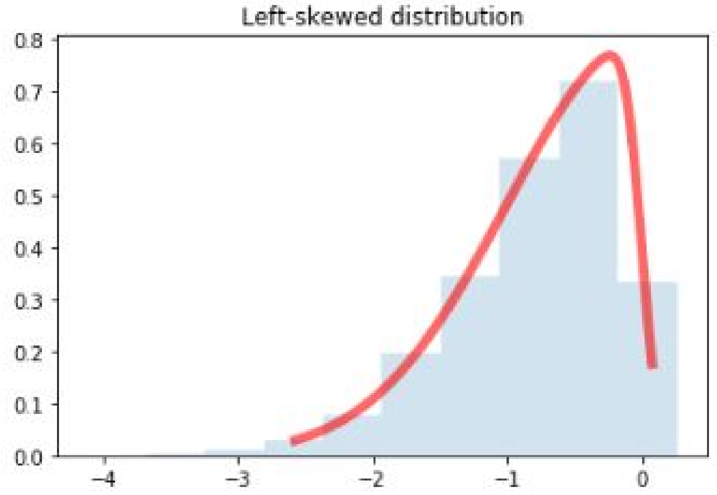
CUSTOMER SEGMENTATION IN PYTHON

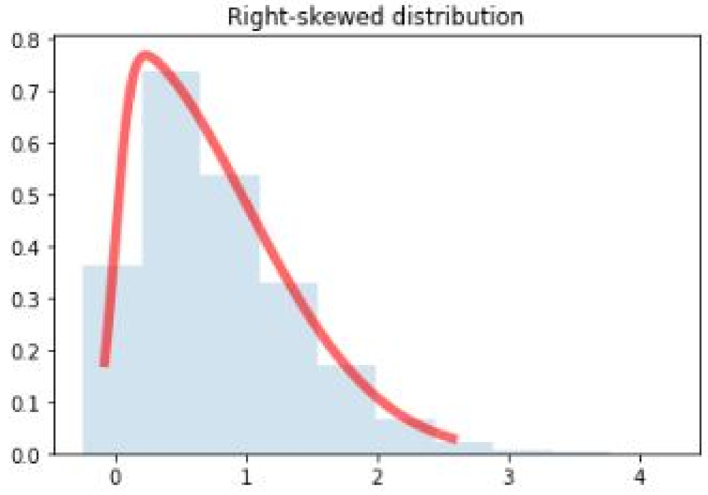
Managing skewed variables

Karolis Urbonas
Head of Data Science, Amazon

Identifying skewness

- Visual analysis of the distribution
- If it has a tail - it's skewed

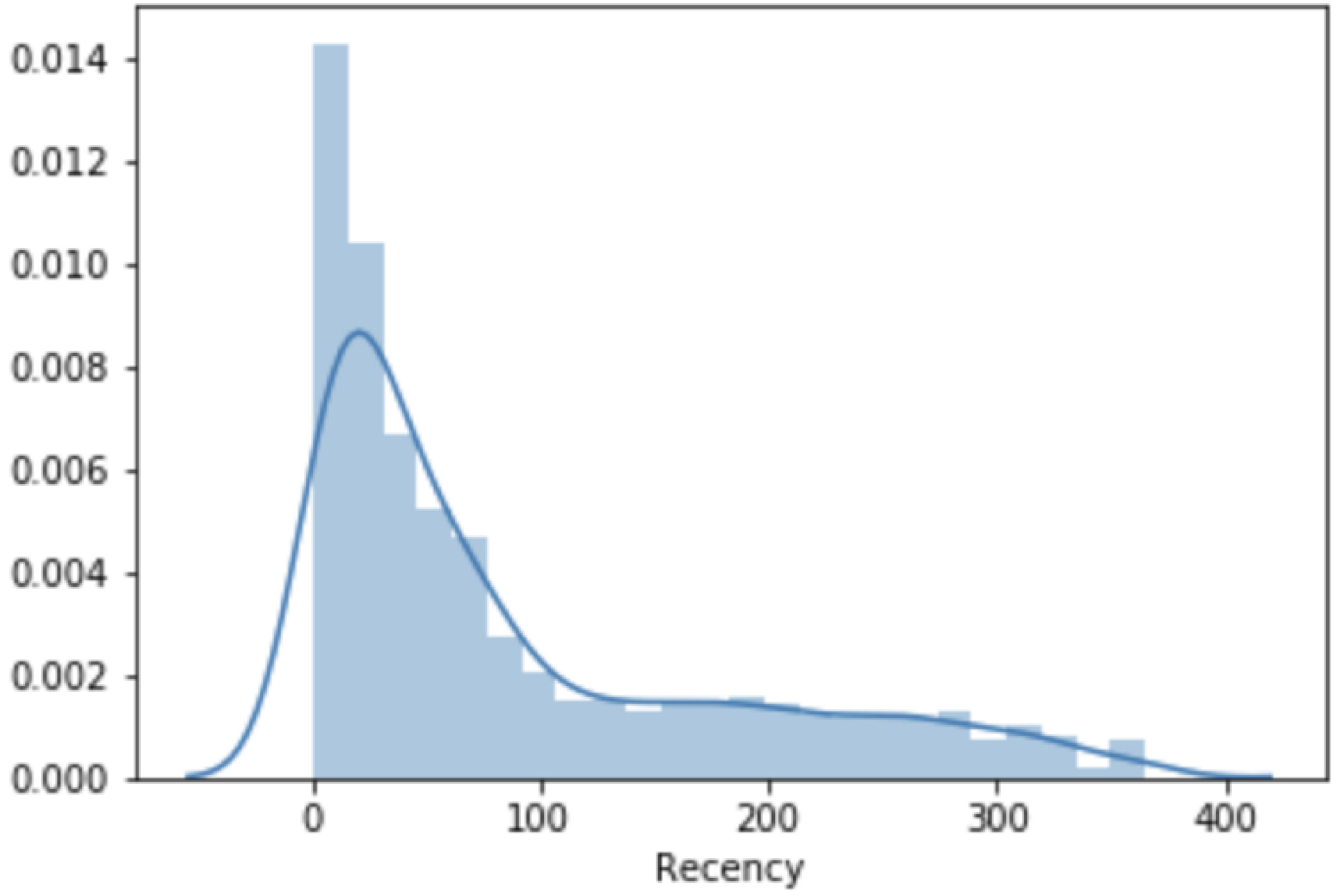




Exploring distribution of Recency

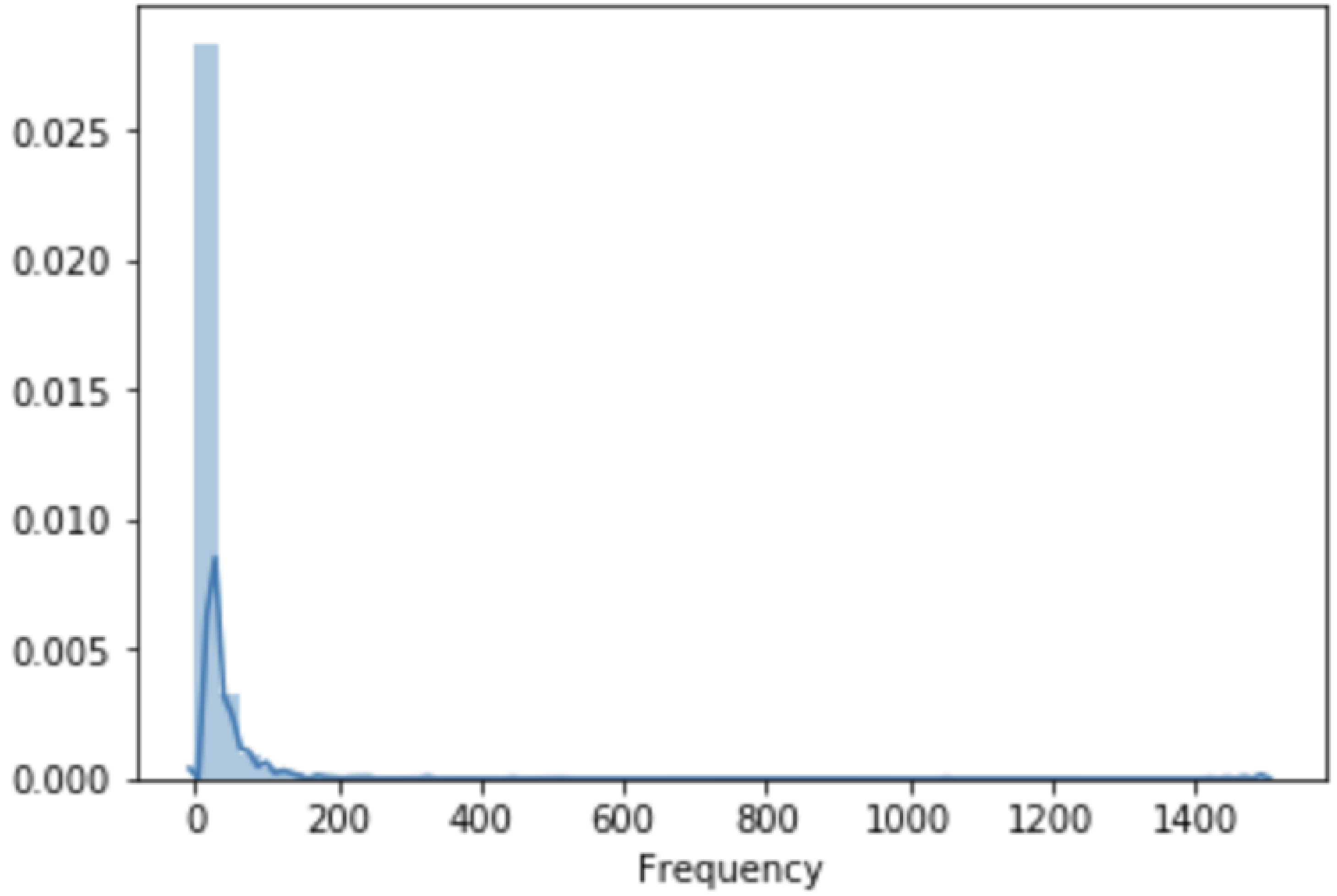
```
import seaborn as sns
from matplotlib import pyplot as plt

sns.distplot(datamart['Recency'])
plt.show()
```



Exploring distribution of Frequency

```
sns.distplot(datamart['Frequency'])
plt.show()
```

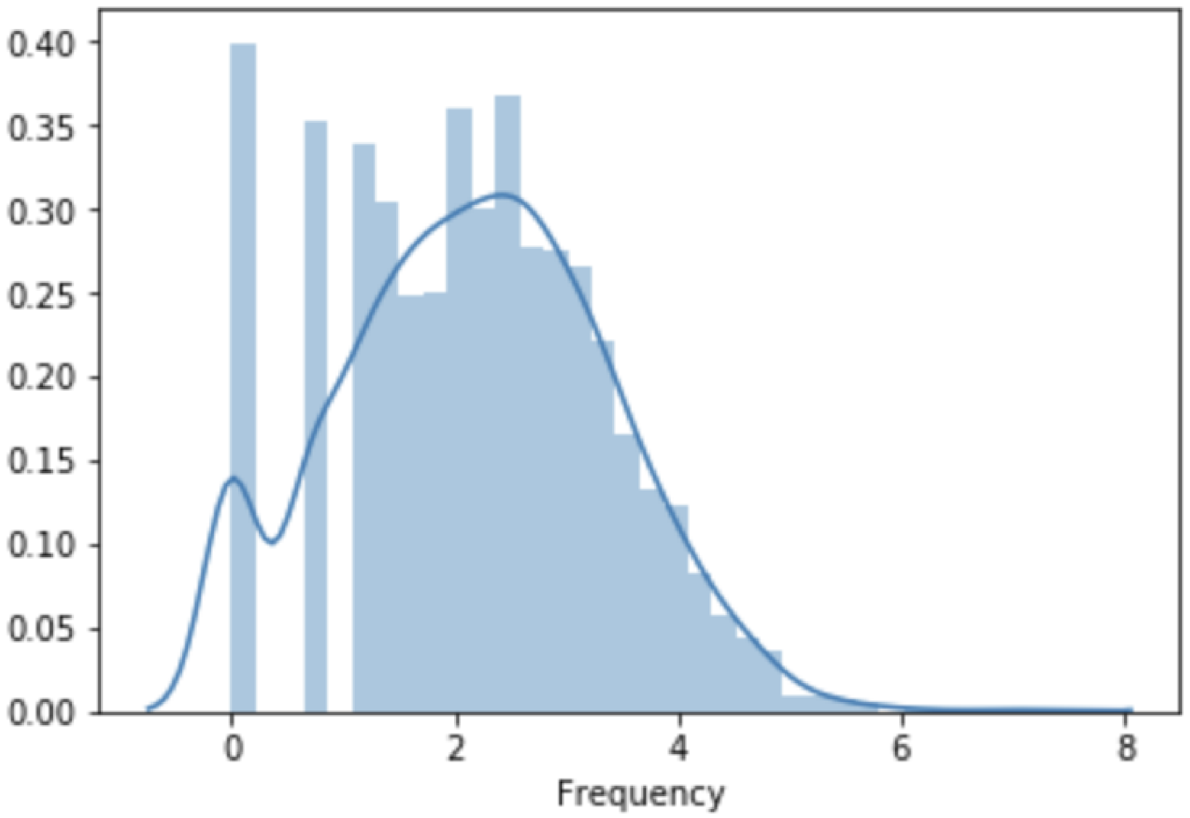


Data transformations to manage skewness

- Logarithmic transformation (positive values only)

```
import numpy as np
frequency_log= np.log(datamart['Frequency'])

sns.distplot(frequency_log)
plt.show()
```



Dealing with negative values

- Adding a constant before log transformation
- Cube root transformation

CUSTOMER SEGMENTATION IN PYTHON

Let's practice how to identify and manage skewed variables!

CUSTOMER SEGMENTATION IN PYTHON

Centering and scaling variables

Karolis Urbonas
Head of Data Science, Amazon

Identifying an issue

- Analyze key statistics of the dataset
- Compare mean and standard deviation

```
datamart_rfm.describe()
```

	Recency	Frequency	MonetaryValue
count	3643.00000	3643.000000	3643.000000
mean	90.43563	18.714247	370.694387
std	94.44651	43.754468	1347.443451
min	1.00000	1.000000	0.650000
25%	19.00000	4.000000	58.705000
50%	51.00000	9.000000	136.370000
75%	139.00000	21.000000	334.350000
max	365.00000	1497.000000	48060.350000

Centering variables with different means

- K-means works well on variables with the same mean
- Centering variables is done by subtracting average value from each observation

```
datamart_centered = datamart_rfm - datamart_rfm.mean()
datamart_centered.describe().round(2)
```

	Recency	Frequency	MonetaryValue
count	3643.00	3643.00	3643.00
mean	0.00	-0.00	0.00
std	94.45	43.75	1347.44
min	-89.44	-17.71	-370.04
25%	-71.44	-14.71	-311.99
50%	-39.44	-9.71	-234.32
75%	48.56	2.29	-36.34
max	274.56	1478.29	47689.66

Scaling variables with different variance

- K-means works better on variables with the same variance / standard deviation
- Scaling variables is done by dividing them by standard deviation of each

```
datamart_scaled = datamart_rfm / datamart_rfm.std()
datamart_scaled.describe().round(2)
```

	Recency	Frequency	MonetaryValue
count	3643.00	3643.00	3643.00
mean	0.96	0.43	0.28
std	1.00	1.00	1.00
min	0.01	0.02	0.00
25%	0.20	0.09	0.04
50%	0.54	0.21	0.10
75%	1.47	0.48	0.25
max	3.86	34.21	35.67

Combining centering and scaling

- Subtract mean and divide by standard deviation manually
- Or use a scaler from `scikit-learn` library (returns `numpy.ndarray` object)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(datamart_rfm)
datamart_normalized = scaler.transform(datamart_rfm)
```

```
print('mean: ', datamart_normalized.mean(axis=0).round(2))
print('std: ', datamart_normalized.std(axis=0).round(2))

mean:  [-0. -0.  0.]
std:  [1. 1. 1.]
```

CUSTOMER SEGMENTATION IN PYTHON

Test different approaches by yourself!

CUSTOMER SEGMENTATION IN PYTHON

Sequence of structuring pre-processing steps

Karolis Urbonas
Head of Data Science, Amazon

Why the sequence matters?

- Log transformation only works with positive data
- Normalization forces data to have negative values and `log` will not work

Sequence

1. Unskew the data - log transformation
2. Standardize to the same average values
3. Scale to the same standard deviation
4. Store as a separate array to be used for clustering

Coding the sequence

Unskew the data with log transformation

```
import numpy as np
datamart_log = np.log(datamart_rfm)
```

Normalize the variables with `StandardScaler`

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(datamart_log)
```

Store it separately for clustering

```
datamart_normalized = scaler.transform(datamart_log)
```

CUSTOMER SEGMENTATION IN PYTHON

Practice on RFM data!