

# Recency, frequency, monetary (RFM) segmentation

CUSTOMER SEGMENTATION IN PYTHON



**Karolis Urbonas**

Head of Data Science, Amazon

# What is RFM segmentation?

Behavioral customer segmentation based on three metrics:

- Recency (R)
- Frequency (F)
- Monetary Value (M)

# Grouping RFM values

The RFM values can be grouped in several ways:

- Percentiles e.g. quantiles
- Pareto 80/20 cut
- Custom - based on business knowledge

We are going to implement percentile-based grouping.

# Short review of percentiles

Process of calculating percentiles:

1. Sort customers based on that metric
2. Break customers into a pre-defined number of groups of equal size
3. Assign a label to each group

# Calculate percentiles with Python

Data with eight `CustomerID` and a randomly calculated `Spend` values.

	CustomerID	Spend
0	0	137
1	1	335
2	2	172
3	3	355
4	4	303
5	5	233
6	6	244
7	7	229

# Calculate percentiles with Python

```
spend_quartiles = pd.qcut(data['Spend'], q=4, labels=range(1,5))  
data['Spend_Quartile'] = spend_quartiles  
data.sort_values('Spend')
```

	CustomerID	Spend	Spend_Quartile
0	0	137	1
2	2	172	1
7	7	229	2
5	5	233	2
6	6	244	3
4	4	303	3
1	1	335	4
3	3	355	4

# Assigning labels

- Highest score to the best metric - best is not always highest e.g. recency
- In this case, the label is inverse - the more recent the customer, the better

	CustomerID	Recency_Days
0	0	37
1	1	235
2	2	396
3	3	72
4	4	255
5	5	393
6	6	203
7	7	133

# Assigning labels

```
# Create numbered labels
r_labels = list(range(4, 0, -1))

# Divide into groups based on quartiles
recency_quartiles = pd.qcut(data['Recency_Days'], q=4, labels=r_labels)

# Create new column
data['Recency_Quartile'] = recency_quartiles

# Sort recency values from lowest to highest
data.sort_values('Recency_Days')
```



# Assigning labels

As you can see, the quartile labels are reversed, since the more recent customers are more valuable.

	CustomerID	Recency_Days	Recency_Quartile
0	0	37	4
3	3	72	4
7	7	133	3
6	6	203	3
1	1	235	2
4	4	255	2
5	5	393	1
2	2	396	1

# Custom labels

We can define a list with string or any other values, depending on the use case.

```
# Create string labels
r_labels = ['Active', 'Lapsed', 'Inactive', 'Churned']
# Divide into groups based on quartiles
recency_quartiles = pd.qcut(data['Recency_Days'], q=4, labels=r_labels)

# Create new column
data['Recency_Quartile'] = recency_quartiles

# Sort values from lowest to highest
data.sort_values('Recency_Days')
```

# Custom labels

Custom labels assigned to each quartile

	CustomerID	Recency_Days	Recency_Quartile
0	0	37	Active
3	3	72	Active
7	7	133	Lapsed
6	6	203	Lapsed
1	1	235	Inactive
4	4	255	Inactive
5	5	393	Churned
2	2	396	Churned

# Let's practice with percentiles!

CUSTOMER SEGMENTATION IN PYTHON

# Calculating RFM metrics

CUSTOMER SEGMENTATION IN PYTHON



**Karolis Urbonas**

Head of Data Science, Amazon

# Definitions

- Recency - days since last customer transaction
- Frequency - number of transactions in the last 12 months
- Monetary Value - total spend in the last 12 months

# Dataset and preparations

- Same `online` dataset like in the previous lessons
- Need to do some data preparation
- New `TotalSum` column = `Quantity` x `UnitPrice` .

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSum
<b>416792</b>	572558	22745	POPPY'S PLAYHOUSE BEDROOM	6	2011-10-25	2.10	14286	United Kingdom	12.60
<b>482904</b>	577485	23196	VINTAGE LEAF MAGNETIC NOTEPAD	1	2011-11-20	1.45	16360	United Kingdom	1.45
<b>263743</b>	560034	23299	FOOD COVER WITH BEADS SET 2	6	2011-07-14	3.75	13933	United Kingdom	22.50
<b>495549</b>	578307	72349B	SET/6 PURPLE BUTTERFLY T-LIGHTS	1	2011-11-23	2.10	17290	United Kingdom	2.10
<b>204384</b>	554656	21756	BATH BUILDING BLOCK WORD	3	2011-05-25	5.95	17663	United Kingdom	17.85

# Data preparation steps

We're starting with a pre-processed `online` DataFrame with only the latest 12 months of data:

```
print('Min: {}; Max: {}'.format(min(online.InvoiceDate),  
                                max(online.InvoiceDate)))
```

```
Min:2010-12-10; Max:2011-12-09
```

Let's create a hypothetical `snapshot_day` data as if we're doing analysis recently.

```
snapshot_date = max(online.InvoiceDate) + datetime.timedelta(days=1)
```



# Calculate RFM metrics

```
# Aggregate data on a customer level
datamart = online.groupby(['CustomerID']).agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,
    'InvoiceNo': 'count',
    'TotalSum': 'sum'})

# Rename columns for easier interpretation
datamart.rename(columns = {'InvoiceDate': 'Recency',
                           'InvoiceNo': 'Frequency',
                           'TotalSum': 'MonetaryValue'}, inplace=True)

# Check the first rows
datamart.head()
```

# Final RFM values

Our table for RFM segmentation is completed!

	Recency	Frequency	MonetaryValue
CustomerID			
12747	3	25	948.70
12748	1	888	7046.16
12749	4	37	813.45
12820	4	17	268.02
12822	71	9	146.15

# Let's practice calculating RFM values!

CUSTOMER SEGMENTATION IN PYTHON

# Building RFM segments

CUSTOMER SEGMENTATION IN PYTHON



**Karolis Urbonas**

Head of Data Science, Amazon

# Data

- Dataset we created previously
- Will calculate quartile value for each column and name then R , F , M

	Recency	Frequency	MonetaryValue
CustomerID			
12747	3	25	948.70
12748	1	888	7046.16
12749	4	37	813.45
12820	4	17	268.02
12822	71	9	146.15

# Recency quartile

```
r_labels = range(4, 0, -1)
r_quartiles = pd.qcut(datamart['Recency'], 4, labels = r_labels)
datamart = datamart.assign(R = r_quartiles.values)
```

	Recency	Frequency	MonetaryValue	R
CustomerID				
12747	3	25	948.70	4
12748	1	888	7046.16	4
12749	4	37	813.45	4
12820	4	17	268.02	4
12822	71	9	146.15	2

# Frequency and monetary quartiles

```
f_labels = range(1,5)
m_labels = range(1,5)
f_quartiles = pd.qcut(datamart['Frequency'], 4, labels = f_labels)
m_quartiles = pd.qcut(datamart['MonetaryValue'], 4, labels = m_labels)
datamart = datamart.assign(F = f_quartiles.values)
datamart = datamart.assign(M = m_quartiles.values)
```

	Recency	Frequency	MonetaryValue	R	F	M
CustomerID						
12747	3	25	948.70	4	4	4
12748	1	888	7046.16	4	4	4
12749	4	37	813.45	4	4	4
12820	4	17	268.02	4	3	3
12822	71	9	146.15	2	2	3

# Build RFM segment and RFM score

- Concatenate RFM quartile values to RFM\_Segment
- Sum RFM quartiles values to RFM\_Score

```
def join_rfm(x): return str(x['R']) + str(x['F']) + str(x['M'])  
datamart['RFM_Segment'] = datamart.apply(join_rfm, axis=1)  
datamart['RFM_Score'] = datamart[['R', 'F', 'M']].sum(axis=1)
```



# Final result

	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score
CustomerID								
18108	255	4	58.60	1	1	1	111	3.0
13803	256	3	57.70	1	1	1	111	3.0
12922	162	4	57.24	1	1	1	111	3.0
13304	352	4	57.21	1	1	1	111	3.0
17496	359	2	57.15	1	1	1	111	3.0
16063	261	4	57.10	1	1	1	111	3.0
17531	191	2	57.00	1	1	1	111	3.0
14206	240	3	57.00	1	1	1	111	3.0
13784	219	2	55.80	1	1	1	111	3.0
14476	258	4	55.75	1	1	1	111	3.0

# Let's practice building RFM segments

CUSTOMER SEGMENTATION IN PYTHON

# Analyzing RFM segments

CUSTOMER SEGMENTATION IN PYTHON



**Karolis Urbonas**

Head of Data Science, Amazon

# Largest RFM segments

```
datamart.groupby('RFM_Segment').size().sort_values(ascending=False)[:10]
```

RFM_Segment	
444	372
111	345
211	169
344	156
233	129
222	128
333	120
122	117
311	114
433	113
dtype:	int64

# Filtering on RFM segments

- Select bottom RFM segment "111" and view top 5 rows

```
datamart[datamart['RFM_Segment']=='111'][:5]
```

	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score
CustomerID								
12837	174	2	10.55	1	1	1	111	3.0
12852	295	2	32.55	1	1	1	111	3.0
12902	265	4	42.03	1	1	1	111	3.0
12915	149	2	35.90	1	1	1	111	3.0
12922	162	4	57.24	1	1	1	111	3.0

# Summary metrics per RFM score

```
datamart.groupby('RFM_Score').agg({  
    'Recency': 'mean',  
    'Frequency': 'mean',  
    'MonetaryValue': ['mean', 'count'] })/  
.round(1)
```

RFM_Score	Recency	Frequency	MonetaryValue	count
	mean	mean	mean	
3.0	246.9	2.1	28.4	345
4.0	162.2	3.1	47.8	337
5.0	138.9	4.3	78.2	393
6.0	101.0	6.3	146.3	444
7.0	78.0	8.5	160.2	382
8.0	62.6	12.8	196.3	376
9.0	46.8	16.7	330.3	345
10.0	31.9	24.0	443.1	355
11.0	21.8	38.9	705.3	294
12.0	8.0	75.6	1653.9	372

# Grouping into named segments

Use RFM score to group customers into **Gold**, **Silver** and **Bronze** segments.

```
def segment_me(df):  
    if df['RFM_Score'] >= 9:  
        return 'Gold'  
    elif (df['RFM_Score'] >= 5) and (df['RFM_Score'] < 9):  
        return 'Silver'  
    else:  
        return 'Bronze'  
  
datamart['General_Segment'] = datamart.apply(segment_me, axis=1)  
datamart.groupby('General_Segment').agg({  
    'Recency': 'mean',  
    'Frequency': 'mean',  
    'MonetaryValue': ['mean', 'count']}).round(1)
```

# New segments and their values

General_Segment	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
1. Gold	27.0	39.4	800.8	1366
2. Silver	95.8	7.9	144.6	1595
3. Bronze	205.0	2.6	38.0	682



# Practice building custom segments

CUSTOMER SEGMENTATION IN PYTHON