Propaganda Exploratory Data Analysis

Created by `Zorian Kulyk`

In [1]:
```
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import plotly.io as pio
pio.renderers.default = "notebook"
```

Intro

The primary goal is to make to EDA based on telegram data from russian propagandistic channels from 2015 to 2023 years.

Columns explanation:

- `id` - *post id*
- `date` - *post date and time (should be UTC+0)*
- `views` - *self explanatory*
- `reactions` - *self explanatory*
- `to_id` - *the same for all posts within channel (can be used to separate posts from comments)*
- `fwd_from` - *the part "from_id=PeerChannel(channel_id=...)" contains info on the id of channels the post of forwarded from*
- `message` - *self explanatory*
- `type` - *text if only contains text, other - if media files are present (does not mean that there's no text)*
- `duration` - *available for posts with media*
- `channel_name` - *name used as @ handle*
- `frw_from_title` - *name of channel (public - not the handle)*
- `frw_from_name` - *channel_name of channel (the handle)*
- `msg_entity` - *not useful - can be dropped*
- `datetime` - *datetime object for post*
- `message_len` - *number of characters and messages*
- `reactions_dict` - *serialized dict with reactions count*
- `reactions_num` - *sum of all reactions*
- `_from_id` - *id of source channel*
- `_to_id` - *id of target channel*
- `sensitive-topic` - *sensitive topic*
  classified with https://huggingface.co/apanc/russian-sensitive-topics (https://huggingface.co/apanc/russian-sensitive-topics) , 18 classes- https://arxiv.org/abs/2103.05345 (https://arxiv.org/abs/2103.05345)
- `toxicity` - *label if message is toxic*
  classified with https://huggingface.co/s-nlp/russian_toxicity_classifier (https://huggingface.co/s-nlp/russian_toxicity_classifier)

Data Preprocessing

At first, we will perform preprocessing, such as dropping unuseful columns, wrangling numeric values, handling data types, etc.
Next, we will start our EDA.

In [3]:
```
PATH = r"data/data.csv"
data = pd.read_csv(PATH)
data.head()
```

Out[3]:

	Unnamed: 0	id	date	views	reactions	to_id	fwd_from	message	type	duration	...	frw_from_name
0	0	189123.0	2022-12-19 09:56:04+00:00	98413.0	NaN	PeerChannel(channel_id=1101170442)	NaN	ФТС России ожидает роста товарооборота с Китае...	text	NaN	...	NaN
1	1	189122.0	2022-12-19 09:51:57+00:00	120179.0	NaN	PeerChannel(channel_id=1101170442)	NaN	NaN	photo	NaN	...	NaN
2	2	189121.0	2022-12-19 09:51:57+00:00	116172.0	NaN	PeerChannel(channel_id=1101170442)	NaN	NaN	video	12.0	...	NaN
3	3	189120.0	2022-12-19 09:51:57+00:00	115171.0	NaN	PeerChannel(channel_id=1101170442)	NaN	NaN	photo	NaN	...	NaN
4	4	189119.0	2022-12-19 09:51:57+00:00	118174.0	NaN	PeerChannel(channel_id=1101170442)	NaN	Буэнос-Айрес наутро после праздника	video	10.0	...	NaN

5 rows × 22 columns

In [4]:
```
data.shape
```

Out[4]: (8108693, 22)

Drop unuseful columns.

In [5]:
```
data.drop(["Unnamed: 0", "date", "reactions", "to_id", "msg_entity"], axis=1, inplace=True)
```

In [6]:
```
data.head(3)
```

Out[6]:

	id	views	fwd_from	message	type	duration	channel	frw_from_title	frw_from_name	datetime	message_len	reactions_dict	reactic
0	189123.0	98413.0	NaN	ФТС России ожидает роста товарооборота с Китае...	text	NaN	rian_ru	NaN	NaN	2022-12-19 09:56:04+00:00	205	[]	
1	189122.0	120179.0	NaN	NaN	photo	NaN	rian_ru	NaN	NaN	2022-12-19 09:51:57+00:00	0	[]	
2	189121.0	116172.0	NaN	NaN	video	12.0	rian_ru	NaN	NaN	2022-12-19 09:51:57+00:00	0	[]	

We have inappropriate channel names when the "channel" column values are not real. We will handle it with predefined column "frw_from_name" and replace it with the correct names.

In [7]:
```
data["frw_from_name"][data["frw_from_name"].notna()][:3]
```

Out[7]:
```
506201      readovkaru
506256      readovkaru
506311    suverennews
Name: frw_from_name, dtype: object
```

In [8]:
```
data.iloc[506311]
```
Out[8]:
```
id                                                            48014.0
views                                                        217874.0
fwd_from           MessageFwdHeader(date=datetime.datetime(2022, ...
message            Почему Россия до сих пор не избавилась от тран...
type                                                            photo
duration                                                          NaN
channel                                                   readovkanews
frw_from_title                                     Суверенная экономика
frw_from_name                                              suverennews
datetime                                    2022-12-01 15:45:47+00:00
message_len                                                      1416
reactions_dict     [{"reaction": "\ud83d\udc4d", "count": 699, "c...
reactions_num                                                     803
_from_id                                                 1551891830.0
_to_id                                                     1260622817
sensitive-topic                                       politics,racism
toxicity                                                      neutral
Name: 506311, dtype: object
```
In [9]:
```
real_channel_name = data["frw_from_name"].fillna(data["channel"])
```
In [10]:
```
_data = data.copy()
```
In [11]:
```
_data["channel"] = real_channel_name
_data.drop(["frw_from_title", "frw_from_name"], axis=1, inplace=True)
```
In [12]:
```
_data.head(3)
```
Out[12]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_id
0	189123.0	98413.0	NaN	ФТС России ожидает роста товарооборота с Китае...	text	NaN	rian_ru	2022-12-19 09:56:04+00:00	205	[]	0	NaN	1101170442
1	189122.0	120179.0	NaN	NaN	photo	NaN	rian_ru	2022-12-19 09:51:57+00:00	0	[]	0	NaN	1101170442
2	189121.0	116172.0	NaN	NaN	video	12.0	rian_ru	2022-12-19 09:51:57+00:00	0	[]	0	NaN	1101170442

Next, we will check data types and wrangle missing values.

In [13]:
```
_data.dtypes
```

Out[13]:
```
id                  float64
views               float64
fwd_from             object
message              object
type                 object
duration            float64
channel              object
datetime             object
message_len           int64
reactions_dict       object
reactions_num         int64
_from_id            float64
_to_id                int64
sensitive-topic      object
toxicity             object
dtype: object
```

In [14]:
```
_data["datetime"] = pd.to_datetime(_data["datetime"])
```

In [15]:
```
_data.shape
```

Out[15]:
```
(8108693, 15)
```

In [16]:
```
messages_num = len(_data.id.unique())
messages_num
```

Out[16]:
```
515512
```

Surprisingly, we have a small number of unique IDs in comparison to the shape of our data. We are about to find the problem.

In [17]:
```
_data["id"].value_counts()
```

Out[17]:
```
1.0          299
440.0        269
569.0        268
337.0        268
504.0        268
            ...
410019.0       1
6635705.0      1
316220.0       1
407800.0       1
7220606.0      1
Name: id, Length: 515512, dtype: int64
```

In [18]:
```
_data[_data["id"] == 1].head(3)
```

Out[18]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_id	sensit
187754	1.0	NaN	NaN	NaN	text	NaN	rian_ru	2017-02-28 16:45:55+00:00	0	[]	0	NaN	1101170442	ı
327499	1.0	NaN	NaN	NaN	text	NaN	bbbreaking	2018-07-05 17:36:16+00:00	0	[]	0	NaN	1394050290	ı
466067	1.0	NaN	NaN	NaN	text	NaN	rt_russian	2016-03-28 10:11:37+00:00	0	[]	0	NaN	1036362176	ı

In [19]:
```
_data[_data["id"] == 440].head(3)
```

Out[19]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_
187338	440.0	310.0	NaN	Автобус насмерть сбил более 30 человек на Гаит...	photo	NaN	rian_ru	2017-03-12 18:26:19+00:00	94	[]	0	NaN	11011704
465657	440.0	726.0	NaN	🏆 Путин вручил кубок «Гран-при Россия» «Формул...	text	NaN	rt_russian	2016-05-01 15:10:33+00:00	161	[]	0	NaN	10363621
505731	440.0	8885.0	NaN	Рукожопые московские мастера уничтожили рарите...	text	NaN	breakingmash	2017-06-23 08:26:52+00:00	1007	[]	0	NaN	11176285

As a result, we have possibly many garbage data that we need to clear.

In [20]:
```
na_messages = _data[(_data["message"].isna()) & (_data["type"] == "text")]
```

In [21]:
```
na_messages.head(3)
```

Out[21]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_id
25070	163979.0	1132817.0	NaN	NaN	text	NaN	rian_ru	2022-05-20 00:00:13+00:00	0	[]	0	NaN	1101170442
25071	163978.0	1132637.0	NaN	NaN	text	NaN	rian_ru	2022-05-20 00:00:13+00:00	0	[]	0	NaN	1101170442
25072	163977.0	1133752.0	NaN	NaN	text	NaN	rian_ru	2022-05-20 00:00:13+00:00	0	[]	0	NaN	1101170442

In [22]:
```
_data.drop(na_messages.index, inplace=True)
```

Finally, we should wrangle our "reactions_dict" and convert it to the *.json* format for future work.

In [23]:
```
_data["reactions_dict"] = _data["reactions_dict"].apply(lambda x: json.loads(x))
```

In [24]:
```
_data.head()
```

Out[24]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_id
0	189123.0	98413.0	NaN	ФТС России ожидает роста товарооборота с Китае...	text	NaN	rian_ru	2022-12-19 09:56:04+00:00	205	[]	0	NaN	1101170442
1	189122.0	120179.0	NaN	NaN	photo	NaN	rian_ru	2022-12-19 09:51:57+00:00	0	[]	0	NaN	1101170442
2	189121.0	116172.0	NaN	NaN	video	12.0	rian_ru	2022-12-19 09:51:57+00:00	0	[]	0	NaN	1101170442
3	189120.0	115171.0	NaN	NaN	photo	NaN	rian_ru	2022-12-19 09:51:57+00:00	0	[]	0	NaN	1101170442
4	189119.0	118174.0	NaN	Буэнос-Айрес наутро после праздника	video	10.0	rian_ru	2022-12-19 09:51:57+00:00	35	[]	0	NaN	1101170442

EDA

Now, we will start to explore data. We will take a look at more general data info, patterns, and distributions. But also, we will try to find more complicated relations and finally make conclusions.

Q1-2: How many unique channels does data consist of? What is the period from the first and last messages in collected data?

In [24]:
```
channel_num = len(_data["_to_id"].unique())
print(f"The number of channels: {channel_num}")
```

```
The number of channels: 309
```

In [25]:
```
period = (_data["datetime"].max() - _data["datetime"].min()).days / 365
print(f"The data time period: {period:.2f} years")
```

```
The data time period: 7.26 years
```

Q3-4: How many messages do we have overall? How is the number of messages distributed over time?

In [26]:
```
messages_num = _data['_from_id'].isna().sum()
print(f"The number of unique messages in data: {messages_num}")
print(f"The number of reposted messages in data: {_data.shape[0] - messages_num}")
```

```
The number of unique messages in data: 6055304
The number of reposted messages in data: 2002753
```

In [27]:
```
data_sorted = _data.sort_values("datetime")
data_sorted.head(3)
```

Out[27]:

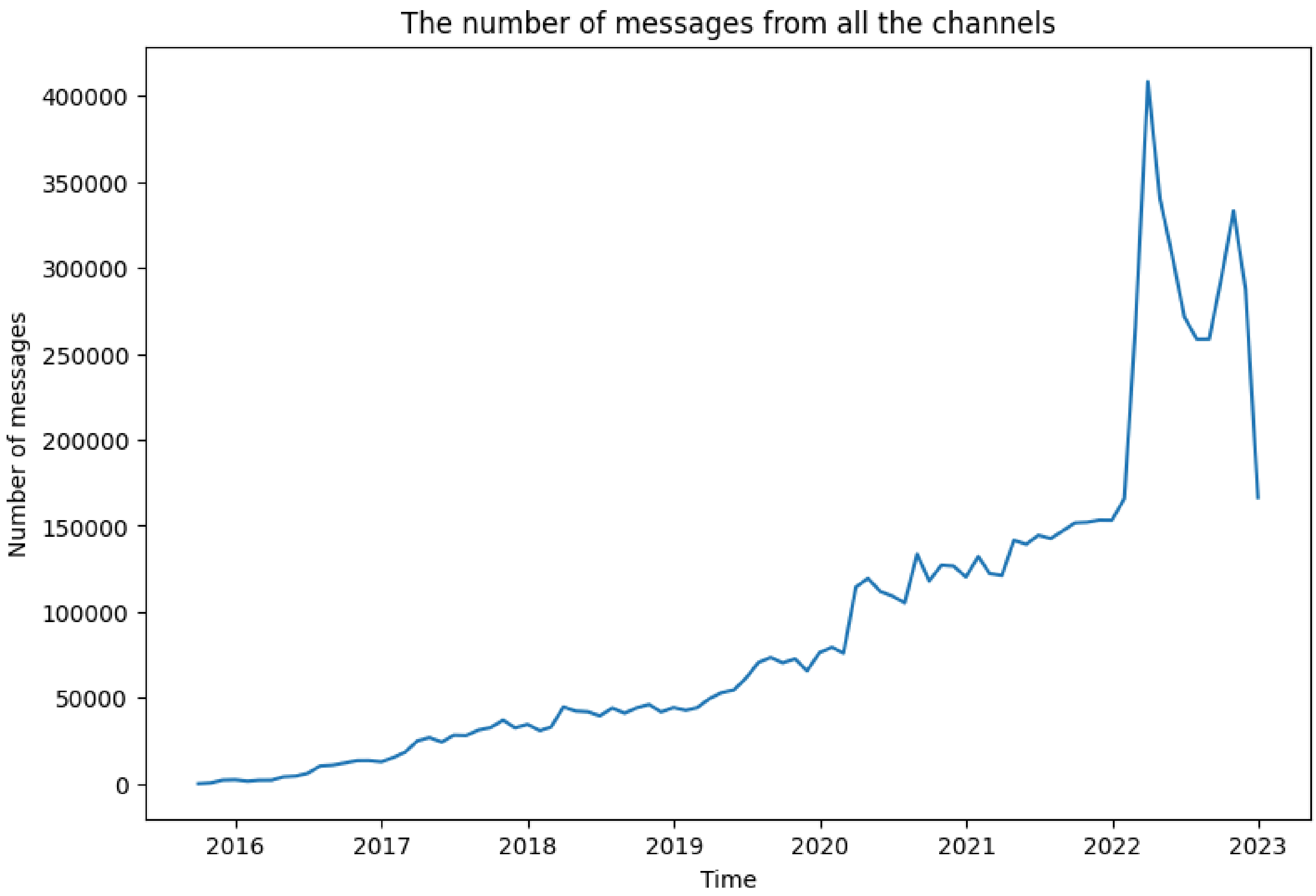
	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reactions_dict	reactions_num	_from_id	_to_id
5282295	9.0	1841.0	NaN	😛	sticker	NaN	varlamov	2015-09-22 16:12:02+00:00	1	[{'reaction': '👍', 'count': 8, 'chosen': False...	12	NaN	1005684212
7768626	2.0	355.0	NaN	Тестирую какую-то новую приблуду телеграма-пу...	text	NaN	otsuka_bld	2015-09-22 21:19:46+00:00	60	[{'reaction': '👏', 'count': 5, 'chosen': False...	6	NaN	1004504016
5282294	19.0	1865.0	NaN	Круто	text	NaN	varlamov	2015-09-23 07:54:47+00:00	5	[{'reaction': '🔥', 'count': 3, 'chosen': False...	5	NaN	1005684212

In [28]:
```
data_sorted.set_index("datetime", inplace=True)
```

In [29]:
```
messages_by_month = data_sorted["_to_id"].resample("1M").count()
```

In [30]:
```
def plot_annotations(xlabel, ylabel, title):
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title);
```

In [31]:
```
fig, ax = plt.subplots(figsize=(9, 6))
sns.lineplot(messages_by_month)
plot_annotations("Time", "Number of messages",
                 "The number of messages from all the channels")
```



We have a general tendency to grow up and expected a dramatic jump in the number of messages after Russian-Ukrainian war beginning.

Q5-6: How many messages are per channel? How is the number of messages distributed per channel over time?

During next EDA questions, we will focus primarily on *the top 20 channels* by specific metrics to get an overall picture from data.
A more detailed explanation about limitations will be in *limitations*.
The number of messages can indicate the size of channel and existence time.

In [32]:
```
messages_per_channel = _data["channel"].value_counts()
messages_per_channel = pd.DataFrame(messages_per_channel).reset_index()
messages_per_channel
```

Out[32]:

	index	channel
0	karaulny	433223
1	glavmedia	214209
2	swodki	195695
3	rian_ru	187882
4	tass_agency	170539
...	...	...
2030	KOnOfff	1
2031	religiontoday	1
2032	RUS_peacekeeper	1
2033	sportrumours	1
2034	rcokmo	1

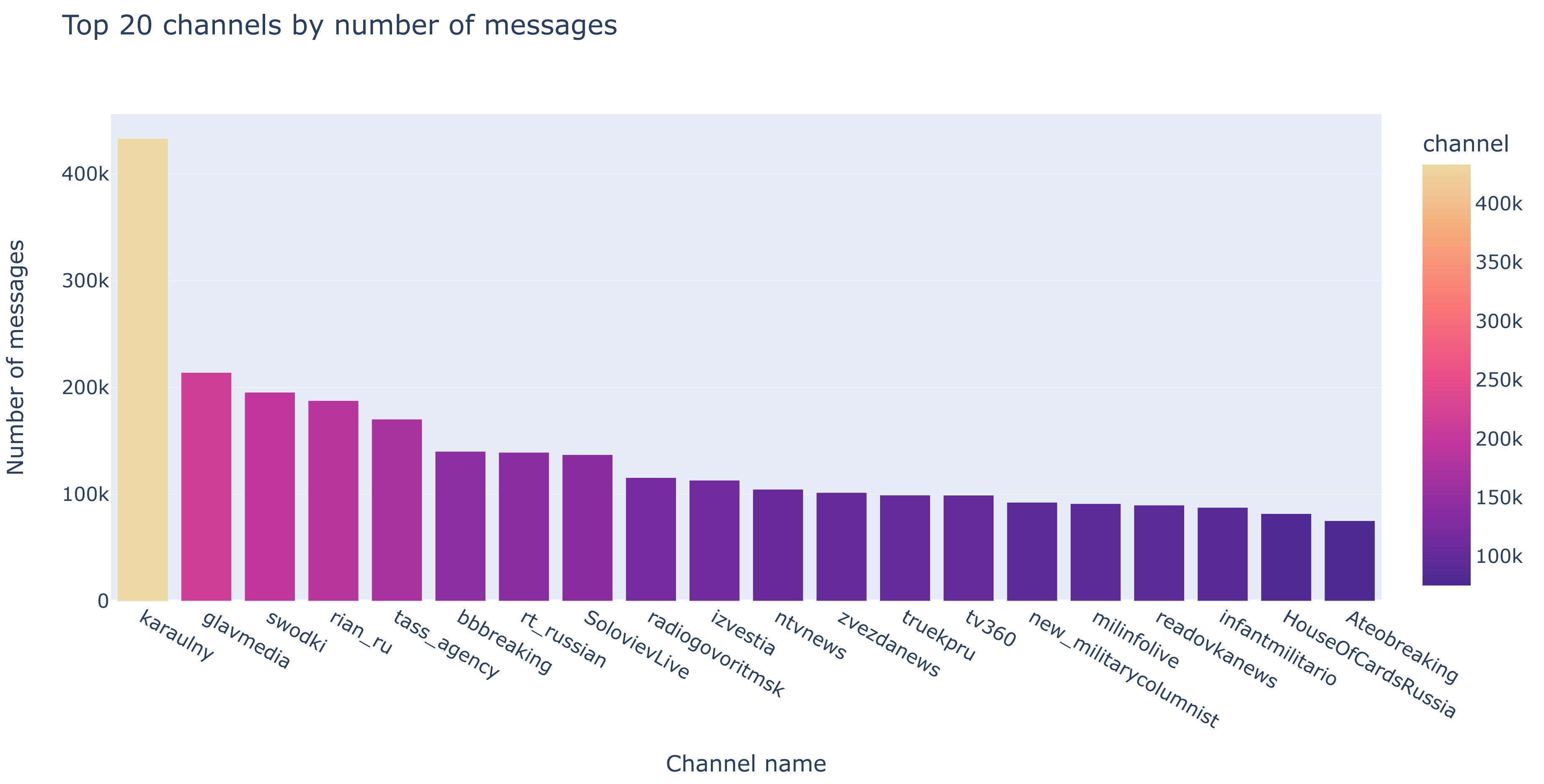
2035 rows × 2 columns

In [33]:
```
top20_by_messages = messages_per_channel[:20]["index"].tolist()
```

In [34]:
```
print(f"The overall number of messages from the top 20 channels by messages:"
      f" {messages_per_channel['channel'][:20].sum() / messages_num * 100:.2f}%")
```

```
The overall number of messages from the top 20 channels by messages: 45.79%
```

```
In [35]: px.bar(messages_per_channel[:20],
               x="index",
               y="channel",
               title="Top 20 channels by number of messages",
               color="channel",
               color_continuous_scale='Agsunset').update_layout(
           xaxis_title="Channel name",
           yaxis_title="Number of messages")
```



We can remark `karaulny` as an undisputable leader by number of messages. Other leaders in the top five are `glavmedia`, `swodki`, `rian_ru`, and `tass_agency`. These great number of messages in leaders can indicate *more maturity of channels, spam channels, or channels with great impact on the community*. We will understand it later after better exploration.

In [36]:
```python
messages_per_channel_new = pd.DataFrame(data_sorted["channel"].copy())
messages_per_channel_new["number"] = 1
messages_per_channel_new.head(3)
```

Out[36]:

	channel	number
datetime		
2015-09-22 16:12:02+00:00	varlamov	1
2015-09-22 21:19:46+00:00	otsuka_bld	1
2015-09-23 07:54:47+00:00	varlamov	1

In [37]:
```python
messages_per_channel_over_time = messages_per_channel_new.groupby("channel").resample("1M").sum()
messages_per_channel_over_time
```

Out[37]:

		number
channel	**datetime**	
AG_DPR	**2022-05-31 00:00:00+00:00**	1
	2022-06-30 00:00:00+00:00	0
	2022-07-31 00:00:00+00:00	0
	2022-08-31 00:00:00+00:00	0
	2022-09-30 00:00:00+00:00	3
...	**...**	...
zvezdanews	**2022-08-31 00:00:00+00:00**	3377
	2022-09-30 00:00:00+00:00	3281
	2022-10-31 00:00:00+00:00	3987
	2022-11-30 00:00:00+00:00	3188
	2022-12-31 00:00:00+00:00	2158

30439 rows × 1 columns

In [38]:
```
messages_per_channel_over_time = messages_per_channel_over_time.sort_index(level="datetime").reorder_levels(
    ["datetime", "channel"]).unstack().droplevel(level=0, axis=1)
messages_per_channel_over_time.head()
```

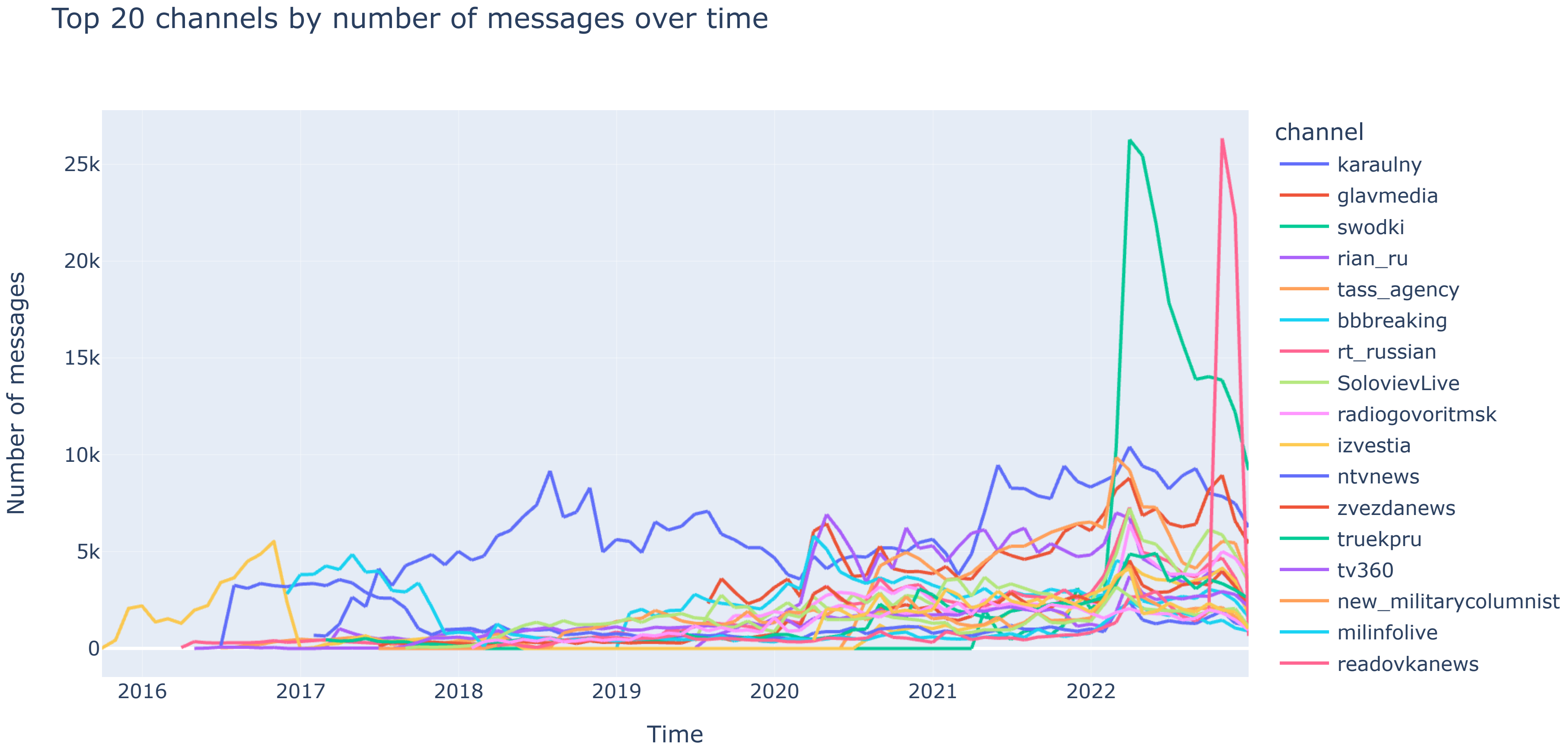
Out[38]:

channel	AG_DPR	AKID_channel	ARTolmachev	ASGasparyan	ATC_ATC	Abbasdjuma	Above_All_Public	Ad_Ping	AdvokatDyablo	Agdchan	...	zlucl
datetime												
2015-09-30 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2015-10-31 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2015-11-30 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2015-12-31 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
2016-01-31 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	

5 rows × 2035 columns

In [39]:
```
top20_by_messages_over_time = messages_per_channel_over_time[top20_by_messages]
```

In [40]:
```
px.line(top20_by_messages_over_time,
        title="Top 20 channels by number of messages over time").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of messages")
```



Again, we can remark `karaulny` as quite inconsistent in terms of the number of messages over time but overall a great amount as expected from the plot in Q5. We can also see that the oldest channels are `izvestia` , `ntvnews` , `rt_russian` , and `tv360` . They tend to become less popular in comparison with new big channels.
All channels suddenly explainably increased in March-April 2020 due to the coronavirus epidemic beginning.
Also, remarkable channels are `swodki` and `readovkanews` both have dramatic increases in March 2022 and October 2022 appropriate. It indicates a

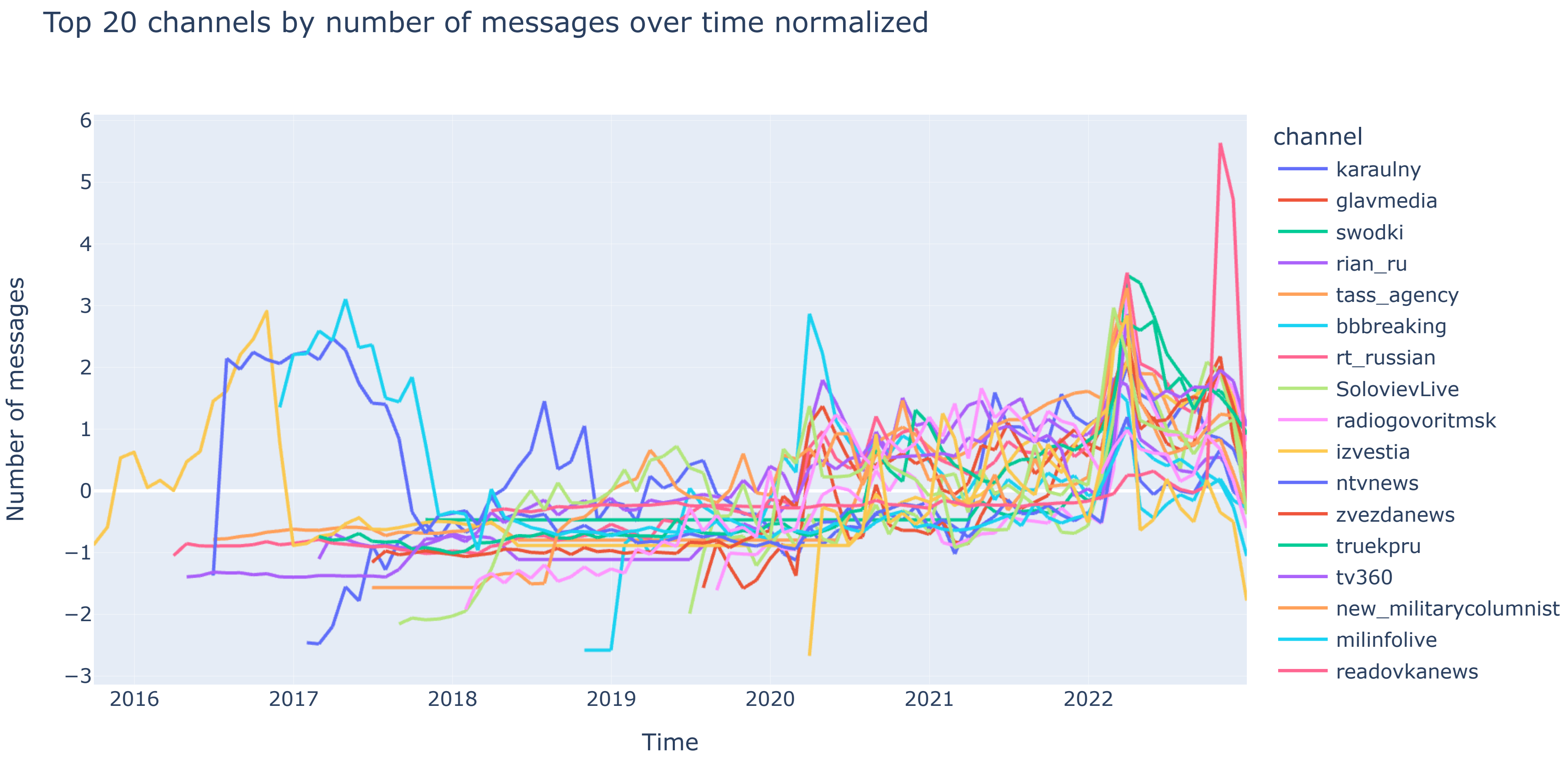
high probability that `swodki` is paid a channel. For `readovkanews` , it seems strange because they posted more than 25% of messages from all channel

Q7: What is normalized message distribution per channel over time?

We are interested in taking a look at a normalized number of message distributions for top channels over time. It can indicate unusual behavior regardless of the number of messages during the period for each channel.

In [41]:
```
top20_by_messages_over_time_norm = (top20_by_messages_over_time -
                                    top20_by_messages_over_time.mean()) / top20_by_messages_over_time.std()
```

In [42]:
```
px.line(top20_by_messages_over_time_norm,
        title="Top 20 channels by number of messages over time normalized").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of messages")
```



With the normalized version, we can remark *"died"* channels such as `izvestia`, `milinfolive`, `ntvnews` from the greatest by the number of messages over time. They tend to post in the early years of channel creation and approximately average the number of messages over time with natural fluctuations.

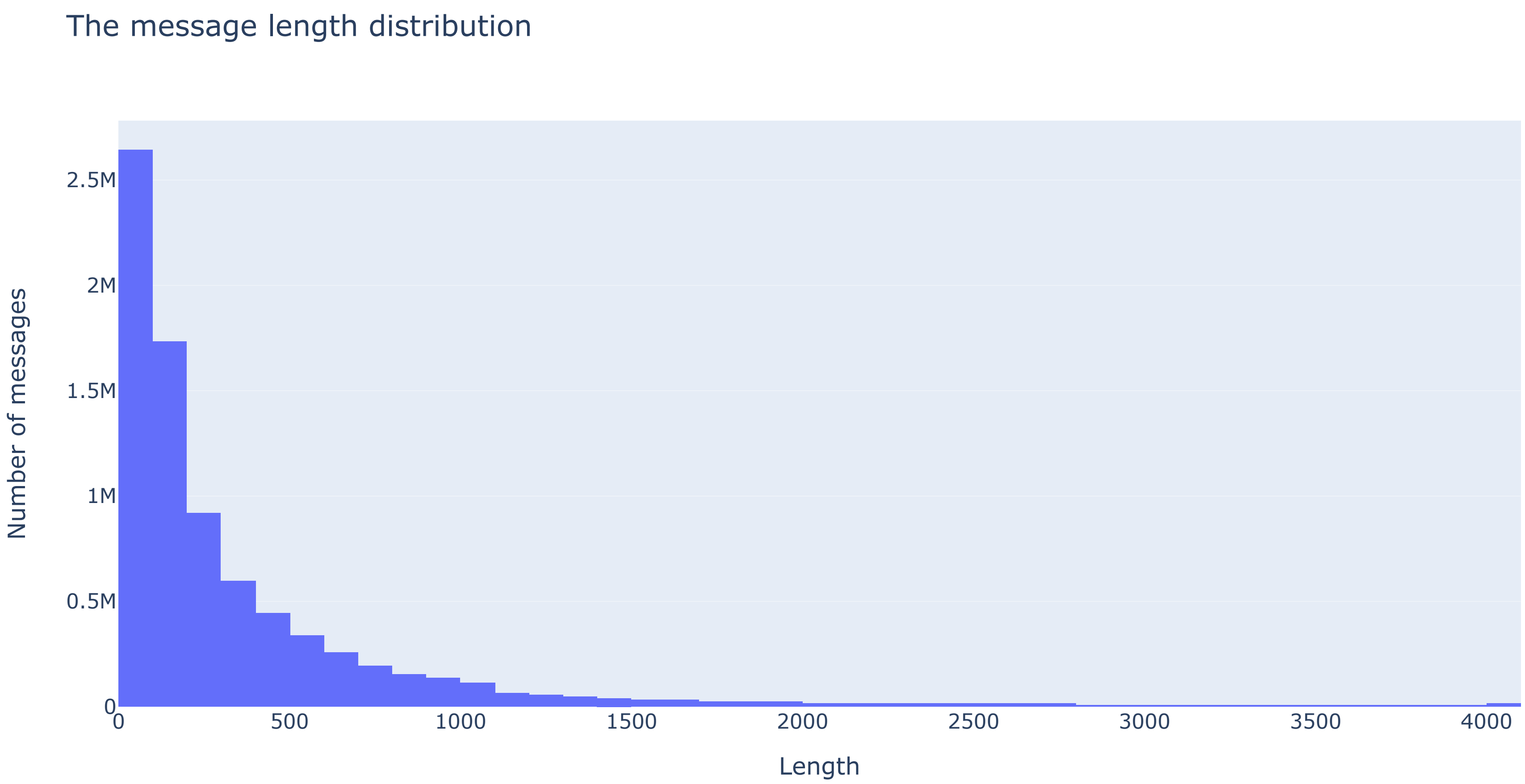
Q8: What is the length of message distribution?

We are also interested in taking a look at the length of message distribution over time. We try to capture unusual lengths and possibly indicate bots or the flow of similar messages among different channels.

In [43]:
```
messages_len = _data["message_len"].value_counts()
messages_len
```

Out[43]:
```
0       1041792
34        49855
90        23809
89        23557
91        23199
          ...
3687         30
3330         30
3689         29
3463         27
3725         20
Name: message_len, Length: 4097, dtype: int64
```

In [44]:
```
px.histogram(x=messages_len.index,
             y=messages_len,
             title="The message length distribution").update_layout(
    xaxis_title="Length",
    yaxis_title="Number of messages")
```

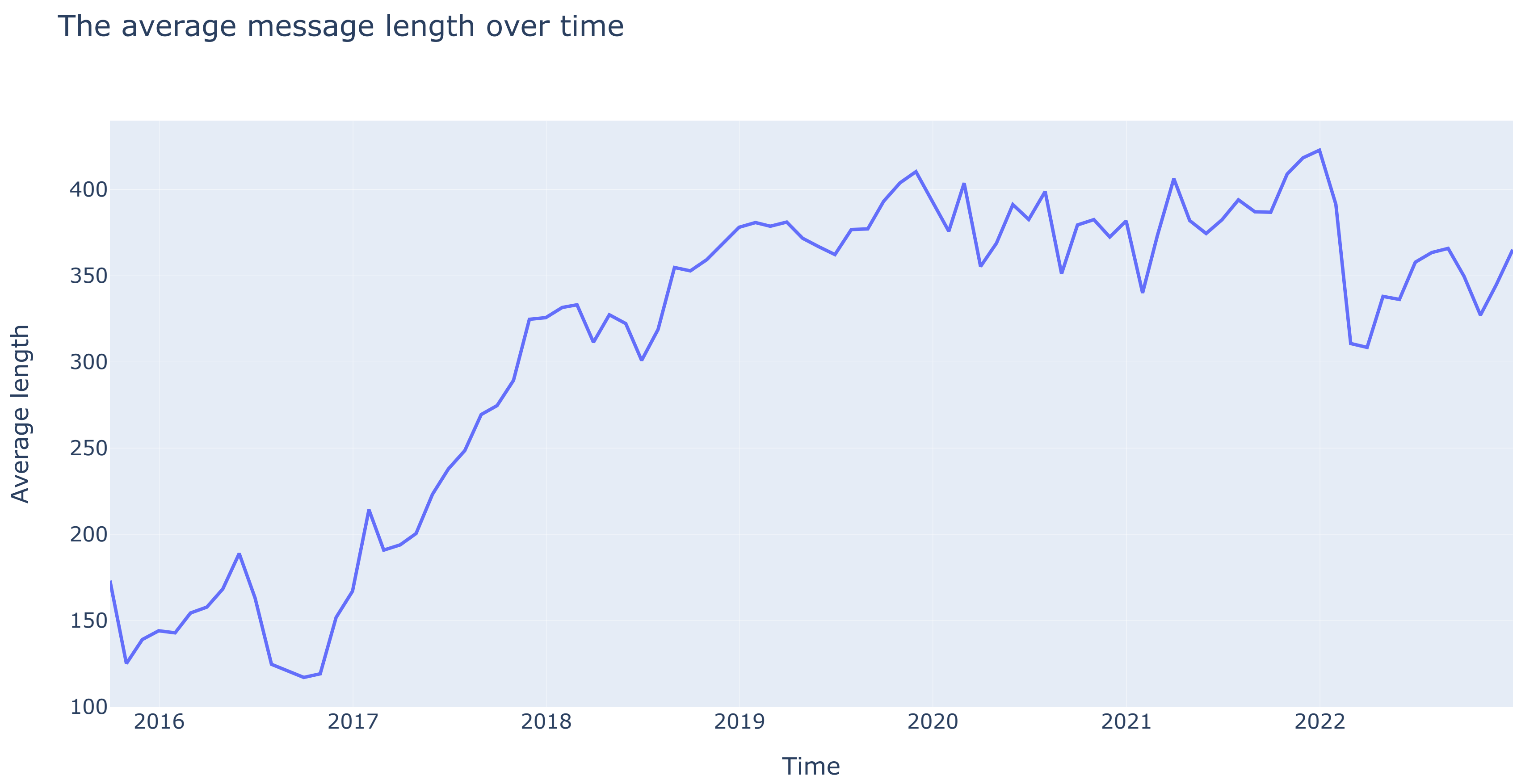


The plot has an exponential distribution. There are no unusual patterns.

In [45]:
```python
avg_messages_len_over_time = data_sorted["message_len"].resample("1M").mean()
avg_messages_len_over_time
```

Out[45]:
```
datetime
2015-09-30 00:00:00+00:00    173.082192
2015-10-31 00:00:00+00:00    124.960080
2015-11-30 00:00:00+00:00    138.917917
2015-12-31 00:00:00+00:00    144.012414
2016-01-31 00:00:00+00:00    142.808650
                                ...
2022-08-31 00:00:00+00:00    365.890684
2022-09-30 00:00:00+00:00    349.736278
2022-10-31 00:00:00+00:00    327.168505
2022-11-30 00:00:00+00:00    344.851138
2022-12-31 00:00:00+00:00    365.161868
Freq: M, Name: message_len, Length: 88, dtype: float64
```

In [46]:
```
px.line(avg_messages_len_over_time,
             title="The average message length over time").update_layout(
    xaxis_title="Time",
    yaxis_title="Average length",
    showlegend=False)
```



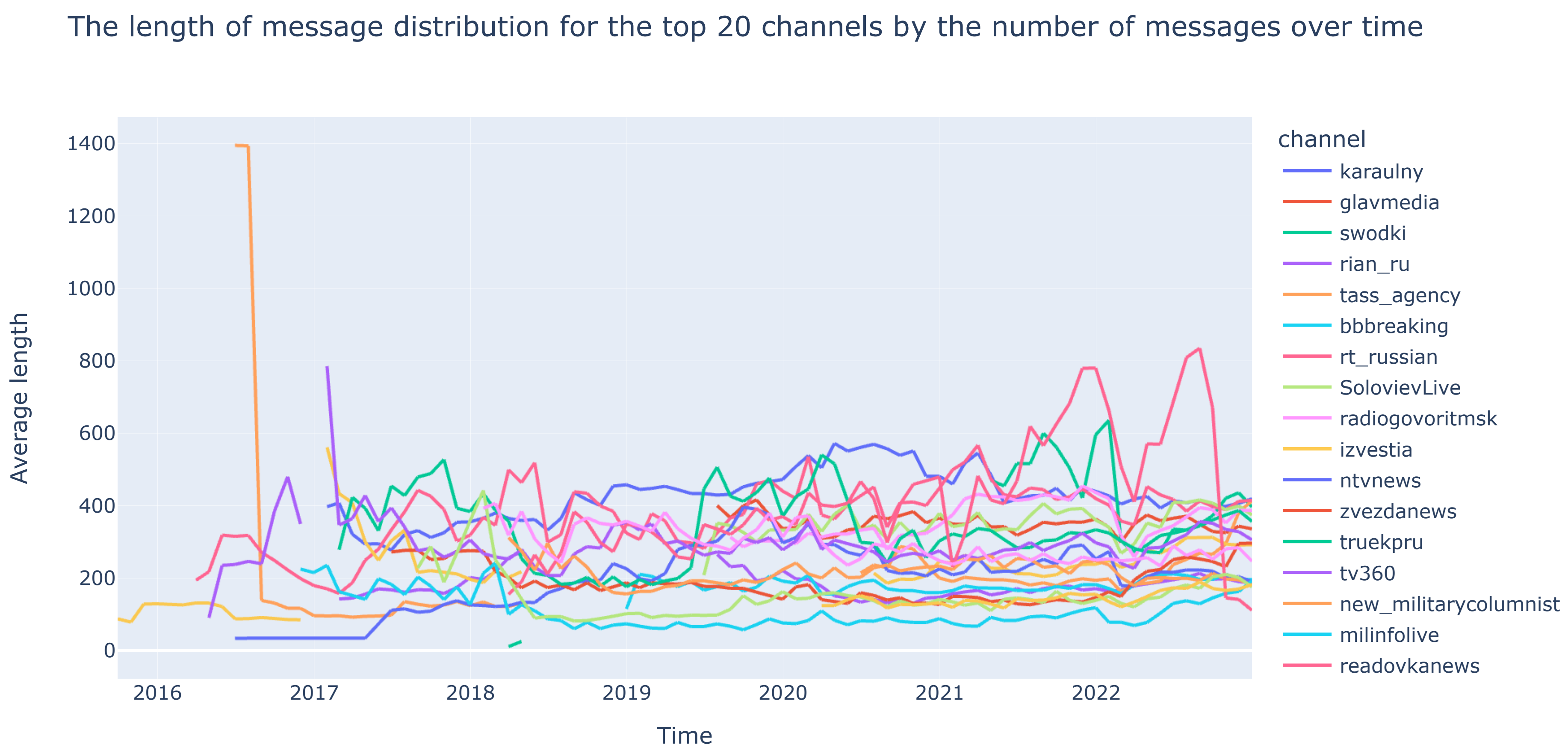
We have a general tendency to grow up over time. The growth of channels can explain the tendency to write longer messages.

Q9: What is the length of message distribution per channel over time?

Now, we explore Q8 more deeply for the greatest channels by the number of messages.

In [48]:
```
message_len_by_channel = data_sorted[["channel", "message_len"]].groupby("channel").resample("1M").mean()
message_len_by_channel = message_len_by_channel.sort_index(level="datetime").reorder_levels(
    ["datetime", "channel"]).unstack().droplevel(level=0, axis=1)
```

In [49]:
```
top20_by_messages_len_distribution = message_len_by_channel[top20_by_messages]
px.line(top20_by_messages_len_distribution,
        title="The length of message distribution for the top 20 channels by the number of messages over time").update
    xaxis_title="Time",
    yaxis_title="Average length")
```



We can see the quite consistent average length of messages except `readovkanews` with weird fluctuations that strengthen hypothesis about paidness above.

Q10: What is the distribution of the message by sensitive topic?

The message distribution by sensitive topic can define more aggressive message context.

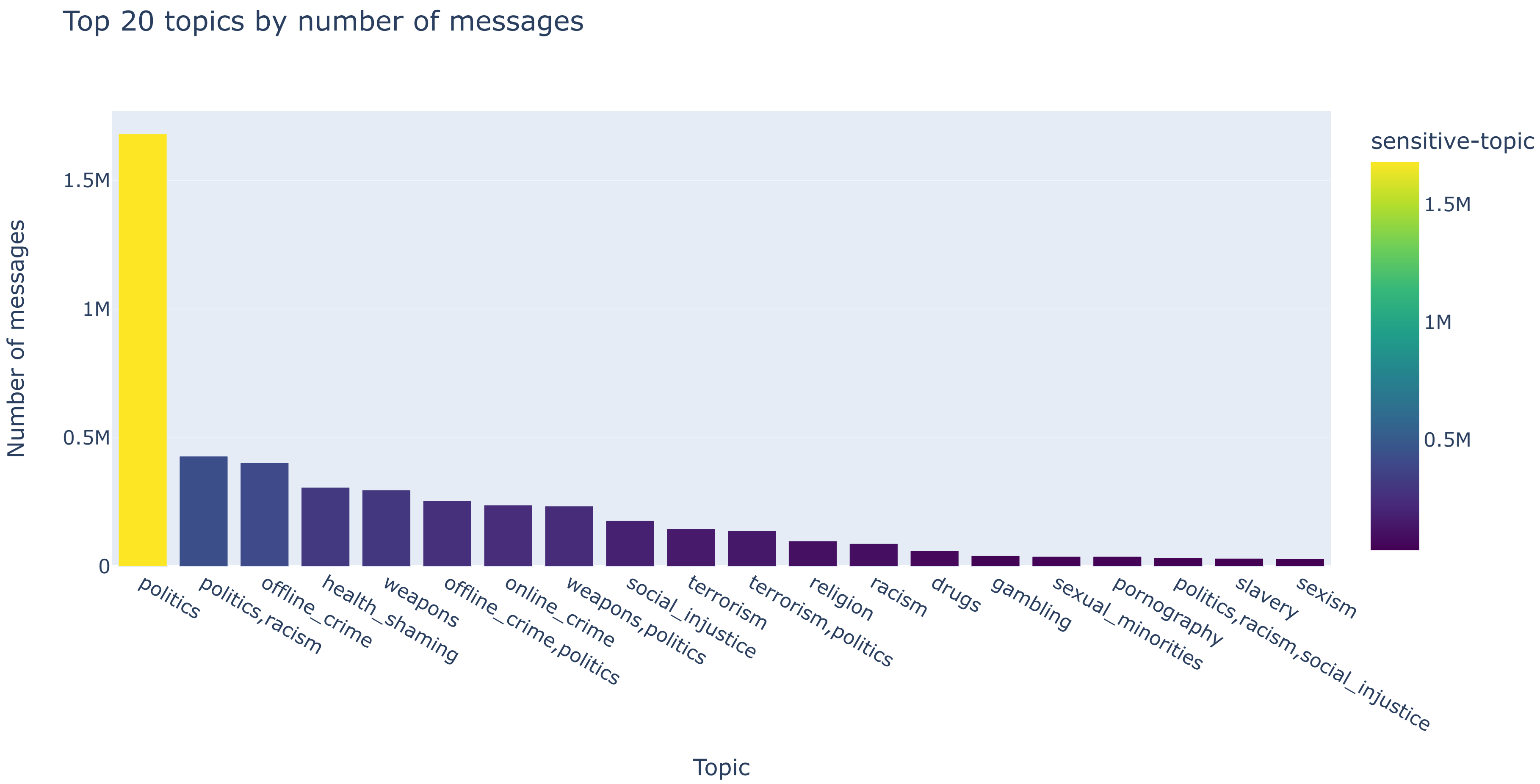
In [25]:
```
sensitive_distribution = pd.DataFrame(_data["sensitive-topic"].value_counts()).reset_index().drop(0)
sensitive_distribution
```

Out[25]:

	index	sensitive-topic
1	politics	1681172
2	politics,racism	428896
3	offline_crime	403535
4	health_shaming	307993
5	weapons	297420
...	...	...
64	offline_crime,terrorism,racism,religion	2
65	offline_crime,pornography,social_injustice	1
66	terrorism,politics,racism,social_injustice	1
67	pornography,slavery	1
68	body_shaming,racism	1

68 rows × 2 columns

In [26]:
```
px.bar(sensitive_distribution[:20],
       x="index",
       y="sensitive-topic",
       title="Top 20 topics by number of messages",
       color="sensitive-topic",
       color_continuous_scale='Viridis').update_layout(
    xaxis_title="Topic",
    yaxis_title="Number of messages")
```



We can see the expected outcome with the majority of messages related to `politics` or `crimes` topics from propagandistic channels. There are unusual `health_shaming` topic in the top 5 among others.
We also drop `None` values as unclassified and noise in general despite the great number.

Q11: What is the distribution of the message by sensitivity over time?

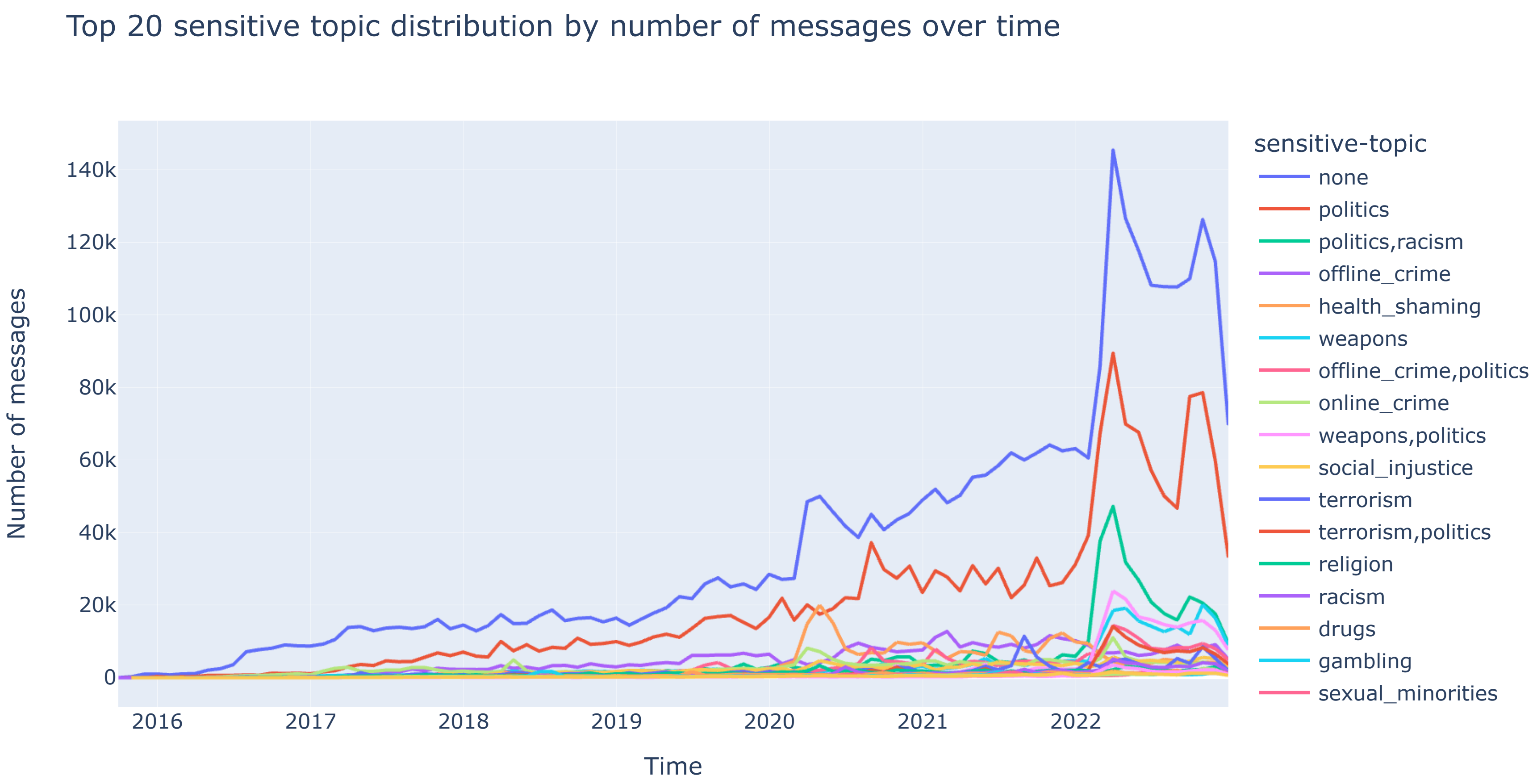
In [52]:
```
sensitive_over_time =  pd.DataFrame(data_sorted["sensitive-topic"]).groupby("sensitive-topic").resample("1M").count()
sensitive_over_time = sensitive_over_time.reorder_levels(
    ["datetime", "sensitive-topic"]).unstack().droplevel(level=0, axis=1)
sensitive_over_time.head()
```

Out[52]:

sensitive-topic datetime	body_shaming	body_shaming,health_shaming	body_shaming,racism	body_shaming,sexism	body_shaming,social_injustice	drugs	drugs,healtl
2015-09-30 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	NaN	
2015-10-31 00:00:00+00:00	NaN	NaN	NaN	NaN	NaN	8.0	
2015-11-30 00:00:00+00:00	1.0	NaN	NaN	NaN	NaN	18.0	
2015-12-31 00:00:00+00:00	0.0	NaN	NaN	NaN	NaN	13.0	
2016-01-31 00:00:00+00:00	5.0	NaN	NaN	NaN	NaN	9.0	

5 rows × 69 columns

In [53]:
```
top20_sensitive_over_time = sensitive_over_time[sensitive_distribution["index"][:20].tolist()]
px.line(top20_sensitive_over_time,
        title="Top 20 sensitive topic distribution by number of messages over time").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of messages")
```

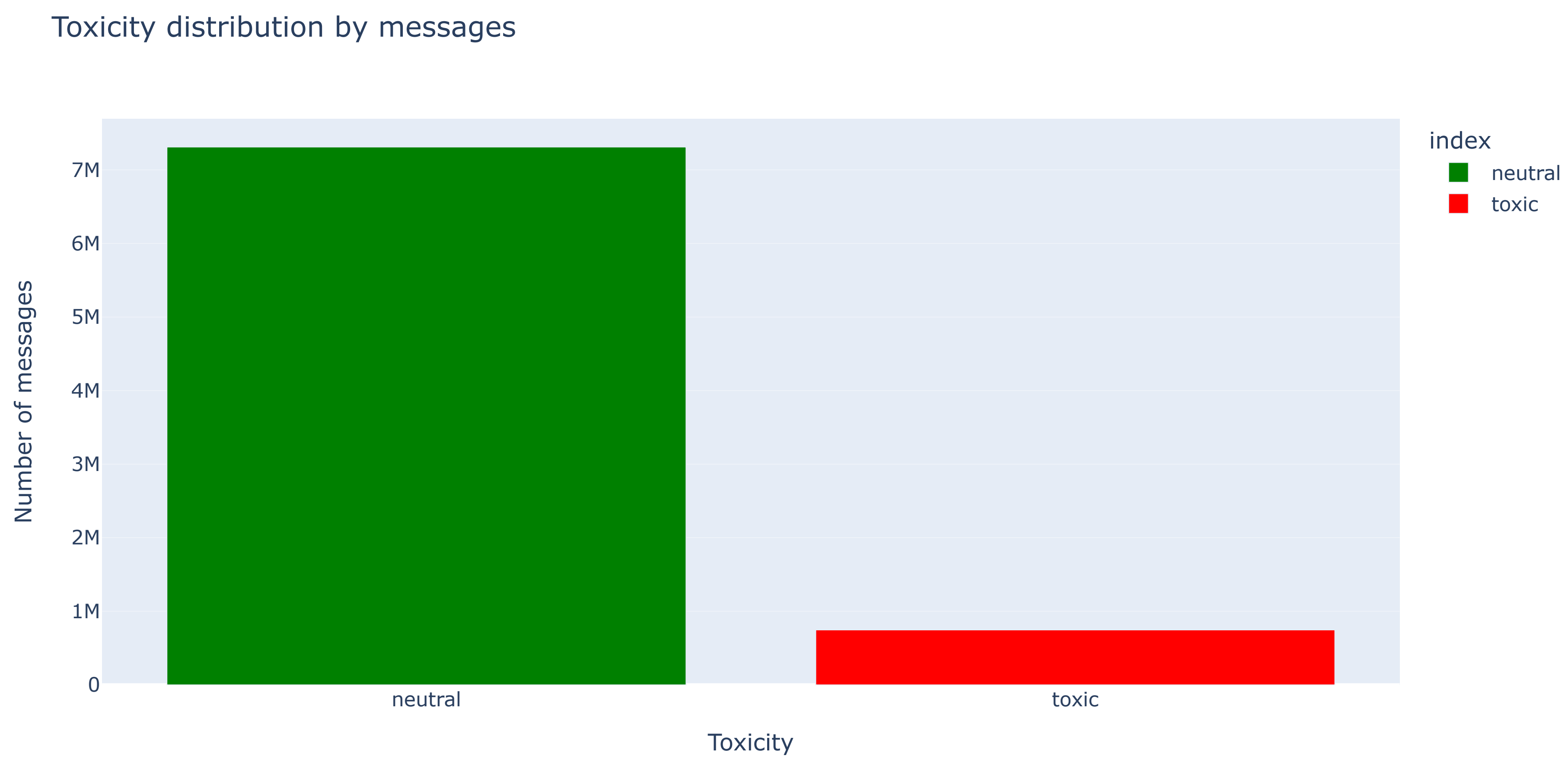


We can explainably see the growth of the `politics` topic over time. There are no unusual patterns.

Q12: What is the message distribution by toxicity?

The toxicity can show increasing of propaganda and the most propagandistic channels.

In [54]:
```
toxicity_distribution = pd.DataFrame(_data["toxicity"].value_counts()).reset_index()
px.bar(toxicity_distribution,
       x="index",
       y="toxicity",
       color="index",
       color_discrete_sequence=["green", "red"],
       title="Toxicity distribution by messages").update_layout(
    xaxis_title="Toxicity",
    yaxis_title="Number of messages")
```



The majority of messages have neutral toxicity but we will focus on explicitly toxic for channel behavior evaluation.

Q13: What is the message distribution by toxicity over time?

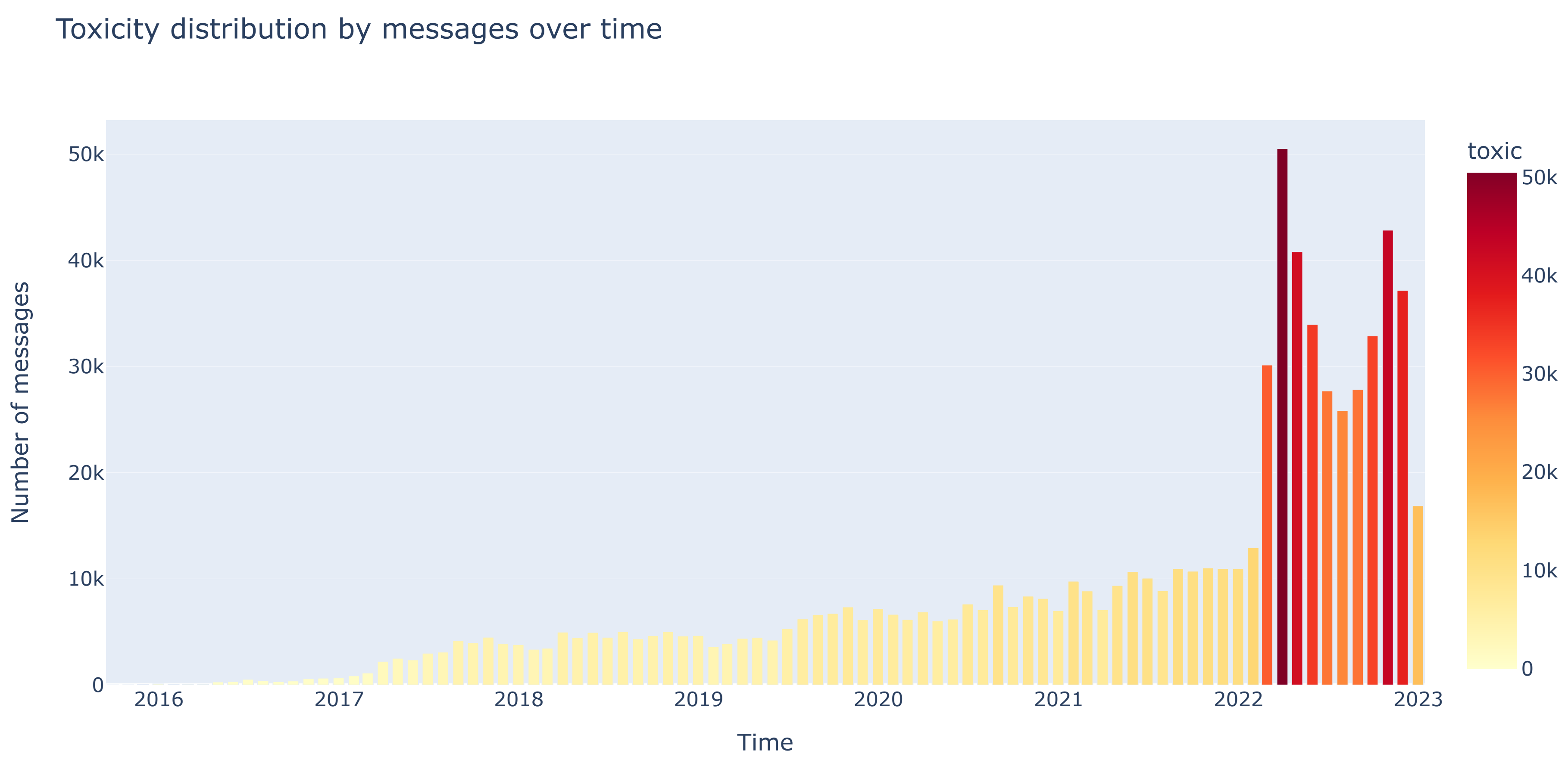
In [55]:
```
toxicity_over_time =  pd.DataFrame(data_sorted["toxicity"]).groupby("toxicity").resample("1M").count()
toxicity_over_time = toxicity_over_time.reorder_levels(
    ["datetime", "toxicity"]).unstack().droplevel(level=0, axis=1)
toxicity_over_time.head()
```

Out[55]:

toxicity	neutral	toxic
datetime		
2015-09-30 00:00:00+00:00	68	5
2015-10-31 00:00:00+00:00	491	10
2015-11-30 00:00:00+00:00	2077	55
2015-12-31 00:00:00+00:00	2237	99
2016-01-31 00:00:00+00:00	1488	38

In [56]:
```
toxic_messages = pd.DataFrame(toxicity_over_time["toxic"]).reset_index()
```

In [57]:
```
px.bar(toxic_messages,
       x="datetime",
       y="toxic",
       color="toxic",
       color_continuous_scale=px.colors.sequential.YlOrRd,
       title="Toxicity distribution by messages over time").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of messages")
```



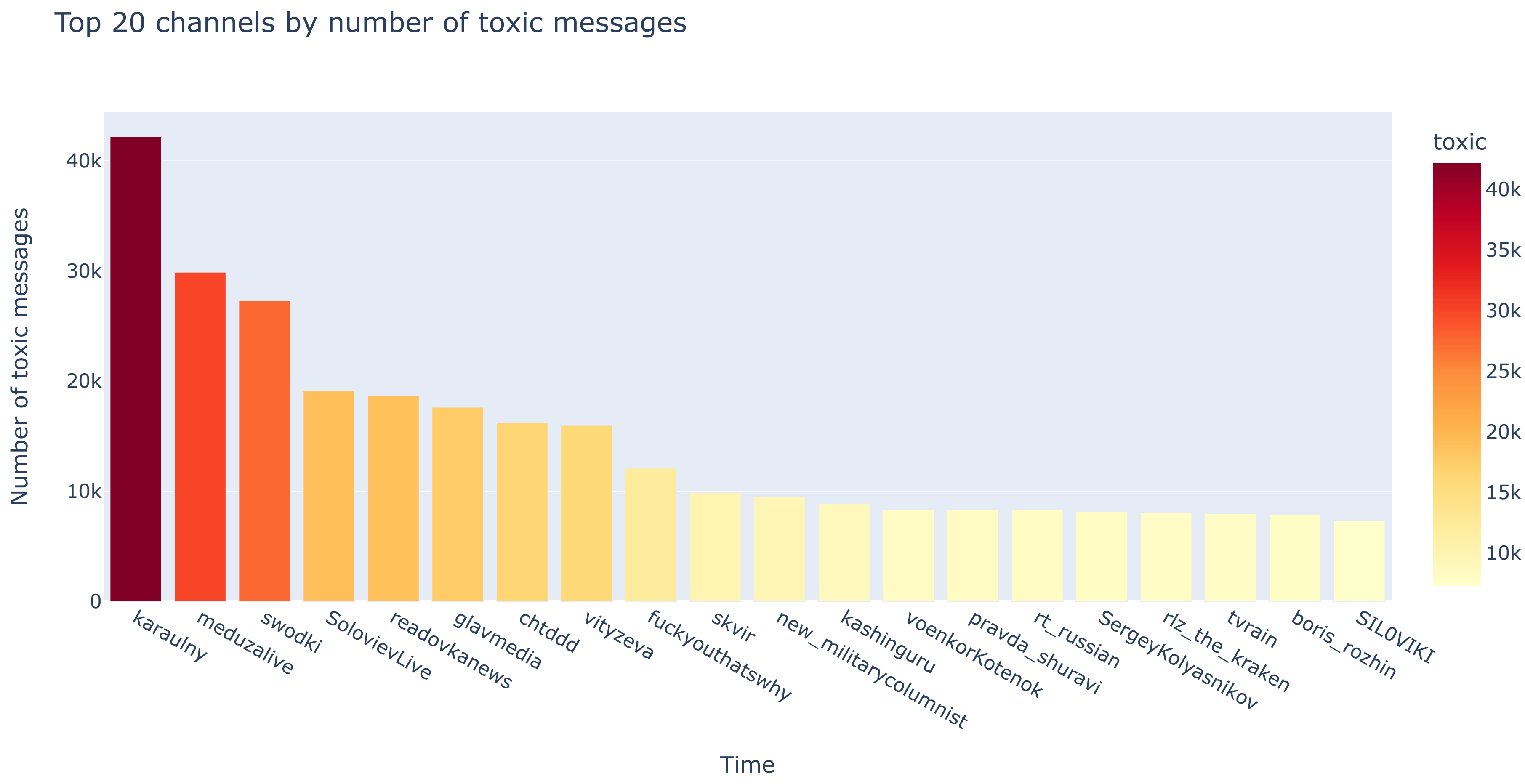
We can see the expected outcome with a tendency to toxicity growth and a dramatic jump after the beginning of the war. There are no unusual patterns.

Q14: Which channels are the most toxic by the number of toxic messages?

In [58]:
```
toxicity_by_channel = data_sorted[["channel", "toxicity"]].value_counts().unstack()["toxic"].sort_values(ascending=Fal
toxicity_by_channel.head()
```

Out[58]:
```
channel
karaulny        42193.0
meduzalive      29878.0
swodki          27294.0
SolovievLive    19095.0
readovkanews    18707.0
Name: toxic, dtype: float64
```

In [59]:
```
top20_toxic_channels = toxicity_by_channel[:20]
px.bar(pd.DataFrame(top20_toxic_channels).reset_index(),
       x="channel",
       y="toxic",
       color="toxic",
       color_continuous_scale=px.colors.sequential.YlOrRd,
       title="Top 20 channels by number of toxic messages").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of toxic messages")
```



We can remark `karaulny` is also an undisputable leader by the number of toxic messages. From the results earlier, it has the greatest number of messages so that's normal. The same is related to `swodki` , `SolovievLive` , `glavmedia` , and `rt_russian` .
There are unexpected other leaders such as `meduzalive` and `readovkanews` in the top 5. The first is not even in the top 20 channels by the number of

messages but it has the second place in toxicity rating. The second is almost last in the previous top but on of the leaders now. It can also indicate paidness.

Q15: What is the number of reactions distribution?

The reaction distributions can tell us about people's resonance with messages — primarily on toxic messages.

In [60]:
```
from collections import defaultdict
reactions_counter = defaultdict(int)
```

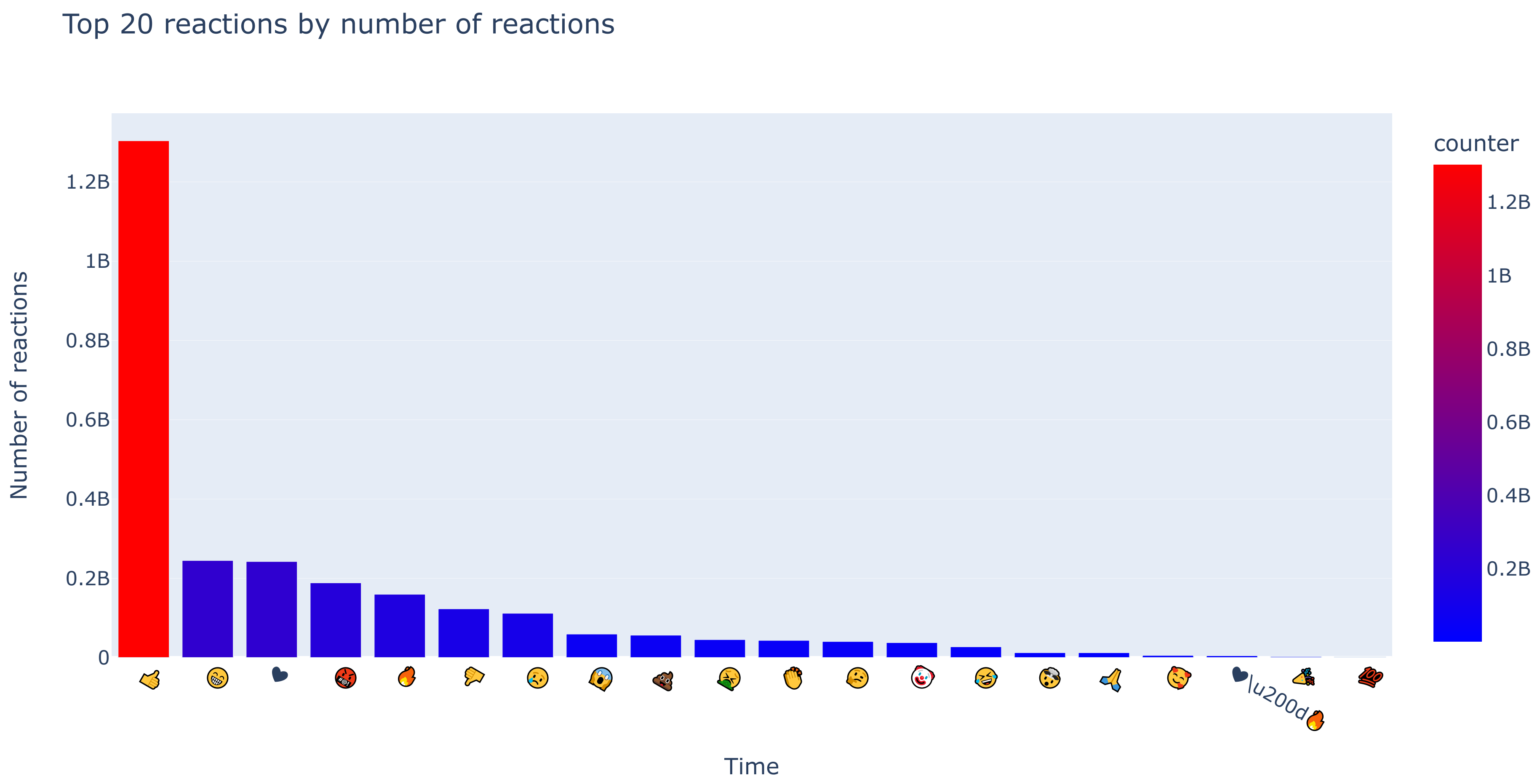
In [61]:
```
for reactions in _data["reactions_dict"][_data["reactions_dict"].apply(len) > 0]:
    for reaction in reactions:
        reactions_counter[reaction["reaction"]] += reaction["count"]
```

In [62]:
```
reactions_counter_df = pd.DataFrame(reactions_counter, index=["counter"]).transpose().sort_values(
    "counter", ascending=False)
reactions_counter_df.head()
```

Out[62]:

	counter
👍	1304025838
😁	245272286
❤	242675507
🤬	188818058
🔥	160038768

In [63]:
```
px.bar(reactions_counter_df.reset_index()[:20],
       x="index",
       y="counter",
       color="counter",
       color_continuous_scale=px.colors.sequential.Bluered,
       title="Top 20 reactions by number of reactions").update_layout(
    xaxis_title="Time",
    yaxis_title="Number of reactions")
```



Q16: What is the number of toxic reactions regarding toxic messages over time?

In [64]:
```python
reactions_on_toxic_message = _data[["datetime", "reactions_dict"]][_data["toxicity"] == "toxic"]
```

In [65]:
```python
reactions_on_toxic_message = reactions_on_toxic_message[reactions_on_toxic_message["reactions_dict"].apply(len) > 0]
reactions_on_toxic_message.head()
```

Out[65]:

	datetime	reactions_dict
327501	2022-12-20 10:07:01+00:00	[{'reaction': '👍', 'count': 280, 'chosen': Fal...
327515	2022-12-20 08:05:12+00:00	[{'reaction': '👍', 'count': 1089, 'chosen': Fa...
327549	2022-12-19 19:41:25+00:00	[{'reaction': '👍', 'count': 1833, 'chosen': Fa...
327562	2022-12-19 17:26:06+00:00	[{'reaction': '👍', 'count': 3074, 'chosen': Fa...
327564	2022-12-19 17:00:02+00:00	[{'reaction': '👍', 'count': 2467, 'chosen': Fa...

In [66]:
```python
reactions_on_toxic_message_df = pd.DataFrame(0, index=list(range(reactions_on_toxic_message.shape[0])),
                                             columns=reactions_counter_df.index.tolist())
reactions_on_toxic_message_df.head()
```

Out[66]:

	👍	😁	❤	🤬	🔥	👎	😢	😱	💩	🤮	...	😨	👨\u200d💻	🤝	😴	✍	🤗	😇	😒	😳	😂
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 57 columns

In [67]:
```python
for ind, reactions in enumerate(reactions_on_toxic_message["reactions_dict"]):
    for reaction in reactions:
        reactions_on_toxic_message_df.loc[ind, reaction["reaction"]] += reaction["count"]
```

In [68]:
```python
reactions_on_toxic_message_df.head()
```

Out[68]:

	👍	😁	❤	🤬	🔥	👎	😢	😱	💩	🤮	...	😨	👨\u200d💻	🤝	😴	✍	🤗	😇	😒	😳	😂
0	280	1	0	21	4	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1089	2	259	0	4	8	405	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1833	50	260	0	0	15	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3074	845	27	8	13	32	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	2467	0	4	412	36	7	12	18	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 57 columns

In [69]:
```python
reactions_on_toxic_message_df.set_index(reactions_on_toxic_message["datetime"], inplace=True)
```

In [70]:
```python
top_reaction_on_toxic_message = reactions_on_toxic_message_df.apply(lambda x: x.idxmax(axis="columns"), axis=1)
```

In [71]:
```python
reactions_on_toxic_message_df_final = pd.DataFrame(top_reaction_on_toxic_message.copy()).reset_index()
reactions_on_toxic_message_df_final["counter"] = 0
reactions_on_toxic_message_df_reset = reactions_on_toxic_message_df.copy().reset_index()
for ind in reactions_on_toxic_message_df_final.index:
    reactions_on_toxic_message_df_final.loc[ind, "counter"] = \
    reactions_on_toxic_message_df_reset.loc[ind, reactions_on_toxic_message_df_final.loc[ind, 0]]
reactions_on_toxic_message_df_final.head()
```

Out[71]:

	datetime	0	counter
0	2022-12-20 10:07:01+00:00	👍	280
1	2022-12-20 08:05:12+00:00	👍	1089
2	2022-12-19 19:41:25+00:00	👍	1833
3	2022-12-19 17:26:06+00:00	👍	3074
4	2022-12-19 17:00:02+00:00	👍	2467

In [72]:
```python
reactions_on_toxic_message_df_final = reactions_on_toxic_message_df_final.rename(columns={0: "reaction"}).set_index("c
```

In [73]:
```python
top10_reactions_on_toxic_messages = reactions_on_toxic_message_df_final.groupby("reaction").sum().sort_values("counter
top10_reactions_on_toxic_messages
```

Out[73]:

	counter
reaction	
👍	227812130
😁	45938776
🤬	35728074
❤	35365099
🔥	18763458
👎	15642011
😢	13237527
🤮	7669266
💩	7595682
😱	6319854

In [74]:
```python
reactions_on_toxic_message_df_final.shape
```

Out[74]:
```
(264641, 2)
```

In [77]:
```
rotm_df_final = reactions_on_toxic_message_df_final.groupby("reaction").resample("2W").sum().reorder_levels(
    ["datetime", "reaction"]).unstack().droplevel(level=0, axis=1)[top10_reactions_on_toxic_messages.index.tolist()]
rotm_df_final
```

Out[77]:

reaction	👍	😁	🤬	❤	🔥	👎	😢	🤮	💩	😱
datetime										
2015-09-27 00:00:00+00:00	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN
2015-10-11 00:00:00+00:00	1.0	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
2015-10-25 00:00:00+00:00	0.0	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
2015-11-08 00:00:00+00:00	0.0	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
2015-11-22 00:00:00+00:00	0.0	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
2022-12-11 00:00:00+00:00	8140108.0	1511614.0	NaN	786474.0	449860.0	776955.0	433844.0	NaN	156729.0	NaN
2022-12-18 00:00:00+00:00	NaN	NaN	1419464.0	NaN	NaN	NaN	NaN	185076.0	NaN	281501.0
2022-12-25 00:00:00+00:00	4444099.0	740712.0	NaN	514389.0	234038.0	353423.0	176587.0	NaN	98768.0	NaN
2023-01-01 00:00:00+00:00	NaN	NaN	52692.0	NaN	NaN	NaN	NaN	21838.0	NaN	24928.0
2023-01-08 00:00:00+00:00	NaN	NaN	NaN	305.0	2405.0	NaN	NaN	NaN	NaN	NaN

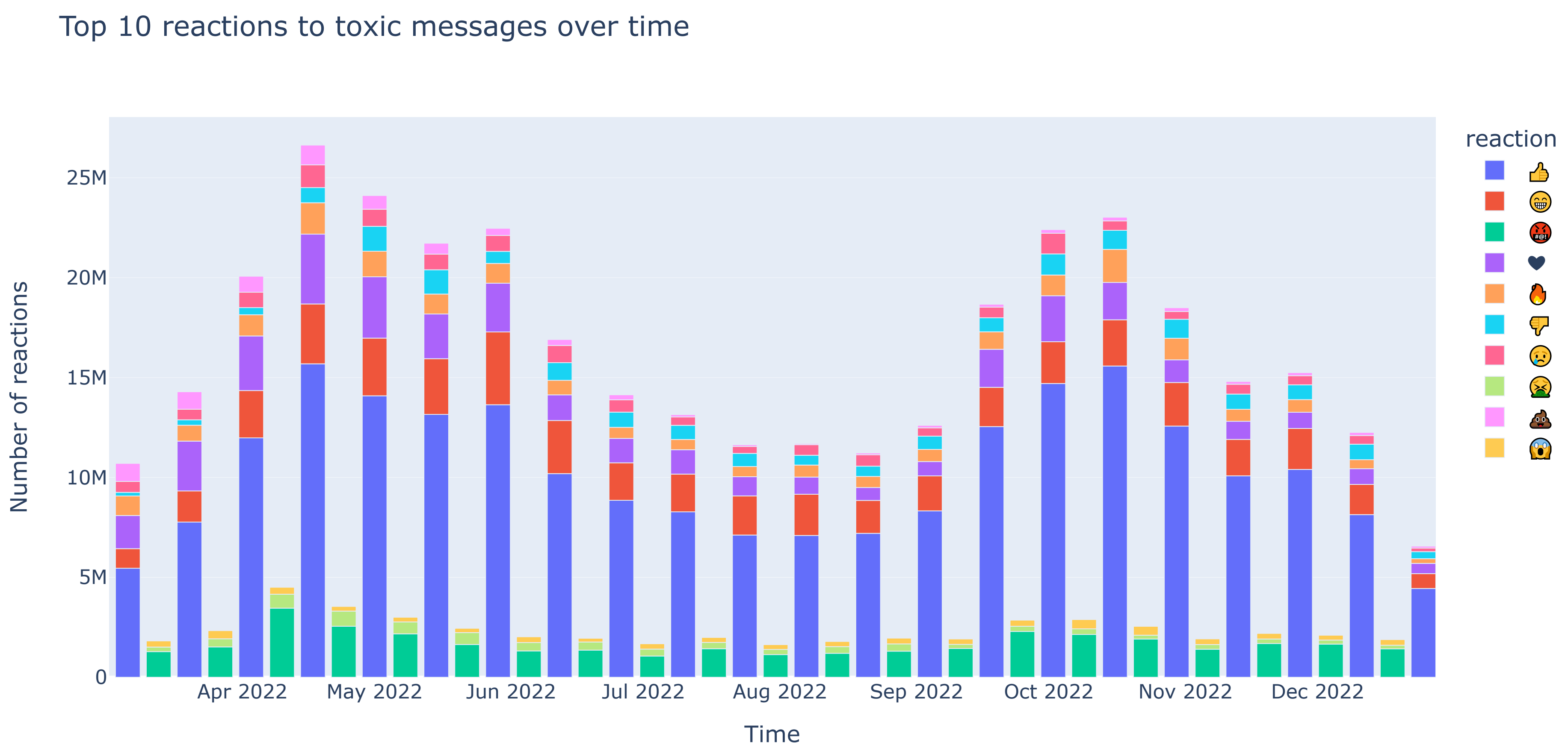
368 rows × 10 columns

In [78]:
```
_iqr = rotm_df_final.sum(axis=1).quantile(0.75) - rotm_df_final.sum(axis=1).quantile(0.25)
_mean = rotm_df_final.sum(axis=1).mean()
print(f"Interquartile range: {_iqr:.2f}\n",
      f"Mean: {_mean:.2f}\n",
      f"mean - 1.5*iqr: {_mean-1.5*_iqr:.2f}\n")
```

```
Interquartile range: 175.25
 Mean: 1125195.32
 mean - 1.5*iqr: 1124932.44
```

In [79]:
```
rotm_df_final = rotm_df_final[rotm_df_final.sum(axis=1) > 1124932]
```

```
In [81]: px.bar(rotm_df_final,
               title="Top 10 reactions to toxic messages over time").update_layout(
            xaxis_title="Time",
            yaxis_title="Number of reactions")
```



We can remark on the great number of positive toxic reaction in toxic messages. It means the majority of people react positively to russian propaganda.

Q17: Which channels have the most toxic auditory?

We will calculate metric of auditory toxicity by formula:

toxic coefficient = toxic reactions on toxic message / views on toxic message

In [82]:
```
toxic_reactions_on_toxic_message = [
    '👍', '😁', '❤', '🤬', '🔥', '👏', '🤣','🙏', '🥰', '🎉',
    '⚡', '🌚', '👌', '😍', '🍌', '😈','💋', '🤗', '😇', '😂']
```

In [83]:
```
toxic_reactions_on_toxic_message_num = reactions_on_toxic_message_df[toxic_reactions_on_toxic_message].sum(axis=1)
```

In [84]:
```
views_per_toxic_message = _data[["datetime", "views", "channel"]][(_data["toxicity"] == "toxic") & (_data["reactions_c
```

In [85]:
```
views_per_toxic_message["toxic_reaction"] = toxic_reactions_on_toxic_message_num.tolist()
views_per_toxic_message.head()
```

Out[85]:

	datetime	views	channel	toxic_reaction
327501	2022-12-20 10:07:01+00:00	18226.0	rt_russian	307
327515	2022-12-20 08:05:12+00:00	56732.0	rt_russian	1488
327549	2022-12-19 19:41:25+00:00	113352.0	rt_russian	2145
327562	2022-12-19 17:26:06+00:00	107156.0	rt_russian	4026
327564	2022-12-19 17:00:02+00:00	112156.0	rt_russian	2976

In [86]:
```
views_per_toxic_message = views_per_toxic_message.set_index("datetime").sort_index()
views_per_toxic_message.head()
```

Out[86]:

	views	channel	toxic_reaction
datetime			
2015-09-23 17:49:06+00:00	1792.0	varlamov	1
2015-09-25 15:14:57+00:00	1629.0	tvrain	3
2015-10-05 06:01:07+00:00	1387.0	tvrain	1
2015-12-22 12:41:20+00:00	170.0	izvestia	1
2015-12-28 14:11:28+00:00	411.0	holmogortalks	3

In [87]:
```
top20_by_toxic_reactions = views_per_toxic_message.groupby("channel").sum().sort_values(
    "toxic_reaction", ascending=False)[:20]
top20_by_toxic_reactions.head()
```

Out[87]:

	views	toxic_reaction
channel		
SolovievLive	2.420025e+09	49350209
boris_rozhin	1.689987e+09	30768815
RVvoenkor	1.398225e+09	27583199
nevzorovtv	7.983234e+08	24172402
RKadyrov_95	6.794709e+08	22972179

In [88]:
```
views_per_toxic_message = views_per_toxic_message.groupby("channel").resample("1M").sum()
views_per_toxic_message.head()
```

Out[88]:

		views	toxic_reaction
channel	**datetime**		
AG_DPR	**2022-09-30 00:00:00+00:00**	135762.0	1839
ASGasparyan	**2019-08-31 00:00:00+00:00**	66100.0	4
	2019-09-30 00:00:00+00:00	0.0	0
	2019-10-31 00:00:00+00:00	11466.0	3
	2019-11-30 00:00:00+00:00	0.0	0

In [89]:
```
top20_by_toxic_reactions_final = pd.merge(pd.DataFrame(top20_by_toxic_reactions.index), views_per_toxic_message.reset_
top20_by_toxic_reactions_final.head()
```

Out[89]:

	channel	datetime	views	toxic_reaction
0	SolovievLive	2019-06-30 00:00:00+00:00	151544.0	31
1	SolovievLive	2019-07-31 00:00:00+00:00	71375.0	17
2	SolovievLive	2019-08-31 00:00:00+00:00	40399.0	35
3	SolovievLive	2019-09-30 00:00:00+00:00	176371.0	16
4	SolovievLive	2019-10-31 00:00:00+00:00	28900.0	7

In [90]:
```
top20_by_toxic_reactions_final["toxic_perc"] = top20_by_toxic_reactions_final["toxic_reaction"] / top20_by_toxic_react
top20_by_toxic_reactions_final.head()
```

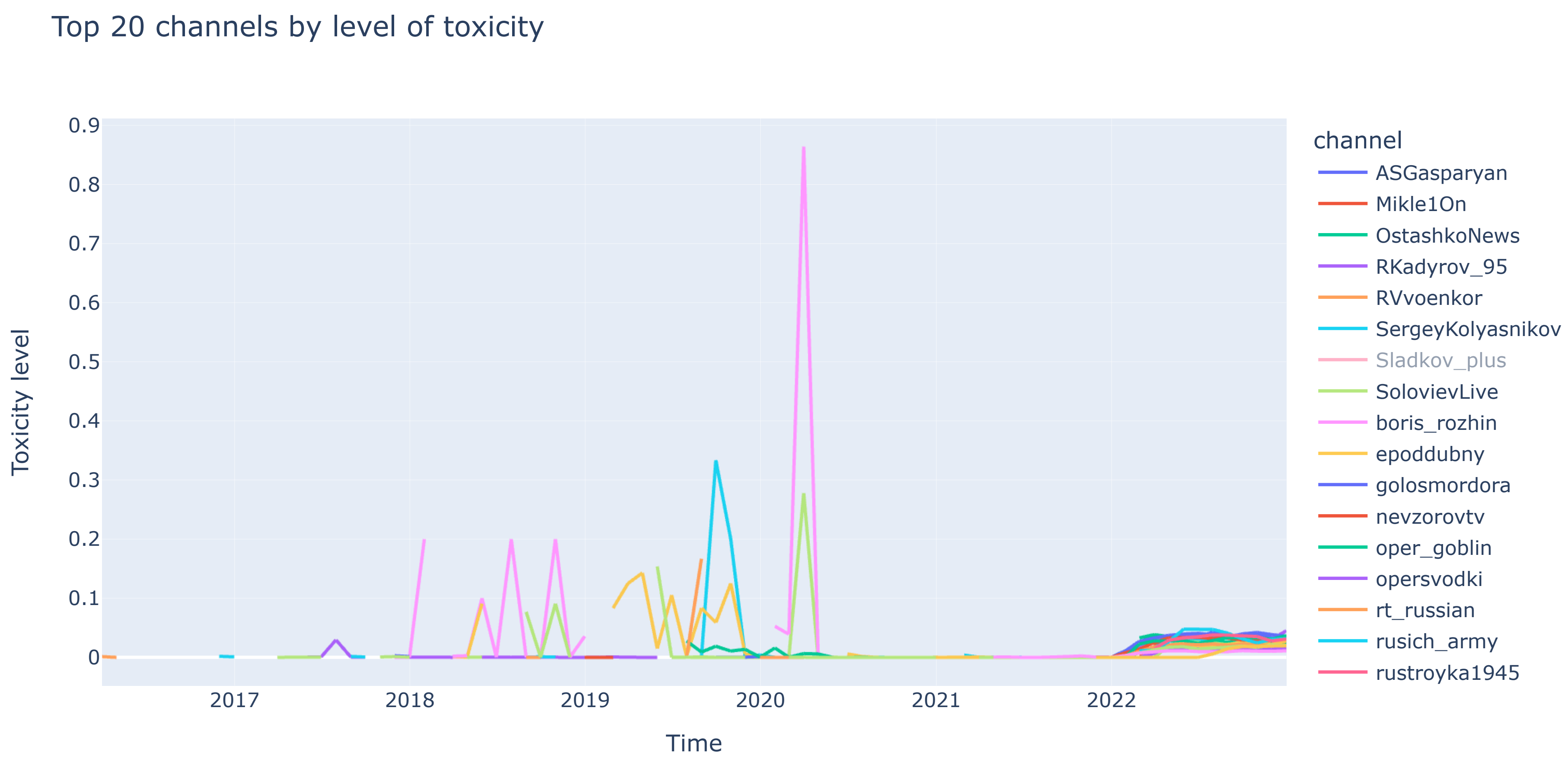
Out[90]:

	channel	datetime	views	toxic_reaction	toxic_perc
0	SolovievLive	2019-06-30 00:00:00+00:00	151544.0	31	0.000205
1	SolovievLive	2019-07-31 00:00:00+00:00	71375.0	17	0.000238
2	SolovievLive	2019-08-31 00:00:00+00:00	40399.0	35	0.000866
3	SolovievLive	2019-09-30 00:00:00+00:00	176371.0	16	0.000091
4	SolovievLive	2019-10-31 00:00:00+00:00	28900.0	7	0.000242

In [91]:
```
top20_by_toxic_reactions_final.drop(["views", "toxic_reaction"], axis=1, inplace=True)
```

In [92]:
```
top20_by_toxic_reactions_final_df = pd.pivot_table(top20_by_toxic_reactions_final, index=['datetime', 'channel']).unst
```

```
In [93]: px.line(top20_by_toxic_reactions_final_df,
                title="Top 20 channels by level of toxicity").update_layout(
             xaxis_title="Time",
             yaxis_title="Toxicity level")
```



We can remark high periodical very toxic resonance fluctuations for `boris_rozhin` and `SergiyKolyasnikov` both also at the top by number of toxic messages.
There are also leaders from previous tops such as `SolovievLive` and `rt_russian`.
The other channels need more exploration regarding the number of messages and overall impact.

Q18-19: How many views do we have overall? What is the number of views distribution over time?

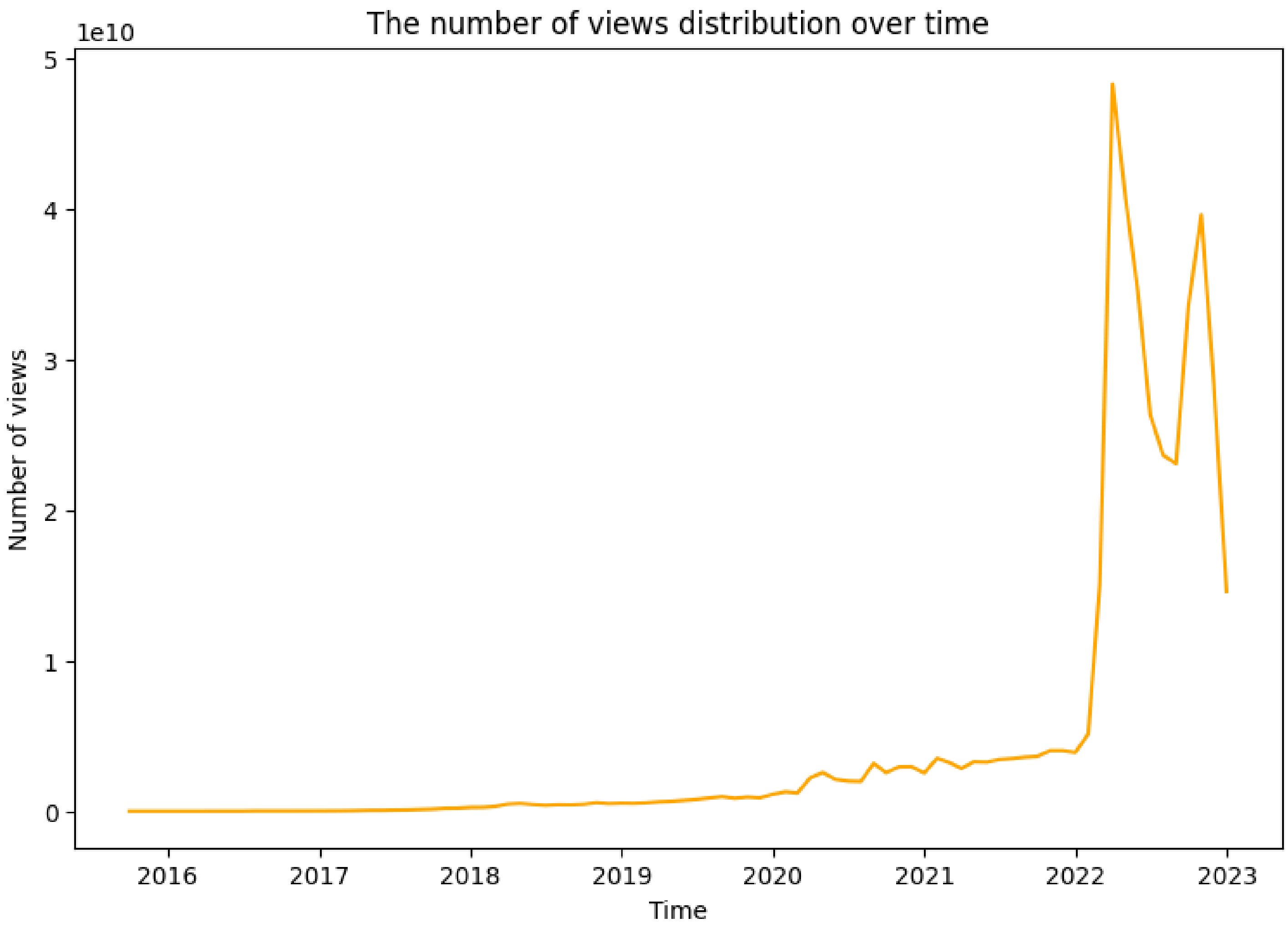
The views can show us a level of active auditory on channels. Some channels can send many messages but do have not many active readers.

In [94]:
```
views_num = _data["views"].sum()
print(f"The number of views for all time: {views_num / 10e9:.2f} billions")
```

```
The number of views for all time: 41.90 billions
```

In [95]:
```
views_by_month = data_sorted["views"].resample("1M").sum()

fig, ax = plt.subplots(figsize=(9, 6))
sns.lineplot(views_by_month, color="orange")
plot_annotations("Time", "Number of views",
                 "The number of views distribution over time")
```

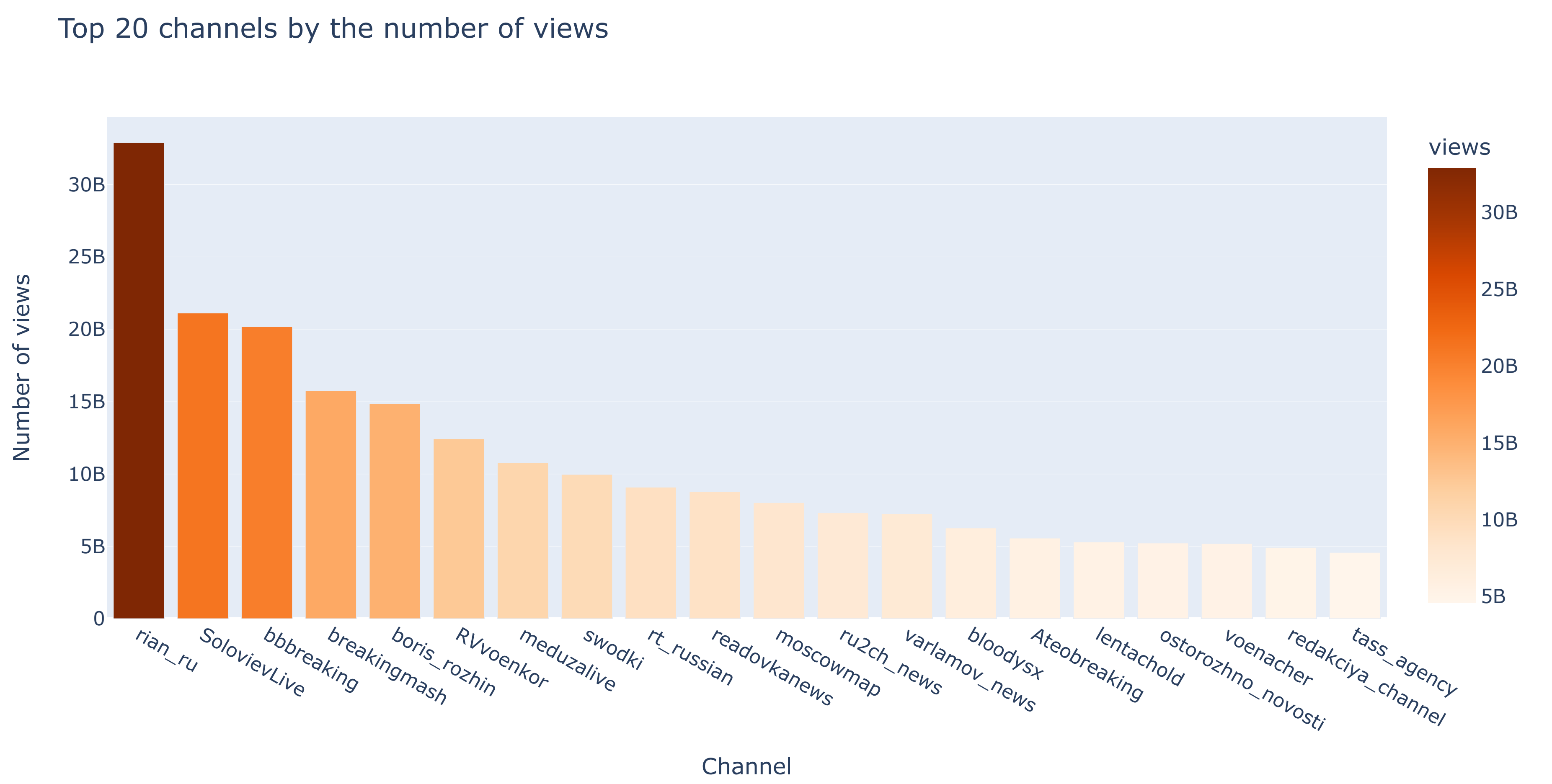


We can see the expected outcome with a tendency to views growth and a dramatic jump after the beginning of the war. There are no unusual patterns.

Q20-21: How many views are per channel? What is the number of views distribution per channel?

```
In [96]: views_by_channel = data_sorted[["channel", "views"]].groupby("channel").sum()
```

```
In [97]: top20_by_views = views_by_channel.sort_values("views", ascending=False)[:20].reset_index()
px.bar(top20_by_views,
       x="channel",
       y="views",
       color="views",
       color_continuous_scale=px.colors.sequential.Oranges,
       title="Top 20 channels by the number of views").update_layout(
    xaxis_title="Channel",
    yaxis_title="Number of views")
```



We can remark the several leaders `rian_ru`, `SolovievLive`, `bbbreaking`, `swodki`, `rt_russian`, `readovkanews`, and `tass_agency` that also leaders by the number of messages.
The `boris_rozhin`, `swodki`, `SolovievLive`, `rt_russian`, `meduzalive`, and `readovkanews` are also leaders in the top by the number of toxic messages.
The `boris_rozhin`, `SolovievLive`, and `rt_russian` are also leaders at the top by the level of auditory toxicity.
These channels are the greatest overall.
Surprisingly, `karaulny` is absent in the top.
The other channels need more exploration regarding the number of messages and overall impact.

In [111]:
```
print(f"The overall number of views from the top 20 channels by views:"
      f" {top20_by_views['views'].sum() / views_num * 100:.2f}%")
```

```
The overall number of views from the top 20 channels by views: 51.47%
```

Q22: What are the top channels by views per message?

The views per message indicate more significance from one message on the channel.

In [98]:
```
views_per_message_by_channel = pd.merge(views_by_channel.reset_index(),
                                       messages_per_channel,
                                       left_on="channel", right_on="index")
views_per_message_by_channel.head()
```

Out[98]:

	channel_x	views	index	channel_y
0	AG_DPR	2.088561e+06	AG_DPR	5
1	AKID_channel	1.914200e+04	AKID_channel	6
2	ARTolmachev	6.989800e+04	ARTolmachev	4
3	ASGasparyan	1.435679e+09	ASGasparyan	30800
4	ATC_ATC	3.300000e+01	ATC_ATC	6

In [99]:
```
views_per_message_by_channel = views_per_message_by_channel.drop("index", axis=1).rename(columns={"channel_y": "messag
views_per_message_by_channel.head()
```

Out[99]:

	channel_x	views	messages
0	AG_DPR	2.088561e+06	5
1	AKID_channel	1.914200e+04	6
2	ARTolmachev	6.989800e+04	4
3	ASGasparyan	1.435679e+09	30800
4	ATC_ATC	3.300000e+01	6

In [100]:
```
_data[_data["channel"] == "AG_DPR"]
```

Out[100]:

	id	views	fwd_from	message	type	duration	channel	datetime	message_len	reaction
1027519	8598.0	719876.0	MessageFwdHeader(date=datetime.datetime(2022, ...	Республика ищет талантливых управленцев. Это м...	video	34.0	AG_DPR	2022-10-04 10:43:56+00:00	139	
1027891	8203.0	573540.0	MessageFwdHeader(date=datetime.datetime(2022, ...	Военные корреспонденты – это бойцы, воюющие ...	photo	NaN	AG_DPR	2022-09-10 09:25:56+00:00	573	
1029147	6896.0	542623.0	MessageFwdHeader(date=datetime.datetime(2022, ...	«Вы их били, а мы их добьем! Победа будет за н...	video	374.0	AG_DPR	2022-05-09 08:30:19+00:00	186	
1053857	41905.0	116760.0	MessageFwdHeader(date=datetime.datetime(2022, ...	Демилитаризация вооруженных формирований Украи...	video	130.0	AG_DPR	2022-09-13 13:10:34+00:00	269	[{'rea '👍', ' 1214, 'ch
1054084	41677.0	135762.0	MessageFwdHeader(date=datetime.datetime(2022, ...	Самый тупой укрофейк...	photo	NaN	AG_DPR	2022-09-10 19:31:42+00:00	21	[{'rea '👍', ' 1004, 'ch

In [101]:
```
views_per_message_by_channel["views_per_message"] = views_per_message_by_channel["views"] / \
views_per_message_by_channel["messages"]
views_per_message_by_channel.head()
```

Out[101]:

	channel_x	views	messages	views_per_message
0	AG_DPR	2.088561e+06	5	417712.200000
1	AKID_channel	1.914200e+04	6	3190.333333
2	ARTolmachev	6.989800e+04	4	17474.500000
3	ASGasparyan	1.435679e+09	30800	46612.957208
4	ATC_ATC	3.300000e+01	6	5.500000

In [102]:
```
_iqr_1 = views_per_message_by_channel["messages"].quantile(0.75) - views_per_message_by_channel["messages"].quantile((
_mean_1 = views_per_message_by_channel["messages"].mean()
print(f"Interquartile range: {_iqr_1:.2f}\n",
      f"Mean: {_mean_1:.2f}\n",
      f"mean - 1.5*iqr: {_mean_1-1.5*_iqr_1:.2f}\n")
```

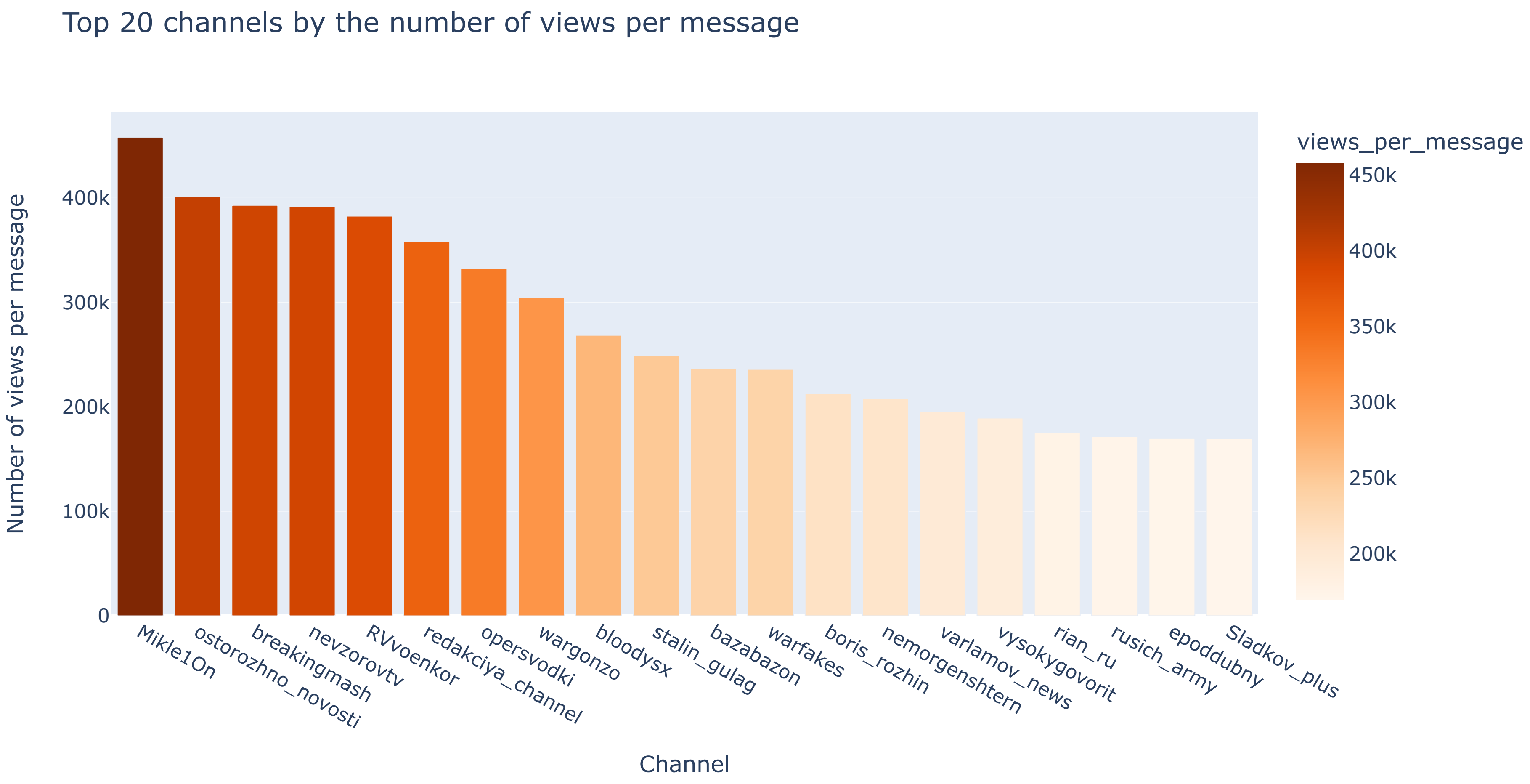
```
Interquartile range: 10.00
 Mean: 3959.73
 mean - 1.5*iqr: 3944.73
```

In [103]:
```
views_per_message_by_channel_final = views_per_message_by_channel[views_per_message_by_channel["messages"] > 3945]
views_per_message_by_channel_final.head()
```

Out[103]:

	channel_x	views	messages	views_per_message
3	ASGasparyan	1.435679e+09	30800	46612.957208
5	Abbasdjuma	1.075527e+08	11965	8988.940326
10	Alekhin_Telega	9.174188e+07	5798	15823.021042
25	Ateobreaking	5.582477e+09	75460	73979.285675
29	Baronova	7.413764e+07	21410	3462.757590

```
In [104]: top20_by_views_per_message = views_per_message_by_channel_final.sort_values("views_per_message", ascending=False)[:20]
          px.bar(top20_by_views_per_message,
                 x="channel_x",
                 y="views_per_message",
                 color="views_per_message",
                 color_continuous_scale=px.colors.sequential.Oranges,
                 title="Top 20 channels by the number of views per message").update_layout(
              xaxis_title="Channel",
              yaxis_title="Number of views per message")
```



We can remark the several leaders `boris_rozhin` and `rian_ru` that are also the leaders in other tops. The channel `epoddubny` is also in the leaders by the level of auditory toxicity.

Q23: How many views per message over time?

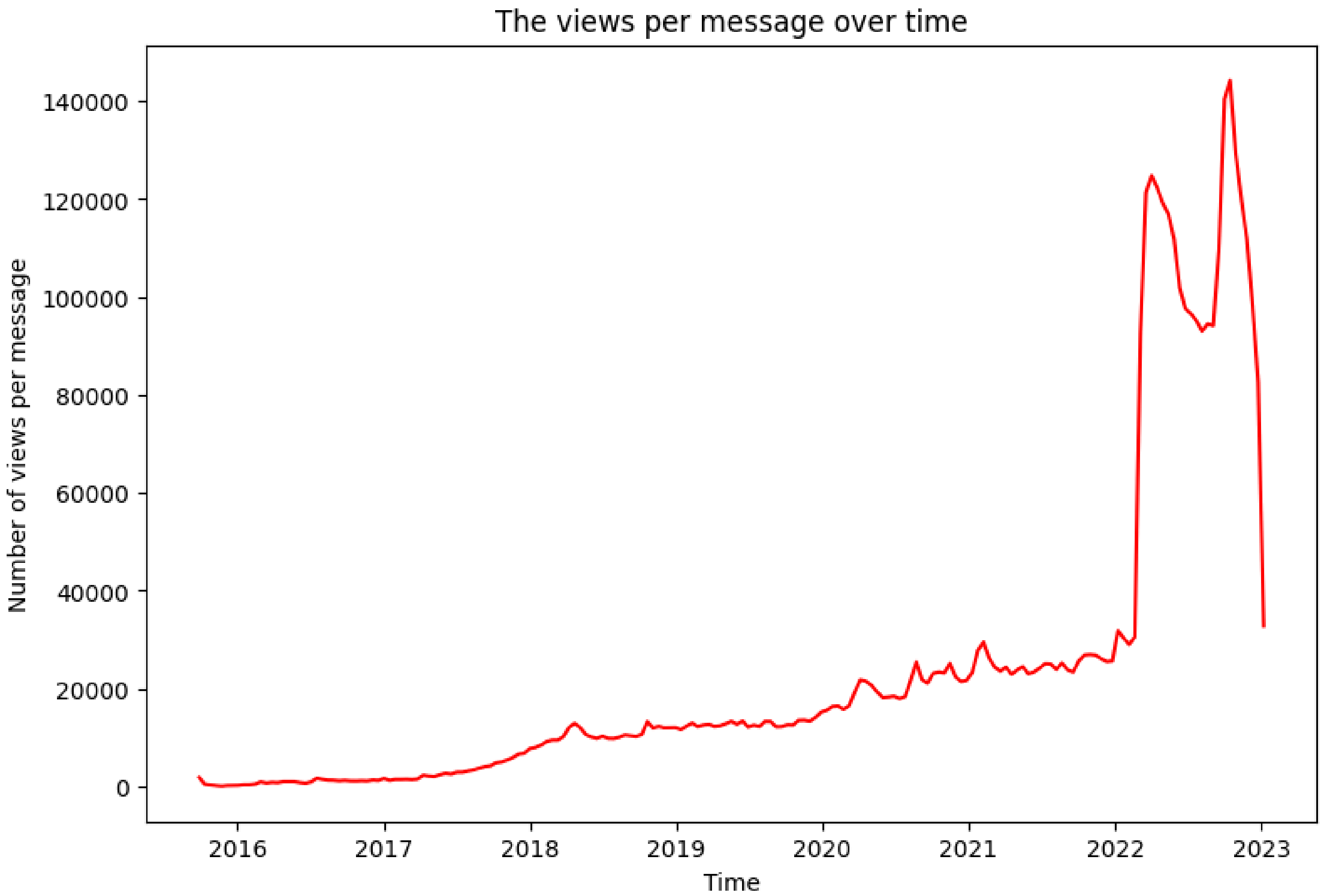
In [105]:
```
views_per_message_over_time = _data[["datetime", "views"]].set_index("datetime")
messages_num_2W = views_per_message_over_time.resample("2W").count().rename(columns={"views": "messages"})
views_num_2W = views_per_message_over_time.resample("2W").sum()
views_num_2W["messages"] = messages_num_2W
views_num_2W["views_per_message"] = views_num_2W["views"] / views_num_2W["messages"]
views_num_2W
```

Out[105]:

datetime	views	messages	views_per_message
2015-09-27 00:00:00+00:00	7.014400e+04	37	1895.783784
2015-10-11 00:00:00+00:00	8.275700e+04	180	459.761111
2015-10-25 00:00:00+00:00	8.572900e+04	245	349.914286
2015-11-08 00:00:00+00:00	9.816600e+04	477	205.798742
2015-11-22 00:00:00+00:00	9.661100e+04	1071	90.206349
...	...	...	...
2022-11-13 00:00:00+00:00	1.363582e+10	113523	120115.026902
2022-11-27 00:00:00+00:00	1.354876e+10	121248	111744.221397
2022-12-11 00:00:00+00:00	1.128060e+10	114573	98457.761314
2022-12-25 00:00:00+00:00	5.858767e+09	70855	82686.713895
2023-01-08 00:00:00+00:00	6.435811e+06	196	32835.770408

191 rows × 3 columns

```
In [106]: fig, ax = plt.subplots(figsize=(9, 6))
          sns.lineplot(views_num_2W, x="datetime", y="views_per_message", color="red")
          plot_annotations("Time", "Number of views per message",
                           "The views per message over time")
```



We can see the tendency to view per-message growth over time and a dramatic jump after the beginning of the war. The plot is almost identical to the plot in Q19. It means that messages and views increase proportionally over time. So, there are overall channels' growth.

Q24: What is channel distribution by most reposted messages?

This distribution indicates the uniqueness of channel content. The more reposted messages, than poorer channel by quality.

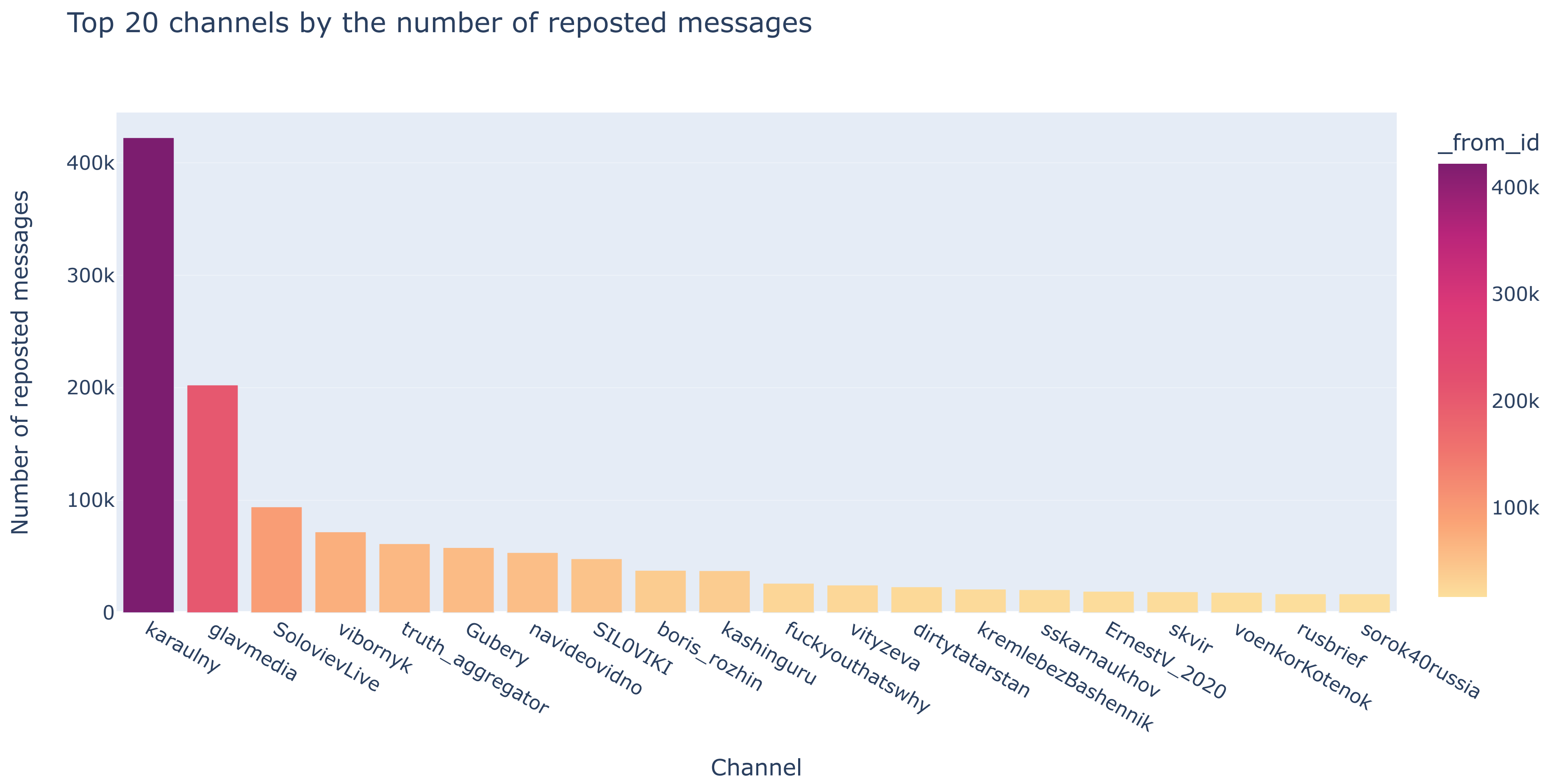
In [107]:

```
channels_by_reposted_messages = _data[["channel", "_from_id"]]
channels_by_reposted_messages = channels_by_reposted_messages.groupby("channel").count()
channels_by_reposted_messages.head()
```

Out[107]:

	_from_id
channel	
AG_DPR	5
AKID_channel	6
ARTolmachev	4
ASGasparyan	5310
ATC_ATC	6

In [108]:
```python
top20_by_reposted_messages = channels_by_reposted_messages.sort_values("_from_id", ascending=False)[:20]
px.bar(top20_by_reposted_messages.reset_index(),
       x="channel",
       y="_from_id",
       color="_from_id",
       color_continuous_scale=px.colors.sequential.Sunsetdark,
       title="Top 20 channels by the number of reposted messages").update_layout(
    xaxis_title="Channel",
    yaxis_title="Number of reposted messages")
```



We can remark the several leaders `karaulny`, `glavmedia`, `boris_rozhin` and `SolovievLive` that are also the leaders in other tops.

Conclusions

There are general tendency to grow the number of messages, views, and toxicity growth by propagandistic channels that have an expected dramatic jump after the Russian-Ukrainian war begins.

The great number of positive toxic reactions in toxic messages reflects that the majority of people react positively to Russian propaganda. Besides, the majority of the channel messages are related to `politics` or `crimes` topics. There are no unusual patterns in message length among data except `readovkanews`.

---

*The overall number of messages from the top 20 channels by messages: **45.79%**
The overall number of views from the top 20 channels by views: **51.47%***

The `karaulny` is the main content accelerator with the greatest number of messages, reposted messages, and toxic messages that has inconsistency in the number of posted messages. The `glavmedia` has a similar pattern in less scale and also presents in the same tops.

The `readovkanews` shows in the same tops as both above but has greater impact and seems to be paid. He is also present at the top by the number of views and has inconsistent the number of messages and average message length over time.

The `rian_ru` and `tass_agency` seem to have more neutral message toxicity despite the sizes of both channels which are in tops by the number of messages and the number of views. The `rian_ru` also has high message significance due to its presence at the top by the number of views per message.

The `swodki` and `rt_russian` show similar behavior as `rian_ru` and `tass_agency` but both have a great number of toxic messages. The `rt_russian` is also one of the most toxic auditory by level of toxicity.

The `SolovievLive` is the most powerful channel by all tops and also one of the main content accelerators. He is one of the greatest by the number of messages, toxic messages, level of toxicity, number of views, and the number of reposted messages.

The `boris_rozhin` and `epoddubny` both have the most toxic auditories by the level of toxicity and message significance. At the same time, `boris_rozhin` has a great number of toxic and reposted messages that also make him one of the main accelerators.

The `izvestia`, `ntvnews`, `rt_russian`, and `tv360` are one of the greatest by the number of messages. They tend to become less popular in comparison with newer big channels.

Other interesting channels are `meduzalive`, `bbbreaking`, `SergiyKolyasnikov`. The first has a great number of toxic messages and a number of views at the same time despite having fewer messages than the previous. The second is an overall big channel with one of the greatest number of messages and views. The third has a smaller size but remains one of the leaders by the number of toxic messages and the level of toxicity.

Limitations

In the research, the main exploration focuses on the biggest channels that accelerate a significant part of all the channels.
More deep insights and exploration of other channels are required.