In [1]:

```python
import pandas as pd

import re
from emoji import UNICODE_EMOJI
from textblob import TextBlob
import altair as alt
import numpy as np
from collections import Counter
import string

import nltk
nltk.download('vader_lexicon')
nltk.download('brown')
nltk.download('punkt')
nltk.download('stopwords')

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/jovyan/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package brown to /home/jovyan/nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

The data cleaning/manipulation functions

In [2]:

```
def extract_tags(text):
    return re.findall("#([a-zA-Z0-9_]{1,50})", text)

def extract_emoji(text):
    return [ch for ch in text if ch in UNICODE_EMOJI['en']]


def clean_tweet(txt):
    temp = re.sub("@[A-Za-z0-9_]+","", txt)
    temp1 = re.sub("#[A-Za-z0-9_]+","", temp)
    temp2 = re.sub(r"http\S+", "", temp1)

    result=''.join(i for i in temp2.lower() if (i.isalpha() or i==' '))
    return result

def word_list(tweet):

    lst = word_tokenize(tweet)
    lst1 = []
    stops = list(stopwords.words('english'))
    for w in lst:
        if w not in stops:
            lst1.append(w)

    return lst1

def sentiment(tweet):
    blob = TextBlob(tweet)

    return blob.sentiment.polarity


def get_date(date):

    return date[:10]

def get_hour(date):

    return date[11:13]
def get_10min(date):

    return date[14]+'0'

def get_min(date):
    return date[14:16]

def firm_pos(score):
    if score >= 0.7:
        return 1
    else: return 0

def pos(score):
    if (score >= 0.25) & (score < 0.7):
        return 1
    else: return 0

def neutral(score):
    if (score >= -0.25) & (score < 0.25):
        return 1
    else: return 0

def neg(score):
    if (score > -0.7) & (score < -0.25):
        return 1
    else: return 0

def firm_neg(score):
    if score <= -0.7:
        return 1
    else: return 0
```

Import data, check duplicate or missing value

In [3]:
```
df= pd.read_csv('Project Data/Lebron 2020 finals.csv')
df['id'].duplicated(keep='last').sum()
```

Out[3]:
```
0
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
id      0
date    0
text    0
dtype: int64
```

Apply data cleaning/manipulation techniques on the data, we now have the used words, tags, emojis, sentiment score, and specific date/hour/min data.

In [5]:

```
df['tags']= df.apply(lambda row: extract_tags(row['text']), axis=1)
df['emojis']= df.apply(lambda row: extract_emoji(row['text']), axis=1)
df['clean_text']= df.apply(lambda row: clean_tweet(row['text']), axis=1)
df['words']= df.apply(lambda row: word_list(row['clean_text']), axis=1)
df['sentiment_score']= df.apply(lambda row: sentiment(row['clean_text']), axis=1)
df['day']= df.apply(lambda row: get_date(row['date']), axis=1)
df['hour']= df.apply(lambda row: get_hour(row['date']), axis=1)
df['10min']= df.apply(lambda row: get_10min(row['date']), axis=1)
df['min']= df.apply(lambda row: get_min(row['date']), axis=1)
df['POS']= df.apply(lambda row: firm_pos(row['sentiment_score']), axis=1)
df['pos']= df.apply(lambda row: pos(row['sentiment_score']), axis=1)
df['neu']= df.apply(lambda row: neutral(row['sentiment_score']), axis=1)
df['neg']= df.apply(lambda row: neg(row['sentiment_score']), axis=1)
df['NEG']= df.apply(lambda row: firm_neg(row['sentiment_score']), axis=1)

df.head()
```

Out[5]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10min	min	POS
0	1311455787101949952	2020-09-30 23:59:59+00:00	#QuestionOfTheDay \nWho will win game 1 of the...	[QuestionOfTheDay, NBAFinals, NBA, LeBronJames...	[]	who will win game of the or	[win, game]	0.200000	2020-09-30	23	50	59	0
1	1311455785973624833	2020-09-30 23:59:59+00:00	@8lackJezus @WokeLotus @egchico3 @WoaXMamba @s...	[MJBEATTRASH]	[]	lol your opinion again show me a team th...	[lol, opinion, show, team, put, points, finals...	0.266667	2020-09-30	23	50	59	0
2	1311455776221941762	2020-09-30 23:59:57+00:00	@netorarefanclub @nigel_dylan @stephenasmith @...	[]	[]	love isnt better than klay of dray and ste...	[love, isnt, better, klay, dray, steph, better...	0.500000	2020-09-30	23	50	59	0
3	1311455758777806849	2020-09-30 23:59:53+00:00	@stephenasmith @KingJames Why still debate thi...	[]	[ ]	why still debate this its all yall talk abou...	[still, debate, yall, talk, basketball, season...	0.000000	2020-09-30	23	50	59	0
4	1311455748673744896	2020-09-30 23:59:50+00:00	@Homeoffree61 How about Stephen Colbert Alec B...	[]	[]	how about stephen colbert alec baldwin amy kl...	[stephen, colbert, alec, baldwin, amy, klobuch...	0.000000	2020-09-30	23	50	59	0

See the overall flow of tweet & sentiment

Group by 'day' and 'hour', we can see the sum of sentiment score and the total tweets count for each hour.

In [6]:

```
score = df.groupby(['day','hour']).agg([np.sum,np.size]).sentiment_score
score = score.reset_index()

## create a column that will be used for visualization, kind of re-create the complete timedelta data
score['date'] = score['day'] + ' ' + score['hour'] + ':00'

## compute the 12 hour rolling average of sentiment score
score[['12hr_senti','12hr_count']] = score.rolling(window=12,min_periods=1).sum()[['sum','size']]
score['12hr_avg'] = score['12hr_senti'] / score['12hr_count']

score.head()
```

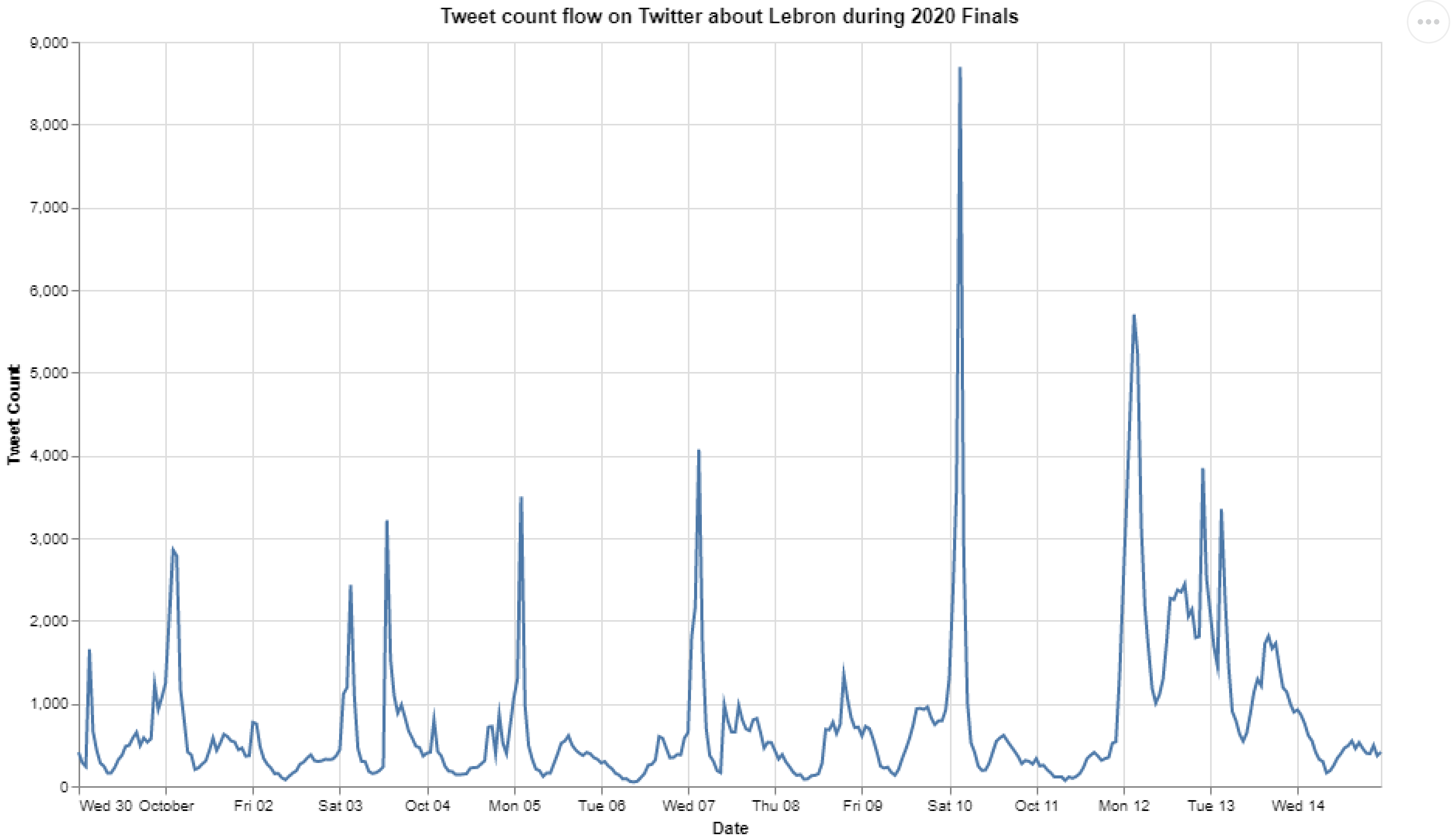
Out[6]:

	day	hour	sum	size	date	12hr_senti	12hr_count	12hr_avg
0	2020-09-30	00	41.799578	410.0	2020-09-30 00:00	41.799578	410.0	0.101950
1	2020-09-30	01	22.001537	287.0	2020-09-30 01:00	63.801115	697.0	0.091537
2	2020-09-30	02	16.443707	238.0	2020-09-30 02:00	80.244821	935.0	0.085823
3	2020-09-30	03	90.483422	1653.0	2020-09-30 03:00	170.728243	2588.0	0.065969
4	2020-09-30	04	45.136855	662.0	2020-09-30 04:00	215.865098	3250.0	0.066420

In [7]:

```
alt.Chart(score).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('size:Q',title='Tweet Count')
).properties(height=480,width=840,title='Tweet count flow on Twitter about Lebron during 2020 Finals')
```

Out[7]:



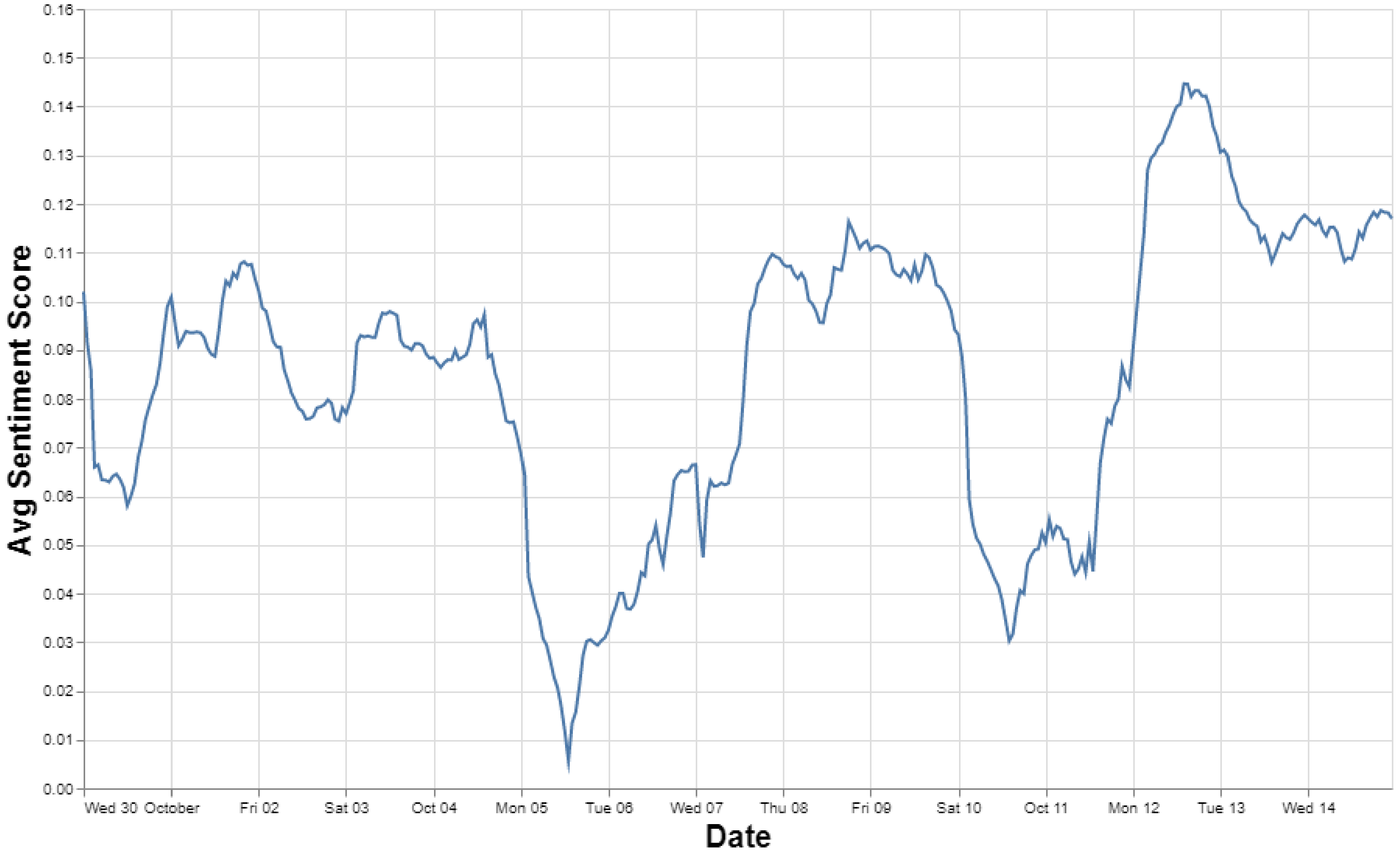
Tweet count flow on Twitter about Lebron during 2020 Finals

Plot the sentiment flow

In [8]:

```
alt.Chart(score).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('12hr_avg:Q',title='Avg Sentiment Score')
).properties(width=840,height=500,title={
      "text": ["Sentiment Flow - Lebron 2020 NBAFinals"],
      "subtitle": ["Sentiment score computed by 12 hour rolling average"]
    }).configure_axis(
    labelFontSize=10,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 25,
    subtitleFontSize = 15
)
```

Out[8]:



Sentiment Flow - Lebron 2020 NBAFinals

Sentiment score computed by 12 hour rolling average

In [9]:

```
flow = df.groupby(['day','hour']).mean()[['POS','pos','neu','neg','NEG']]
flow = flow.reset_index()
flow['date'] = flow['day'] + ' ' + flow['hour'] + ':00'

flow[['POSITIVE','positive','neutral','negative','NEGATIVE']] = flow[['POS','pos','neu','neg','NEG']].rolling(window=6,min_periods

flow1 = pd.DataFrame()

dates = []
values = []
labels = []

for i in ['POSITIVE','positive','neutral','negative','NEGATIVE']:
    lst = []
    lst1 = []
    lst2 = list(flow.date.values)

    for j in range(len(flow)):
        lst.append(i)
        lst1.append(flow[i][j])

    dates += lst2
    labels += lst
    values += lst1


flow1['date'] = pd.Series(dates)
flow1['sentiment_label'] = pd.Series(labels)
flow1['percentage_6hr_avg'] = pd.Series(values)
flow1.head(10)
```

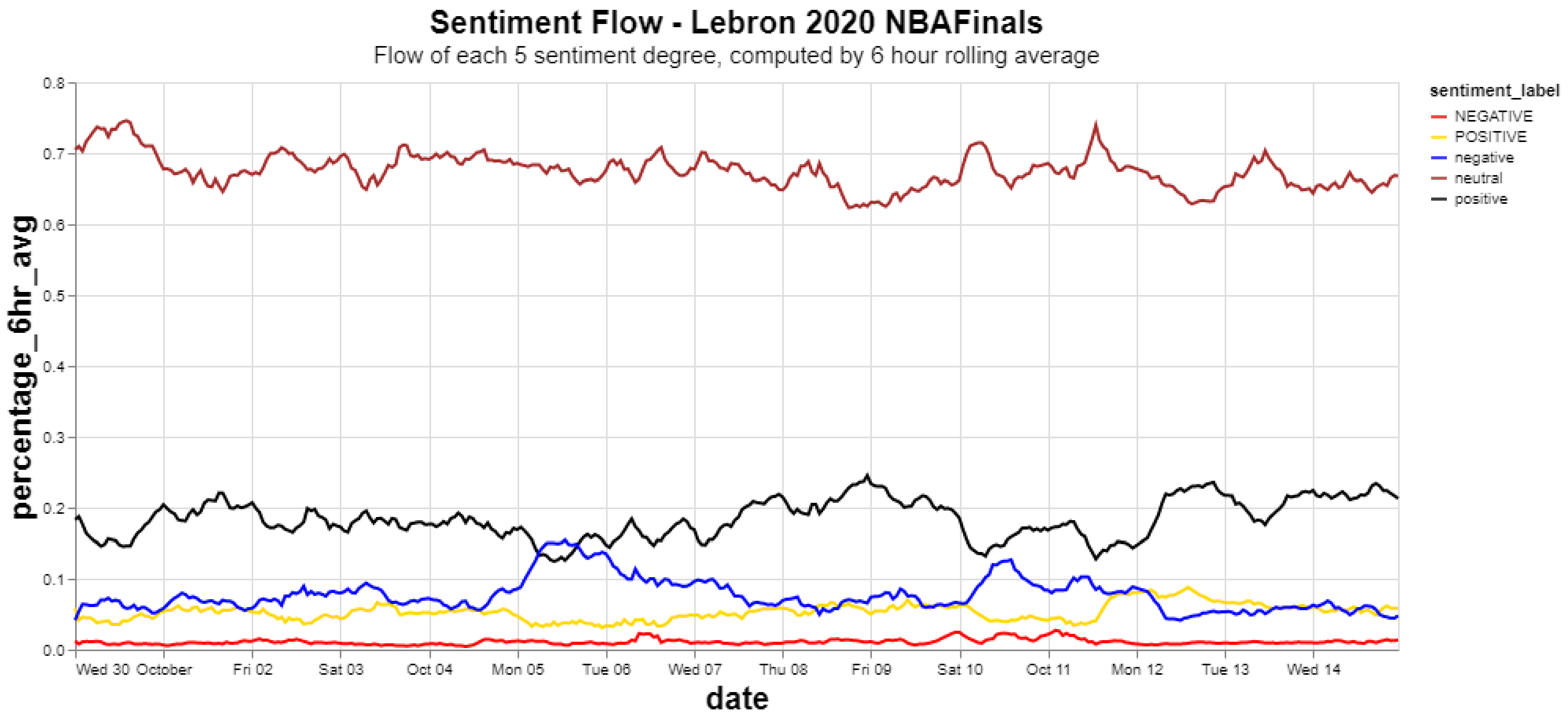
Out[9]:

	date	sentiment_label	percentage_6hr_avg
0	2020-09-30 00:00	POSITIVE	0.058537
1	2020-09-30 01:00	POSITIVE	0.041463
2	2020-09-30 02:00	POSITIVE	0.045850
3	2020-09-30 03:00	POSITIVE	0.044823
4	2020-09-30 04:00	POSITIVE	0.044922
5	2020-09-30 05:00	POSITIVE	0.042032
6	2020-09-30 06:00	POSITIVE	0.037614

	date	sentiment_label	percentage_6hr_avg
7	2020-09-30 07:00	POSITIVE	0.039082
8	2020-09-30 08:00	POSITIVE	0.039472
9	2020-09-30 09:00	POSITIVE	0.040796

In [10]:
```
alt.Chart(flow1).mark_line().encode(
    x='date:T',
    y='percentage_6hr_avg:Q',
    color=alt.Color('sentiment_label',
                   scale=alt.Scale(
            range=['red', 'gold','blue','brown','black']))

).properties(width=840,height=360,title={
      "text": ["Sentiment Flow - Lebron 2020 NBAFinals"],
      "subtitle": ["Flow of each 5 sentiment degree, computed by 6 hour rolling average"]
    }).configure_axis(
    labelFontSize=10,
    titleFontSize=20
).configure_title(
    fontSize = 20,
    subtitleFontSize = 15
)
```

Out[10]:



Sentiment Flow - Lebron 2020 NBAFinals

Flow of each 5 sentiment degree, computed by 6 hour rolling average

In [11]:

```
stats = pd.read_csv("Project Data/Lebron Finals Stats.csv")
stats['time'] = stats['Date'] + ' 00:00'
stats
```

Out[11]:

	Date	Series	Tm	Result	Stats	MP	TRB	AST	STL	TOV	PTS	GmSc	+/-	time
0	2020/10/1	FINALS	GAME1	W (+18)	36min, 25PT, 9AST, 13REB, +10 WIN	36	13	9	1	2	25	24.7	10	2020/10/1 00:00
1	2020/10/3	FINALS	GAME2	W (+10)	39min, 33PT, 9AST, 9REB, +7 WIN	39	9	9	1	0	33	30.6	7	2020/10/3 00:00
2	2020/10/5	FINALS	GAME3	L (-11)	39min ,25PT, 8AST, 10REB, -4 LOSE	39	10	8	0	8	25	17.8	-4	2020/10/5 00:00
3	2020/10/7	FINALS	GAME4	W (+6)	39min, 28PT, 8AST, 12REB, -2 WIN	39	12	8	1	6	28	23.4	-2	2020/10/7 00:00
4	2020/10/10	FINALS	GAME5	L (-3)	42min, 40PT, 7AST, 13REB, +7 WIN	42	13	7	3	4	40	39.1	7	2020/10/10 00:00
5	2020/10/12	FINALS	GAME6	W (+13)	41min, 28PT, 10AST, 14REB, +18 WIN	41	14	10	1	1	28	29.2	18	2020/10/12 00:00

In [12]:

```
# the 12-hour rolling average sentiment flow plot above

senti_line = alt.Chart(score).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('12hr_avg:Q',title='Avg Sentiment Score')
)


# create the dataframe used for text annotations on plot

annotations = [['2020-10-01 00:00:00',0.14, 'Game1 (W)'],
               ['2020-10-01 00:00:00',0.13, '36min, 25PT, 9AST, 13REB'],
               ['2020-10-03 00::00',0.16, 'Game2 (W)'],
               ['2020-10-03 00:00:00',0.15, '39min, 33PT, 9AST, 9REB'],
              ['2020-10-05 00::00',0.12, 'Game3 (L)'],
               ['2020-10-05 00:00:00',0.11, ' 39min ,25PT, 8AST, 10REB'],
               ['2020-10-07 00::00',0.14, 'Game4 (W)'],
               ['2020-10-07 00:00:00',0.13, '39min, 28PT, 8AST, 12REB'],
              ['2020-10-10 00::00',0.15, 'Game5 (L)'],
               ['2020-10-10 00:00:00',0.14, '42min, 40PT, 7AST, 13REB'],
              ['2020-10-12 00::00',0.18, 'Game6 (Champ)'],
              ['2020-10-12 00:00:00',0.17, '41min, 28PT, 10AST, 14REB']]
a_df = pd.DataFrame(annotations, columns=['date','values','note'])
a_df

annotate = [['2020-10-01 00:00:00',0.125,'first'],
               ['2020-10-01 00:00:00',0.105,'first'],
               ['2020-10-03 00:00:00',0.098,'second'],
               ['2020-10-03 00:00:00',0.145,'second'],
               ['2020-10-05 00:00:00',0.08,'third'],
               ['2020-10-05 00:00:00',0.105,'third'],
               ['2020-10-07 00:00:00',0.07,'four'],
               ['2020-10-07 00:00:00',0.125,'four'],
               ['2020-10-10 00:00:00',0.1,'five'],
               ['2020-10-10 00:00:00',0.135,'five'],
               ['2020-10-12 00:00:00',0.13,'six'],
               ['2020-10-12 00:00:00',0.165,'six']]

adf = pd.DataFrame(annotate, columns=['date','value','line'])
```

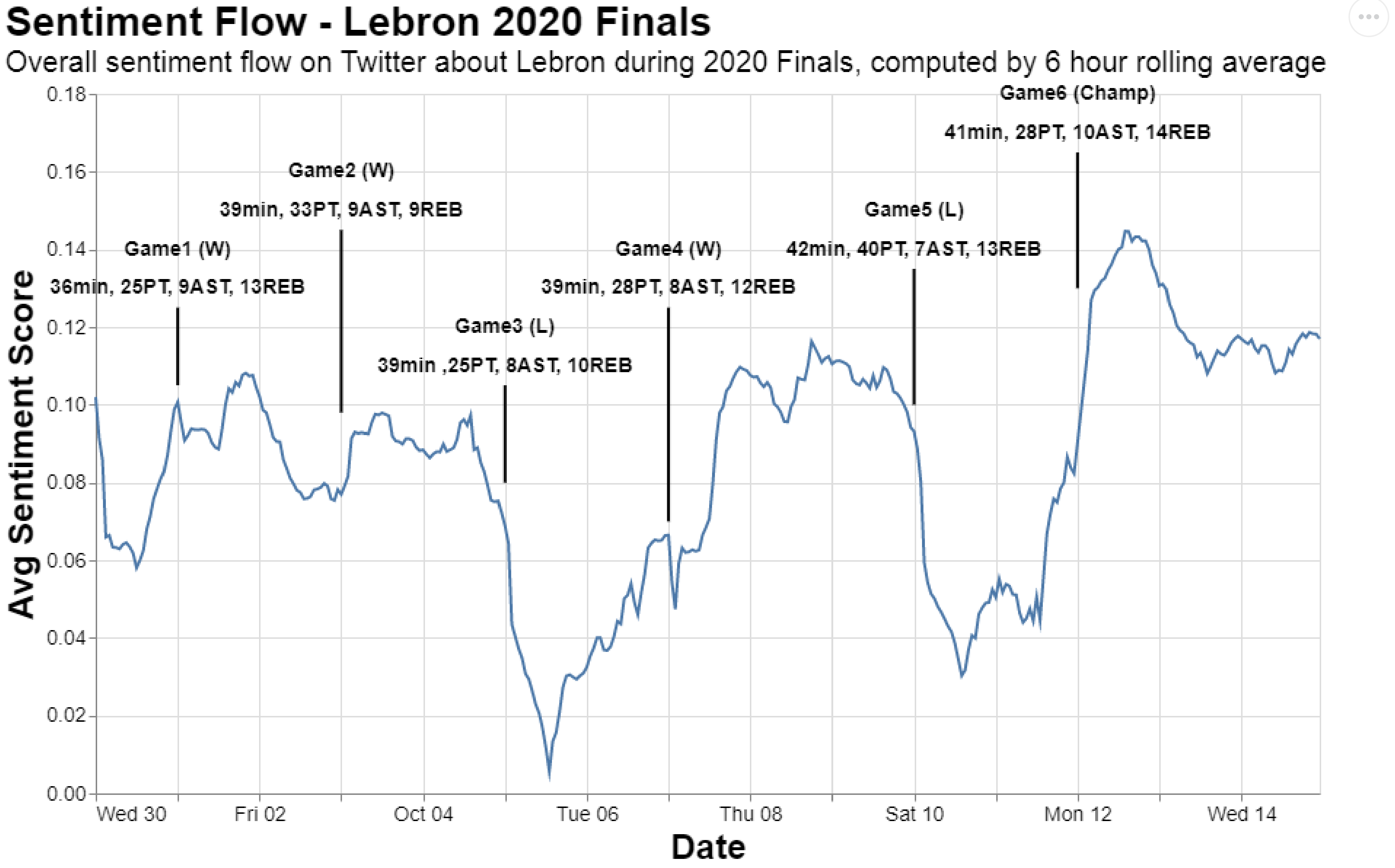
In [13]:

```
text=alt.Chart(a_df).encode(
    x=alt.X('date:T'),
    y=alt.Y('values:Q'),
    text='note').mark_text(size=14,fontWeight='bold').properties(height=390,width=580)

line1=alt.Chart(adf).transform_filter(
        alt.datum.line == 'first'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')

line2=alt.Chart(adf).transform_filter(
        alt.datum.line == 'second'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')

line3=alt.Chart(adf).transform_filter(
        alt.datum.line == 'third'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')

line4=alt.Chart(adf).transform_filter(
        alt.datum.line == 'four'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')

line5=alt.Chart(adf).transform_filter(
        alt.datum.line == 'five'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')

line6=alt.Chart(adf).transform_filter(
        alt.datum.line == 'six'
    ).encode(
    x=alt.X('date:T'),
    y=alt.Y('value:Q')
).mark_line(color='black')
```

In [14]:

```
(senti_line + text + line1 + line2 + line3 + line4 + line5 + line6).properties(
    width=840,height=480,
    title={
      "text": ["Sentiment Flow - Lebron 2020 Finals"],
      "subtitle": ["Overall sentiment flow on Twitter about Lebron during 2020 Finals, computed by 6 hour rolling average"],
      "color": "black",
      "subtitleFontSize":20
    }).configure_axis(
    labelFontSize=14,
    titleFontSize=24
).configure_title(
    anchor='start',
    fontSize = 28,
    subtitleFontSize = 20
)
```

Out[14]:



Sentiment Flow - Lebron 2020 Finals

Overall sentiment flow on Twitter about Lebron during 2020 Finals, computed by 6 hour rolling average

Emoji/Tags Analysis

In [15]:

```
# this return the top 50 most common items in the columns (emoji/tag/word)

def top_item(data,label):

    lst = []
    for i in data[label]:
        lst += i

    C = Counter(lst)
    top50 = C.most_common(50)
    count_df = pd.DataFrame(top50,columns = [label,'count'])

    return count_df
```

In [16]:

```python
c = top_item(df,'tags')
c

# the 50 most popular used tag about Lebron in the 2020 finals
```

Out[16]:

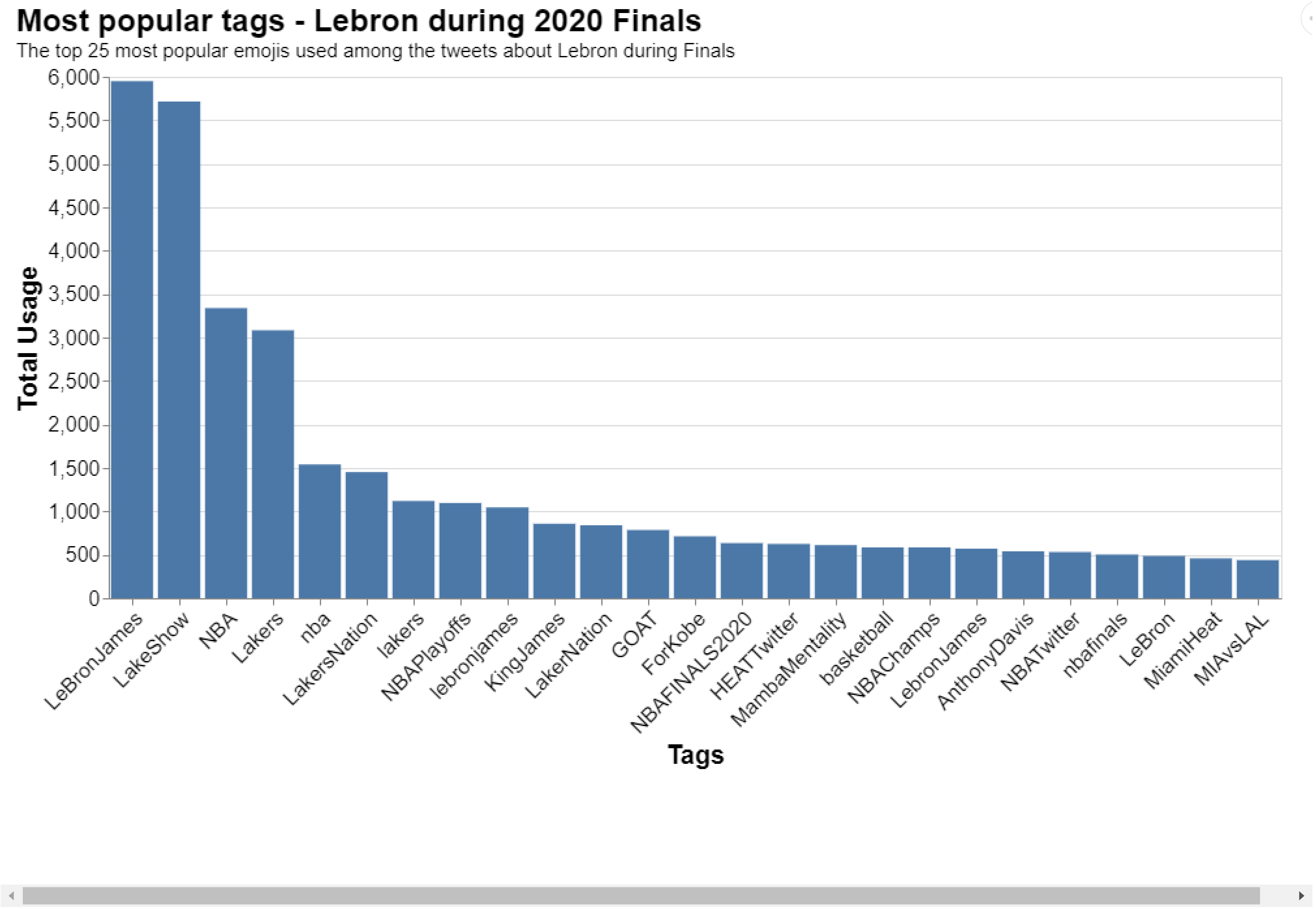
	tags	count
0	NBAFinals	10559
1	LeBronJames	5950
2	LakeShow	5715
3	NBA	3339
4	Lakers	3083
5	nba	1541
6	LakersNation	1453
7	lakers	1120
8	NBAPlayoffs	1097
9	lebronjames	1046
10	KingJames	858
11	LakerNation	840
12	GOAT	786
13	ForKobe	713
14	NBAFINALS2020	635
15	HEATTwitter	625
16	MambaMentality	612
17	basketball	585
18	NBAChamps	585
19	LebronJames	571
20	AnthonyDavis	540
21	NBATwitter	532
22	nbafinals	503
23	LeBron	487
24	MiamiHeat	459

	tags	count
25	MIAvsLAL	439
26	MVP	437
27	EndSARS	432
28	LosAngelesLakers	404
29	4	401
30	KobeBryant	390
31	1	389
32	17	383
33	Heat	373
34	LALvsMIA	365
35	MambaForever	358
36	StriveForGreatness	344
37	lakeshow	329
38	lebron	313
39	GoatJames	289
40	JimmyButler	281
41	lakersvsheat	279
42	kingjames	277
43	Lakeshow	253
44	Kobe	249
45	2	247
46	BlackLivesMatter	247
47	sports	246
48	MichaelJordan	246
49	BLM	242

In [17]:

```
c1=c[1:26]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('tags',sort=['count'],title='Tags',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular tags - Lebron during 2020 Finals"],
      "subtitle":["The top 25 most popular emojis used among the tweets about Lebron during Finals"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[17]:



Most popular tags - Lebron during 2020 Finals

The top 25 most popular emojis used among the tweets about Lebron during Finals

In [18]:
```
c= top_item(df,'emojis')
c
```

Out[18]:

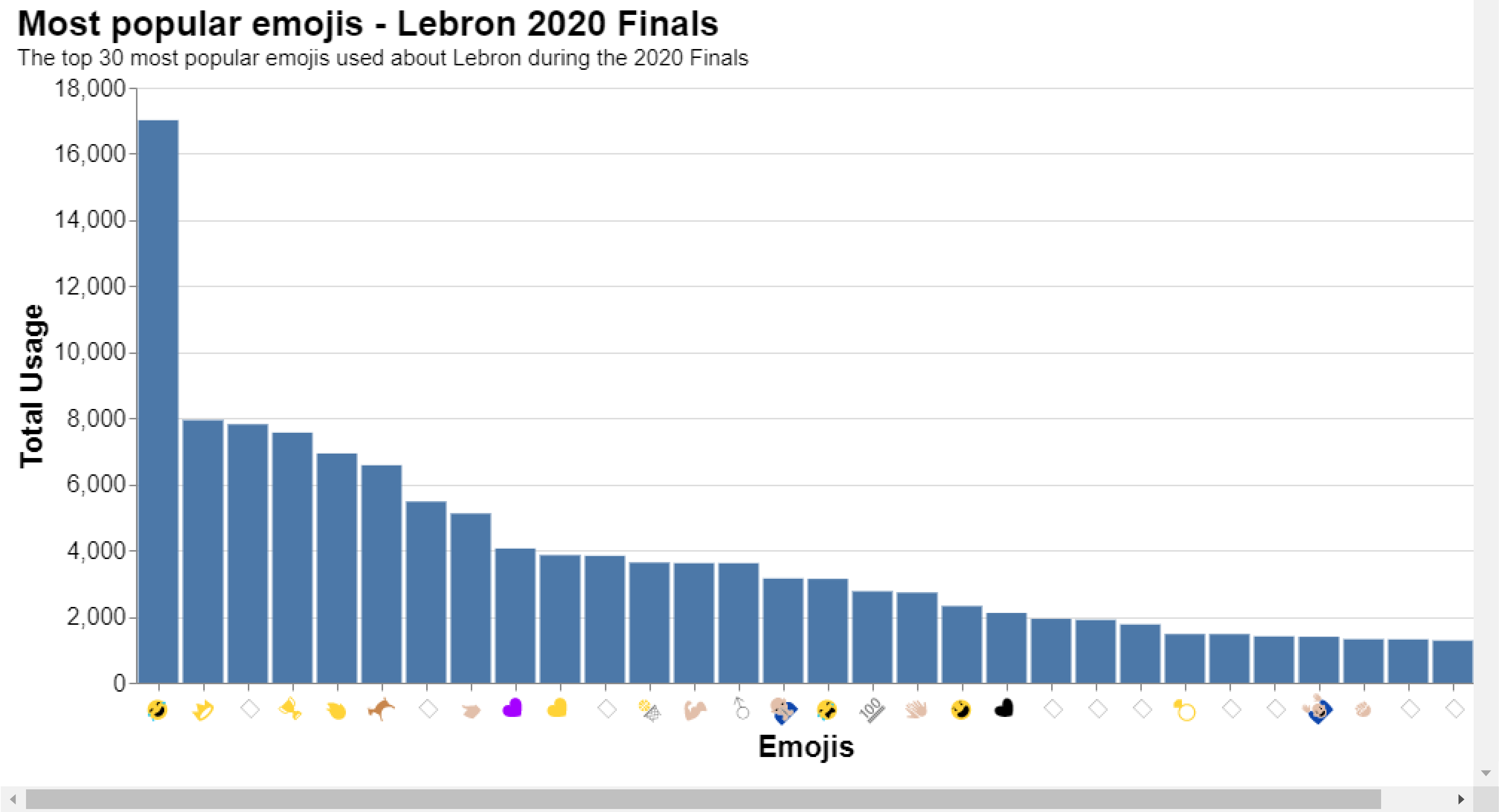
	emojis	count
0	😂	17015
1	👑	7942
2		7816
3	🏆	7564
4	🔥	6933
5	🐐	6576
6		5475
7	👍	5114
8	💜	4054
9	💛	3851
10		3832
11	🏀	3628
12	💪	3610
13	♂	3609
14	🙏	3156
15	😭	3145
16	💯	2768
17	👏	2730
18	😃	2321
19	❤	2114
20		1934
21		1903
22		1766
23	💍	1473
24		1472

	emojis	count
25		1404
26	🙌	1392
27	✊	1318
28		1313
29		1276
30	‼	1256
31	👀	1058
32	😎	1010
33	🎉	947
34		866
35	💀	828
36	😂	802
37		689
38		664
39	😨	622
40	👇	552
41		552
42	😍	546
43		544
44		512
45	💦	491
46	🐍	487
47	😁	472
48	🤪	465
49		463

In [19]:

```
c1=c[:30]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular emojis - Lebron 2020 Finals"],
      "subtitle":["The top 30 most popular emojis used about Lebron during the 2020 Finals"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[19]:



Most popular emojis - Lebron 2020 Finals

The top 30 most popular emojis used about Lebron during the 2020 Finals

In [20]:

```
c=top_item(df,'words')
c
```

Out[20]:

	words	count
0	lebron	84752
1	james	70800
2	game	19960
3	lakers	18526
4	nba	18210
5	finals	16321
6	like	15325
7	one	13249
8	get	13242
9	win	12647
10	dont	12536
11	team	12286
12	time	10536
13	go	10211
14	goat	9629
15	got	9103
16	amp	9064
17	im	9004
18	hes	8609
19	player	8423
20	jordan	8288
21	man	8282
22	king	8206
23	know	7727
24	would	7706
25	people	7451

	words	count
26	thats	7405
27	heat	7082
28	best	7059
29	kobe	7046
30	good	7027
31	championship	6950
32	still	6815
33	better	6710
34	never	6674
35	u	6671
36	basketball	6576
37	th	6318
38	play	6234
39	even	6232
40	mj	5991
41	lol	5910
42	cant	5838
43	great	5832
44	back	5739
45	say	5728
46	see	5663
47	love	5656
48	mvp	5571
49	davis	5541

Target Game6 for detailed analysis

Game6 Lebron won his 4th championship & 4th Finals MVP.

In [21]:
```
df['Date'] = pd.to_datetime(df['date'])
mask = (df['Date'] > '2020-10-11 22:00') & (df['Date'] < '2020-10-12 06:00')
game6 = df.loc[mask].sort_values('Date')
game6 = game6.reset_index()
game6.drop(columns=['index','Date'],inplace=True)

game6.head()
```

Out[21]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10min	min	POS	pos	ne
0	1315411894820364288	2020-10-11 22:00:09+00:00	@JerryLawler @Browns The true King @KingJames!...	[]	[]	the true king you better recognize	[true, king, better, recognize]	0.425	2020-10-11	22	00	00	0	1	
1	1315411899505356802	2020-10-11 22:00:10+00:00	@nicekicks @Lakers @KingJames Come on James I ...	[]	[]	come on james i put money on your ass	[come, james, put, money, ass]	0.000	2020-10-11	22	00	00	0	0	
2	1315412155248906240	2020-10-11 22:01:11+00:00	Finish the job @KingJames https://t.co/eFm50C...	[]	[ ]	finish the job	[finish, job]	0.000	2020-10-11	22	00	01	0	0	
3	1315412155014021122	2020-10-11 22:01:11+00:00	@Joker__Slays @Lakers @KingJames Are you on la...	[]	[]	are you on lakers twitter page on	[lakers, twitter, page]	0.000	2020-10-11	22	00	01	0	0	
4	1315412198248919040	2020-10-11 22:01:21+00:00	@WilliamHill Miami heat to win -1.5. Both team...	[Yourodds]	[]	miami heat to win both teams to score over ...	[miami, heat, win, teams, score, points, antho...	0.800	2020-10-11	22	00	01	1	0	

In [22]:
```
c=top_item(game6,'tags')
c
```

Out[22]:

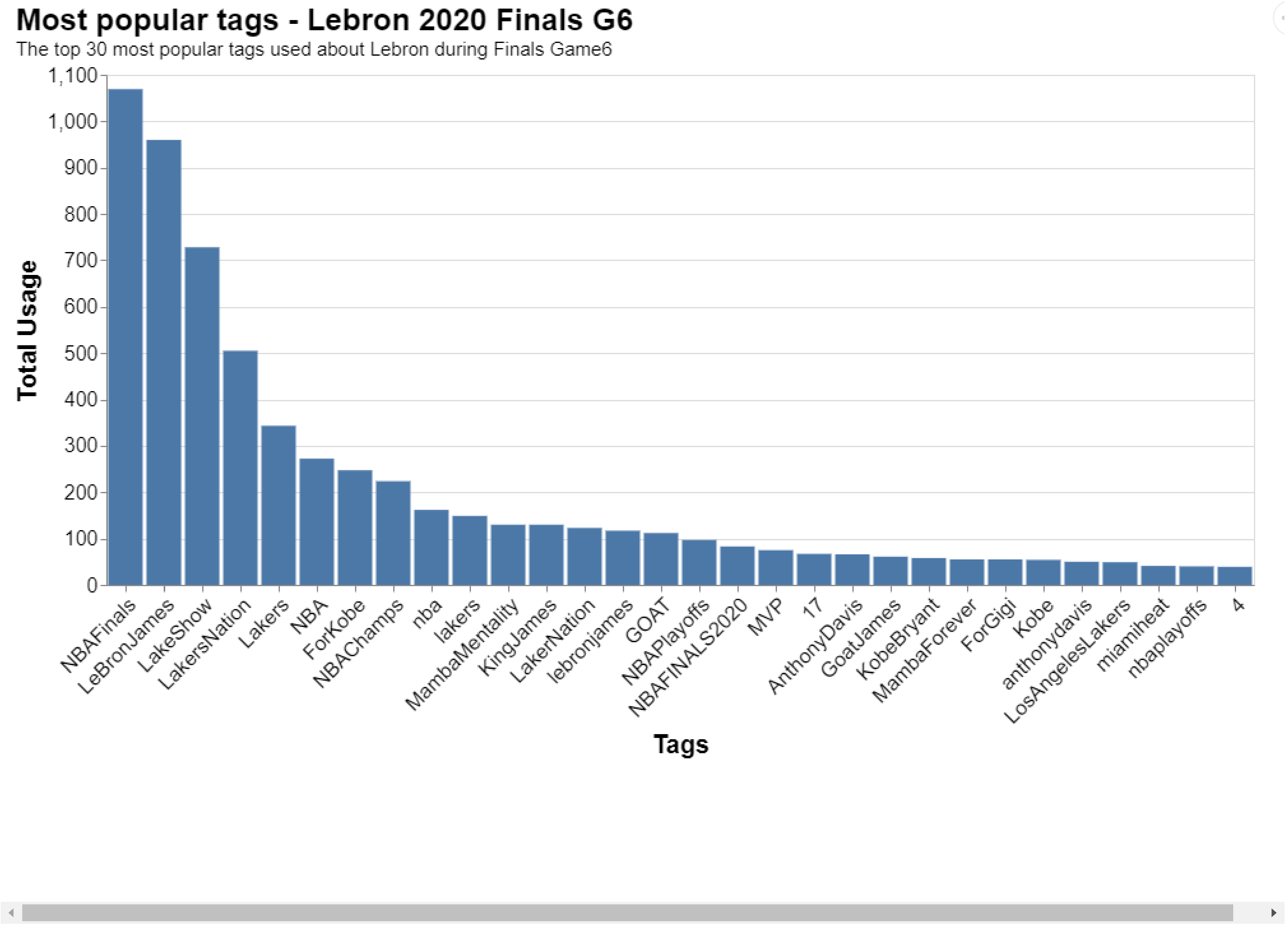
	tags	count
0	NBAFinals	1069
1	LeBronJames	960
2	LakeShow	728
3	LakersNation	505
4	Lakers	343
5	NBA	272
6	ForKobe	247
7	NBAChamps	224
8	nba	162
9	lakers	149
10	MambaMentality	130
11	KingJames	130
12	LakerNation	123
13	lebronjames	117
14	GOAT	112
15	NBAPlayoffs	97
16	NBAFINALS2020	83
17	MVP	75
18	17	67
19	AnthonyDavis	66
20	GoatJames	61
21	KobeBryant	58
22	MambaForever	55
23	ForGigi	55
24	Kobe	54
25	anthonydavis	50

	tags	count
26	LosAngelesLakers	49
27	miamiheat	41
28	nbaplayoffs	40
29	4	39
30	LakersVsHeat	39
31	Champions	39
32	basketball	35
33	HEATTwitter	35
34	rajonrondo	35
35	LALakers	35
36	PROPSBET	34
37	HeatNation	34
38	kylekuzma	32
39	StriveForGreatness	32
40	lakeshow	32
41	jimmybutler	31
42	LebronJames	30
43	Kobethisisforyou	30
44	1	28
45	bamadebayo	28
46	FinalsMVP	28
47	DuncanRobinson	26
48	lakersnation	26
49	kobe	26

In [23]:

```
c1=c[:30]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('tags',sort=['count'],title='Tags',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular tags - Lebron 2020 Finals G6"],
      "subtitle":["The top 30 most popular tags used about Lebron during Finals Game6"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[23]:



Most popular tags - Lebron 2020 Finals G6

The top 30 most popular tags used about Lebron during Finals Game6

In [24]:

```
c= top_item(game6,'emojis')
c
```

Out[24]:

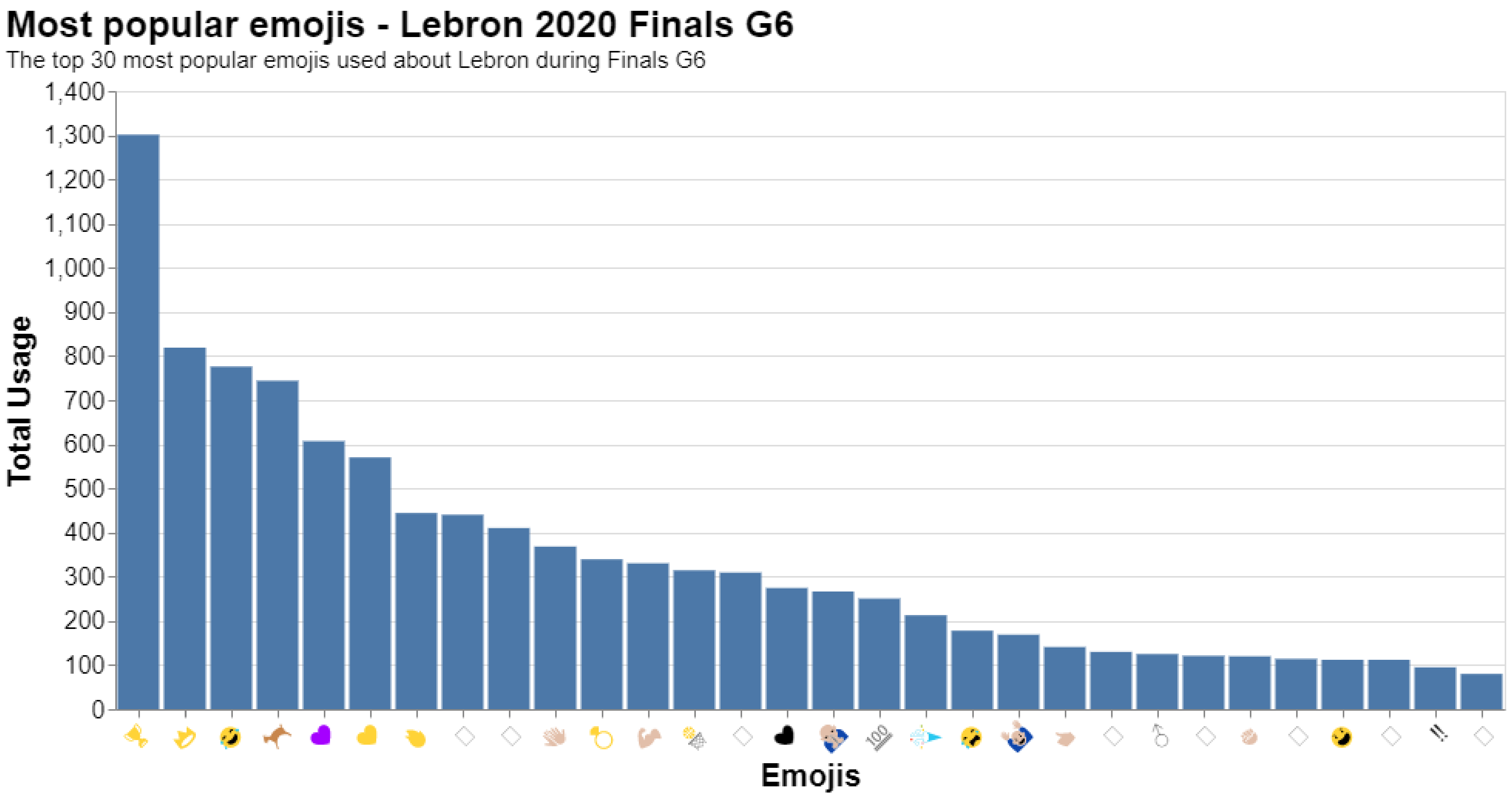
	emojis	count
0	🏆	1301
1	👑	819
2	😂	776
3	🐐	744
4	💜	607
5	💛	570
6	🔥	444
7		440
8		410
9	👏	368
10	💍	339
11	💪	330
12	🏀	314
13		309
14	❤	274
15	🙏	266
16	💯	250
17	🎉	213
18	😭	178
19	🙌	169
20	👍	141
21		130
22	♂	125
23		121
24	✊	120

	emojis	count
25		114
26	😃	112
27		112
28	‼	95
29		80
30	🎊	69
31		64
32		57
33		55
34	🐍	54
35		54
36		52
37	😎	51
38	💔	51
39		50
40	😍	48
41		46
42	👀	43
43		38
44	👇	35
45		34
46		33
47	😂	32
48		32
49	💎	31

In [25]:

```
c1=c[:30]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular emojis - Lebron 2020 Finals G6"],
      "subtitle":["The top 30 most popular emojis used about Lebron during Finals G6"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[25]:


Most popular emojis - Lebron 2020 Finals G6
The top 30 most popular emojis used about Lebron during Finals G6
Total Usage
1,400
1,300
1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
Emojis

```
In [ ]:
```