Information Visualization Altair Lab 5

1. Basic Chart

```
import altair as alt
import pandas as pd
from vega_datasets import data as vega_data# load the data
datasetURL='https://raw.githubusercontent.com/Siyinz/SI-649/master/president_polls.csv'
df=pd.read_csv(datasetURL, encoding="latin-1")
df['end_date']= pd.to_datetime(df.end_date)
```

```
df.head()
```

	question_id	poll_id	cycle	state	pollster_id	pollster	sponsor_ids	sponsors	display_name	pol
0	116815	63513	2020	New Hampshire	1528	AtlasIntel	NaN	NaN	AtlasIntel	
1	116815	63513	2020	New Hampshire	1528	AtlasIntel	NaN	NaN	AtlasIntel	
2	116816	63513	2020	New Hampshire	1528	AtlasIntel	NaN	NaN	AtlasIntel	
3	116816	63513	2020	New Hampshire	1528	AtlasIntel	NaN	NaN	AtlasIntel	
4	116817	63513	2020	New Hampshire	1528	AtlasIntel	NaN	NaN	AtlasIntel	

```
# to resample in the way we are below,
# we need to use the end_date as the index (so instead of 0,1,2...
# as the index, we'll have the date of the poll. This does not need
# to be unique).
df1 = df.copy
df1 = df.set_index('end_date')# a new dataframe to hold our weekly average
df_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
# groupby will group by candidate, pct will grab the percent
# column, resample('W') will group by week, and mean() will calculate
# the mean. This form only works because we made the index the
# end_date (which is a time).
df_weekly['pct'] = df1.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets rid
# we have a more standard dataframe with plain columns (0,1,...n)
df_weekly = df_weekly.dropna().reset_index()
df_weekly.head()
```

	answer	end_date	pct
0	Amash	2019-06-02	9.7
1	Amash	2019-07-28	5.5
2	Amash	2019-09-15	4.5
3	Bennet	2019-06-30	26.0
4	Bennet	2019-07-28	28.0

```
df2 = df.copy
df2 = df.set_index('end_date')# a new dataframe to hold our weekly average
df_monthly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
```

```
df_monthly['pct'] = df2.groupby('answer').pct.resample('M').mean()# clean up weeks with no data and gets ric
df_monthly = df_monthly.dropna().reset_index()
df_monthly.head()
```

	answer	end_date	pct
0	Amash	2019-05-31	9.7
1	Amash	2019-07-31	5.5
2	Amash	2019-09-30	4.5
3	Bennet	2019-06-30	26.0
4	Bennet	2019-07-31	28.0

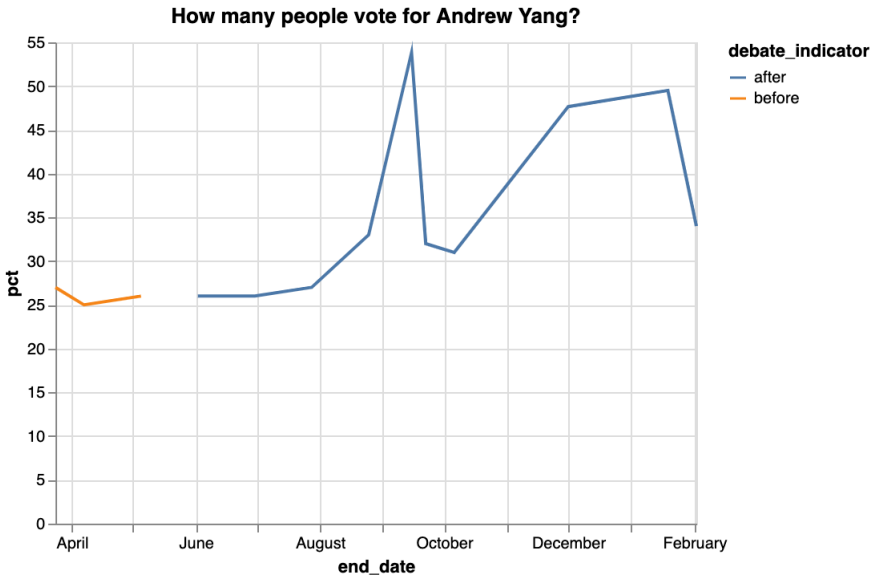
```
df2 = df.copy
df2 = df.set_index('end_date')# a new dataframe to hold our weekly average
df_daily = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
df_daily['pct'] = df2.groupby('answer').pct.resample('D').mean()# clean up weeks with no data and gets rid c
df_daily = df_daily.dropna().reset_index()
df_daily.head()
```

	answer	end_date	pct
0	Amash	2019-05-30	9.7
1	Amash	2019-07-28	5.5
2	Amash	2019-09-11	4.5
3	Bennet	2019-06-24	26.0
4	Bennet	2019-07-27	28.0

Insight 1

After the first Democratic debate, Andrew Yang slowly but steadily got the votes. The thrid dempcratic debate witnessed the hig Yang, even he only got 7.8 minutes' speaking. But the approval rate did not last long.

```
# make a line chart
insight1 = alt.Chart(df_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Yang")
).transform_calculate(
    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('debate_indicator:N')  # color by indicator
).properties(title='How many people vote for Andrew Yang?')
insight1
```

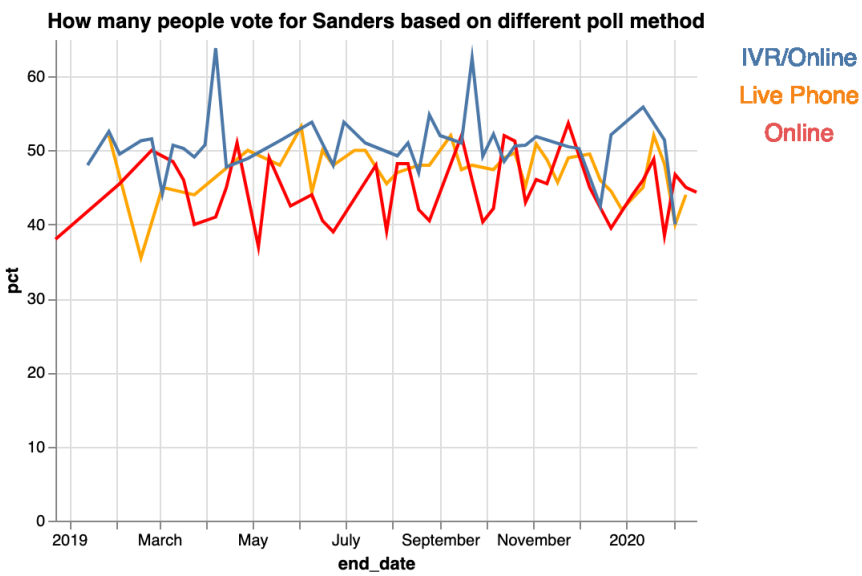


Insight 2

Bernie Sanders got higher approve rate on the polls done by the methodology of IVR/Online.

```
live_phone = df[df['methodology']=='Live Phone']
online = df[df['methodology']=='Online']
ivr = df[(df['methodology']=='IVR/Online')|(df['methodology']=='Online/IVR')]

live_phone = live_phone.set_index('end_date')# a new dataframe to hold our weekly average
phone_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
phone_weekly['pct'] = live_phone.groupby('answer').pct.resample('W').mean()# clean up weeks with no data anc
phone_weekly = phone_weekly.dropna().reset_index()

online = online.set_index('end_date')# a new dataframe to hold our weekly average
online_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
online_weekly['pct'] = online.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and ge
online_weekly = online_weekly.dropna().reset_index()

ivr = ivr.set_index('end_date')# a new dataframe to hold our weekly average
ivr_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
ivr_weekly['pct'] = ivr.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets ric
ivr_weekly = ivr_weekly.dropna().reset_index()

Sanders_phone = alt.Chart(phone_weekly).mark_line(color = 'orange').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Sanders")
).transform_calculate(
    # calculate an indicator column called debate_indicator
    # if the end_date is before 5/12/2019 set this indicator to before
    # otherwise set it to after
    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)
Sanders_online = alt.Chart(online_weekly).mark_line(color = 'red').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Sanders")
).transform_calculate(
    # calculate an indicator column called debate_indicator
    # if the end_date is before 5/12/2019 set this indicator to before
```

```
    # if the end_date is before 5/12/2019 set this indicator to before
    # otherwise set it to after
    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)

Sanders_IVR = alt.Chart(ivr_weekly).mark_line().transform_filter(
    # only keep around biden
    (alt.datum.answer == "Sanders")
).transform_calculate(
    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
).properties(title='How many people vote for Sanders based on different poll method')

methodology = alt.Chart(df,height=70).mark_text(size = 15
).encode(
    alt.Y('methodology:N', sort='ascending',title=None, axis=None),
    alt.Text('methodology:N'),
    alt.Color('methodology:N',legend=None)
).transform_filter(alt.FieldOneOfPredicate(field='methodology', oneOf=["Live Phone","Online", "IVR/Online"]
)

(Sanders_phone + Sanders_online + Sanders_IVR |methodology).configure_view(strokeOpacity=0)
```



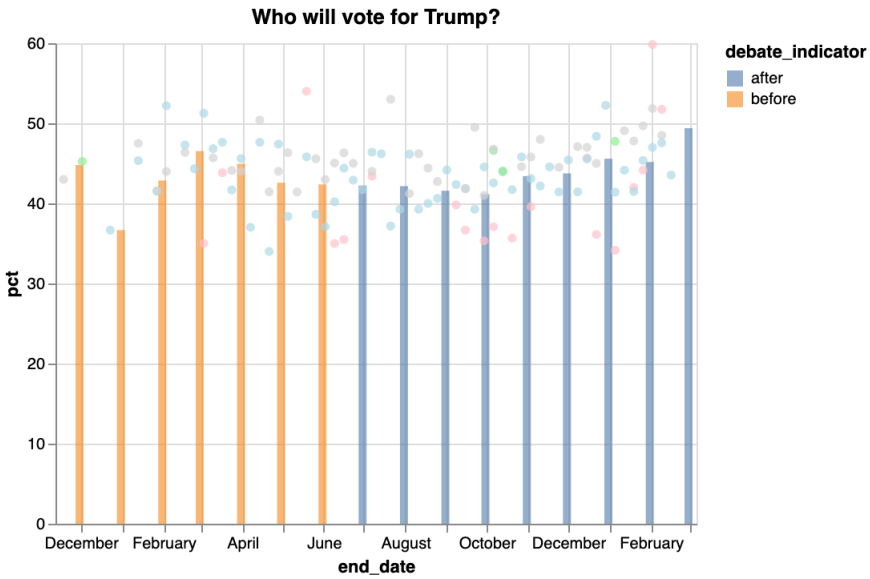
Insight 3

Donald Trump's approval rate was steadily going up after the Democratic Debate, and got the highest approval percentage in Fel after Iowa caucus. The increasing approval rate in Feburary mainly results from the increasing population voting for Trump in th

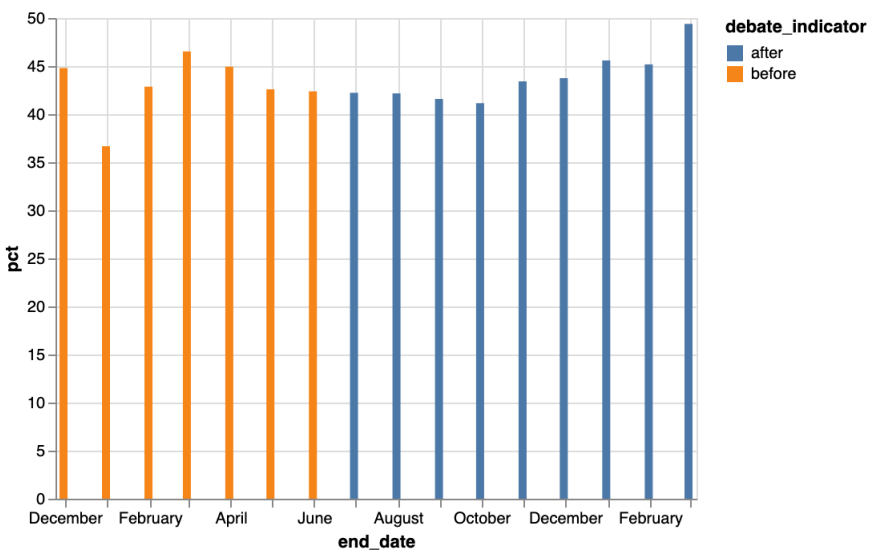
```
adult = df[df['population']=='a']
registerd_voters = df[df['population']=='rv']
voters = df[df['population']=='v']
likely_voters = df[df['population']=='lv']

adult = adult.set_index('end_date')# a new dataframe to hold our weekly average
adult_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
adult_weekly['pct'] = adult.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets
adult_weekly = adult_weekly.dropna().reset_index()
```

```
registerd_voters = registerd_voters.set_index('end_date')# a new dataframe to hold our weekly average
registerd_voters_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
registerd_voters_weekly['pct'] = registerd_voters.groupby('answer').pct.resample('W').mean()# clean up weeks
registerd_voters_weekly = registerd_voters_weekly.dropna().reset_index()

voters = voters.set_index('end_date')# a new dataframe to hold our weekly average
voters_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
voters_weekly['pct'] = voters.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and ge
voters_weekly = voters_weekly.dropna().reset_index()

likely_voters = likely_voters.set_index('end_date')# a new dataframe to hold our weekly average
likely_voters_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
likely_voters_weekly['pct'] = likely_voters.groupby('answer').pct.resample('W').mean()# clean up weeks with
likely_voters_weekly = likely_voters_weekly.dropna().reset_index()

Trump = alt.Chart(df_monthly).mark_bar(opacity=0.6).transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).transform_calculate(
    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('debate_indicator:N')  # color by indicator
).properties(title='Who will vote for Trump?')


Trump_a = alt.Chart(adult_weekly).mark_circle(color = 'pink').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)

Trump_rv = alt.Chart(registerd_voters_weekly).mark_circle(color = 'lightblue').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)

Trump_v = alt.Chart(voters_weekly).mark_circle(color = 'lightgreen').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)

Trump_lv = alt.Chart(likely_voters_weekly).mark_circle(color = 'lightgrey').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
)

population = alt.Chart(df,height=70).mark_text(size = 15
).encode(
    alt.Y('population:N', sort='ascending',title=None, axis=None),
    alt.Text('population:N'),
    alt.Color('population:N',legend=None)
```

```
    alt.color('population:N',legend=None)
)
Trump + Trump_a + Trump_rv + Trump_v + Trump_lv
```




```
Trump_bar = alt.Chart(df_monthly).mark_bar().transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).transform_calculate(

    debate_indicator = "if(datum.end_date < datetime(2019, 5, 1), 'before', 'after')"
).encode(
    x=alt.X('end_date:T'),              # time on the X axis
    y=alt.Y('pct:Q'),                   # pct polling on the Y
    color=alt.Y('debate_indicator:N')   # color by indicator
)
Trump_bar
```



Insight 4

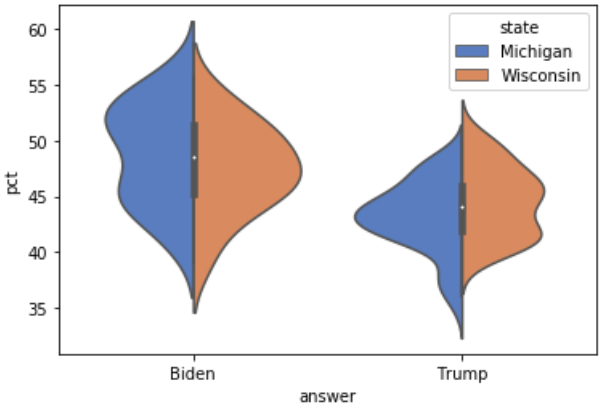
In the Swing state of Michigan and Wisconsin, for most of the time, Biden's approval rate is higher than Trump's.

```
MI_polls = df[df['state']=='Michigan']
MI_polls = MI_polls.set_index('end_date')# a new dataframe to hold our weekly average
```

```
MI_polls = MI_polls.set_index('end_date')# a new dataframe to hold our weekly average
MI_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
MI_weekly['pct'] = MI_polls.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets
MI_weekly = MI_weekly.dropna().reset_index()

WI_polls = df[df['state']=='Wisconsin']
WI_polls = WI_polls.set_index('end_date')# a new dataframe to hold our weekly average
WI_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
WI_weekly['pct'] = WI_polls.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets
WI_weekly = WI_weekly.dropna().reset_index()

Biden_MI = alt.Chart(MI_weekly).mark_line().transform_filter(
    # only keep around biden
    (alt.datum.answer == "Biden")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    #color=alt.Y('debate_indicator:N')  # color by indicator
)
Biden_WI = alt.Chart(WI_weekly).mark_line(color = 'red').transform_filter(
    # only keep around biden
    (alt.datum.answer == "Biden")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    #color=alt.Y('debate_indicator:N')  # color by indicator
)
Trump_WI = alt.Chart(WI_weekly).mark_line(color = 'red',strokeDash = [1,1]).transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    #color=alt.Y('debate_indicator:N')  # color by indicator
)
Trump_MI = alt.Chart(MI_weekly).mark_line(strokeDash = [1,1]).transform_filter(
    # only keep around biden
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    #color=alt.Y('debate_indicator:N')  # color by indicator
)
(Biden_MI + Trump_MI).properties(title='Biden(line) VS. Trump(dash) in Michigan') | (Biden_WI + Trump_WI).pr
```

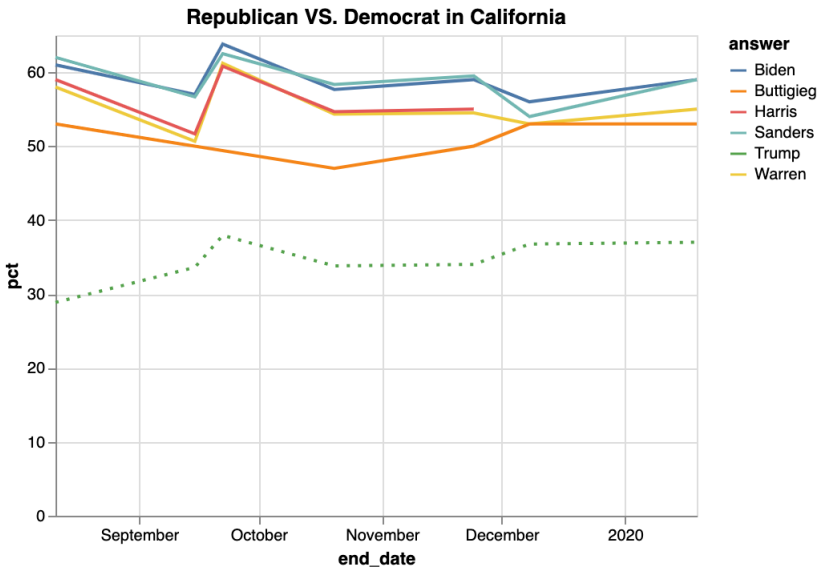
```
import seaborn as sns
df_mi_wi = df[(df['state']=='Michigan')|(df['state']=='Wisconsin')]
df_mi_wi = df_mi_wi[(df_mi_wi['answer']=='Biden')|(df_mi_wi['answer']=='Trump')]
sns.violinplot(x = 'answer', y = 'pct', data = df_mi_wi, hue = 'state',palette="muted", split=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ffa1b3fd0>
```




```
CA_polls = df[df['state']=='California']
CA_polls = CA_polls.set_index('end_date')# a new dataframe to hold our weekly average
CA_weekly = pd.DataFrame()# group by the candidate name, and then calculate weekly averages
CA_weekly['pct'] = CA_polls.groupby('answer').pct.resample('W').mean()# clean up weeks with no data and gets
CA_weekly = CA_weekly.dropna().reset_index()

Warren_CA = alt.Chart(CA_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Warren")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by name
)
Biden_CA = alt.Chart(CA_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Biden")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by indicator
)
Sanders_CA = alt.Chart(CA_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Sanders")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by indicator
)
Buttigieg_CA = alt.Chart(CA_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Buttigieg")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by indicator
)
Harris_CA = alt.Chart(CA_weekly).mark_line().transform_filter(
    (alt.datum.answer == "Harris")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by indicator
)
Trump_CA = alt.Chart(CA_weekly).mark_line(strokeDash = [2,5]).transform_filter(
```

```
    (alt.datum.answer == "Trump")
).encode(
    x=alt.X('end_date:T'),             # time on the X axis
    y=alt.Y('pct:Q'),                  # pct polling on the Y
    color=alt.Y('answer:N')  # color by indicator
)
(Warren_CA + Trump_CA +Biden_CA+Sanders_CA+Buttigieg_CA + Harris_CA).properties(title='Republican VS. Democ
```



Insight 5

For the state of California, all the democratic candidates got higher approval percentages than Trump.

```
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
!pip install selenium
```

```
insight1.to_json()
```

```
insight1.save('insight1.json')
```

Double-click (or enter) to edit