Analysis of early 2020 Democratic campaign co-donors

This notebook analyzes contribution data from Democratic presidential campaigns' FEC filings for the first quarter of 2019.

Load candidate, committee, and filing data

In [1]:
```
import pandas as pd
import fecfile
```

Candidates

In [2]:
```
candidates = pd.read_csv("../inputs/candidates.csv")

candidates
```

Out[2]:

	Candidate Name	Candidate Sex	Committee Name	committee_id
0	Amy Klobuchar	F	Amy for America	C00696419
1	Andrew Yang	M	Friends of Andrew Yang	C00659938
2	Bernie Sanders	M	Bernie 2020	C00696948
3	Beto O'Rourke	M	Beto for America	C00699090
4	Cory Booker	M	Cory 2020	C00695510
5	Elizabeth Warren	F	Warren for President	C00693234
6	Jay Inslee	M	Inslee for America	C00698050
7	John Delaney	M	Friends of John Delaney	C00508416
8	John Hickenlooper	M	Hickenlooper 2020	C00698258
9	Julián Castro	M	Julián for the Future	C00693044
10	Kamala Harris	F	Kamala Harris for the People	C00694455
11	Kirsten Gillibrand	F	Gillibrand 2020	C00694018
12	Pete Buttigieg	M	Pete for America	C00697441
13	Tulsi Gabbard	F	Tulsi Now	C00693713
14	Wayne Messam	M	Wayne Messam for America	C00699280

Filing metadata

Here, we load basic metadata about each filing, and also calculate what proportion of money raised from individual contributions has been itemized in each candidate/committee's filings:

In [3]:
```
filings = pd.read_csv(
    "../inputs/filings.csv",
    low_memory = False
)

filings
```

Out[3]:

	Candidate Name	committee_id	filing_id	report_title	date_coverage_from	date_coverage_to	ar
0	Amy Klobuchar	C00696419	1326529	APR QUARTERLY	2019-02-07	2019-03-31	
1	Andrew Yang	C00659938	1326379	APR QUARTERLY	2019-01-01	2019-03-31	
2	Bernie Sanders	C00696948	1326070	APR QUARTERLY	2019-01-01	2019-03-31	
3	Beto O'Rourke	C00699090	1326481	APR QUARTERLY	2019-03-14	2019-03-31	
4	Cory Booker	C00695510	1326465	APR QUARTERLY	2019-01-01	2019-03-31	
5	Elizabeth Warren	C00693234	1326299	APR QUARTERLY	2019-01-01	2019-03-31	
6	Jay Inslee	C00698050	1326136	APR QUARTERLY	2019-02-14	2019-03-31	
7	John Delaney	C00508416	1324749	APR QUARTERLY	2019-01-01	2019-03-31	
8	John Hickenlooper	C00698258	1326014	APR QUARTERLY	2019-01-01	2019-03-31	
9	Julián Castro	C00693044	1326324	APR QUARTERLY	2019-01-01	2019-03-31	
10	Kamala Harris	C00694455	1326016	APR QUARTERLY	2019-01-21	2019-03-31	
11	Kirsten Gillibrand	C00694018	1326061	APR QUARTERLY	2019-01-01	2019-03-31	
12	Pete Buttigieg	C00697441	1324922	APR QUARTERLY	2019-01-01	2019-03-31	
13	Tulsi Gabbard	C00693713	1326558	APR QUARTERLY	2019-01-11	2019-03-31	
14	Wayne Messam	C00699280	1326345	APR QUARTERLY	2019-01-01	2019-03-31	

In [4]:
```
def get_additional_metadata(filing_id):
    filing = fecfile.from_file(f"../inputs/filings/{int(filing_id)}.fec")
    data = dict((c, filing["filing"][c]) for c in [
        "col_a_individuals_itemized",
        "col_a_individuals_unitemized",
        "col_a_individual_contribution_total",
    ])
    data["filing_id"] = int(filing_id)
    return data
```

In [5]:
```
additional_filing_metadata = pd.DataFrame([
    get_additional_metadata(filing_id)
    for filing_id in filings["filing_id"]
])

additional_filing_metadata.head()
```

Out[5]:

	col_a_individual_contribution_total	col_a_individuals_itemized	col_a_individuals_unitemized	fili
0	5232375.87	3421762.07	1810613.80	13:
1	1776875.12	342170.20	1434704.92	13:
2	18186300.21	2904271.23	15282028.98	13:
3	9369861.40	3827220.52	5542640.88	13:
4	5044390.15	4238894.87	805495.28	13:

In [6]:
```
(
    filings
    [[
        "filing_id",
        "Candidate Name",
    ]]
    .merge(
        additional_filing_metadata
        [[
            "filing_id",
            "col_a_individuals_itemized",
            "col_a_individuals_unitemized",
            "col_a_individual_contribution_total",
        ]]
        .assign(
            prop_itemized = lambda df: (
                df["col_a_individuals_itemized"] /
                df["col_a_individual_contribution_total"]
            ).round(3)
        ),
        how = "right",
        on = "filing_id"
    )
    .sort_values("col_a_individuals_itemized", ascending = False)
)
```

Out[6]:

	filing_id	Candidate Name	col_a_individuals_itemized	col_a_individuals_unitemized	col_a_individu
10	1326016	Kamala Harris	7603293.36	4420828.19	
4	1326465	Cory Booker	4238894.87	805495.28	
3	1326481	Beto O'Rourke	3827220.52	5542640.88	
0	1326529	Amy Klobuchar	3421762.07	1810613.80	
2	1326070	Bernie Sanders	2904271.23	15282028.98	
12	1324922	Pete Buttigieg	2549602.40	4536552.22	
11	1326061	Kirsten Gillibrand	2497960.90	499923.20	
8	1326014	John Hickenlooper	1813358.33	200741.04	
5	1326299	Elizabeth Warren	1786711.53	4229723.85	
6	1326136	Jay Inslee	1488634.36	766821.05	
13	1326558	Tulsi Gabbard	881878.80	1067196.12	
9	1326324	Julián Castro	719775.12	373165.90	
1	1326379	Andrew Yang	342170.20	1434704.92	
7	1324749	John Delaney	331244.84	73056.33	
14	1326345	Wayne Messam	31960.00	11571.62	

Contributors

Here, we extract contributions to the committees from individuals, and assign `donor_id` s: a combination of the contributor's listed first name, last name, and five-digit ZIP code.

In [7]:
```
def make_donor_ids(df):
    return (
        df
        .assign(
            donor_id = lambda df: (
                df
                .assign(
                    zip5 = lambda df: (
                        df["contributor_zip_code"]
                        .fillna("-----")
                        .str.slice(0, 5)
                    )
                )
                [[
                    "contributor_first_name",
                    "contributor_last_name",
                    "zip5",
                ]]
                .apply(lambda x: (
                    x
                    .fillna("")
                    .astype(str)
                    # Remove periods, commas, extra whitespace
                    .str.replace(r"[\.,\s]+", " ")
                    .str.strip()
                    # Convert everything to upper-case
                    .str.upper()
                ))
                .apply("|".join, axis = 1)
            )
        )
    )
```

In [8]:
```
def extract_indiv_contributions(filing_id):
    filing = fecfile.from_file(f"../inputs/filings/{int(filing_id)}.fec")
    df = pd.DataFrame(filing["itemizations"]["Schedule A"])
    return (
        df
        # Extract only individual contributions
        .loc[lambda df: df["entity_type"] == "IND"]
        # Remove memo lines
        .loc[lambda df: df["memo_code"] == ""]
        .pipe(make_donor_ids)
        .assign(
            filing_id = int(filing_id)
        )
        [[
            "filer_committee_id_number",
            "filing_id",
            "transaction_id",
            "contribution_date",
            "contribution_amount",
            "contribution_aggregate",
            "donor_id",
            "contributor_first_name",
            "contributor_last_name",
            "contributor_zip_code",
        ]]
    )
```

In [9]:
```
all_indiv_contribs = (
    pd.concat([
        extract_indiv_contributions(filing_id)
        for filing_id in filings["filing_id"]
    ])
    .merge(
        (
            filings
            [[
                "filing_id",
                "committee_id",
            ]]
        ),
        how = "left",
        on = "filing_id",
    )
    .merge(
        (
            candidates
            [[
                "committee_id",
                "Candidate Name",
            ]]
        ),
        how = "left",
        on = "committee_id",
    )
)

len(all_indiv_contribs)
```

Out[9]:
```
92830
```

In [10]:
```
all_indiv_contribs.head()
```

Out[10]:

	filer_committee_id_number	filing_id	transaction_id	contribution_date	contribution_amount	c
0	C00696419	1326529	561500	2019-02-20 00:00:00-05:00	2800.0	
1	C00696419	1326529	564400	2019-02-19 00:00:00-05:00	2800.0	
2	C00696419	1326529	564500	2019-02-18 00:00:00-05:00	250.0	
3	C00696419	1326529	565000	2019-02-19 00:00:00-05:00	100.0	
4	C00696419	1326529	566000	2019-02-19 00:00:00-05:00	25.0	

Aggregate contributions to donor-campaign level

The raw FEC data includes one row for each contribution, so contributors can show up multiple times for a given campaign. Here, we aggregate the data so that it has just one row per contributor-campaign combination:

In [11]:
```
# There appear to be some donors who've been refunded to $200 or less
(
    all_indiv_contribs
    .loc[lambda df: df["contribution_aggregate"] <= 200]
    ["donor_id"]
    .nunique()
)
```

Out[11]:
```
519
```

In [12]:
```
contributor_totals = (
    all_indiv_contribs
    # Line below removes donors who appear to have been refunded
    # to $200 aggregate or less
    .loc[lambda df: df["contribution_aggregate"] > 200]
    .groupby([
        "donor_id",
        "Candidate Name"
    ])
    ["contribution_amount"]
    .sum()
    .reset_index()
)

contributor_totals.head()
```

Out[12]:

	donor_id	Candidate Name	contribution_amount
0	0-DEREK|EILER|30306	Pete Buttigieg	250.0
1	A - DANA SMITH|SMITH|80534	Bernie Sanders	250.0
2	A ALEX|LARI|10128	Kirsten Gillibrand	2700.0
3	A C|HUDGINS|10025	Cory Booker	250.0
4	A J|AGUILA|07631	Andrew Yang	250.0

Distinct donor counts, by candidate

In [13]:
```
distinct_donor_counts = (
    contributor_totals
    ["Candidate Name"]
    .value_counts()
    .to_frame("Distinct Donor IDs")
)

distinct_donor_counts
```

Out[13]:

	Distinct Donor IDs
Bernie Sanders	9321
Kamala Harris	7489
Beto O'Rourke	4879
Pete Buttigieg	4045
Elizabeth Warren	3177
Cory Booker	3063
Amy Klobuchar	2867
Kirsten Gillibrand	1723
Jay Inslee	1214
John Hickenlooper	1090
Tulsi Gabbard	801
Andrew Yang	659
Julián Castro	576
John Delaney	372
Wayne Messam	33

Find donors who gave to any two candidates, and any three candidates

In [14]:
```
candidate_pairs = (
    contributor_totals
    .rename(columns = {
        "Candidate Name": "candidate"
    })
    [[
        "donor_id",
        "candidate"
    ]]
    .pipe(lambda df: (
        df
        .merge(
            df,
            how = "left",
            on = "donor_id",
            suffixes = [ "_x", "_y" ],
        )
    ))
    # This filter prevents us from double-counting candidate-combinations
    .loc[lambda df: df["candidate_x"] < df["candidate_y"]]
    .sort_values([
        "candidate_x",
        "candidate_y",
        "donor_id"
    ])
)

candidate_pairs.head(10)
```

Out[14]:

	donor_id	candidate_x	candidate_y
8287	COLLIER|PERRY|76567	Amy Klobuchar	Andrew Yang
18921	JEAN|YNGVE|46304	Amy Klobuchar	Bernie Sanders
23339	KAREN|ALLIN|30305	Amy Klobuchar	Bernie Sanders
24146	KATHY|GIBBONS|20008	Amy Klobuchar	Bernie Sanders
29269	MARK|MOLLOY|50214	Amy Klobuchar	Bernie Sanders
29344	MARK|ROTHACHER|84117	Amy Klobuchar	Bernie Sanders
29456	MARK|WIZNITZER|22205	Amy Klobuchar	Bernie Sanders
34001	PARKE|CAPSHAW|22902	Amy Klobuchar	Bernie Sanders
963	ALICE|JARCHO|10065	Amy Klobuchar	Beto O'Rourke
1480	AMY|LOFGREN|85250	Amy Klobuchar	Beto O'Rourke

In [15]:
```
candidate_triplets = (
    contributor_totals
    .rename(columns = {
        "Candidate Name": "candidate"
    })
    [[
        "donor_id",
        "candidate"
    ]]
    .pipe(lambda df: (
        df
        .merge(
            df,
            how = "left",
            on = "donor_id",
            suffixes = [ "_x", "_y" ],
        )
        .merge(
            df.rename(columns = { "candidate": "candidate_z" }),
            how = "left",
            on = "donor_id",
        )
    ))
    # This filter prevents us from double-counting candidate-combinations
    .loc[lambda df: df["candidate_x"] < df["candidate_y"]]
    .loc[lambda df: df["candidate_y"] < df["candidate_z"]]
    .sort_values([
        "candidate_x",
        "candidate_y",
        "candidate_z",
        "donor_id"
    ])
)

candidate_triplets.head(10)
```

Out[15]:

	donor_id	candidate_x	candidate_y	candidate_z
10547	COLLIER|PERRY|76567	Amy Klobuchar	Andrew Yang	Elizabeth Warren
43322	PARKE|CAPSHAW|22902	Amy Klobuchar	Bernie Sanders	John Hickenlooper
37377	MARK|MOLLOY|50214	Amy Klobuchar	Bernie Sanders	Kamala Harris
43323	PARKE|CAPSHAW|22902	Amy Klobuchar	Bernie Sanders	Kirsten Gillibrand
29581	KAREN|ALLIN|30305	Amy Klobuchar	Bernie Sanders	Pete Buttigieg
31040	KEENAN|KELSEY|94939	Amy Klobuchar	Beto O'Rourke	Cory Booker
5635	BILL|SIMS|75209	Amy Klobuchar	Beto O'Rourke	Elizabeth Warren
39698	MICHAEL|AUERBACH|10013	Amy Klobuchar	Beto O'Rourke	Elizabeth Warren
2275	ANDREW|FREDMAN|33156	Amy Klobuchar	Beto O'Rourke	John Hickenlooper
2445	ANDREW|MELLETT|90004	Amy Klobuchar	Beto O'Rourke	John Hickenlooper

Identify the most common two-candidate combinations

Here, we count how many times donors has given to both Candidate X and Candidate Y, irrespective of any other contributions they might have made:

In [16]:
```
pair_counts = (
    candidate_pairs
    .groupby([
        "candidate_x",
        "candidate_y",
    ])
    .size()
    .to_frame("count")
    .sort_values("count", ascending = False)
    .reset_index()
)

pair_counts.to_csv(
    "../outputs/candidate-pair-counts.csv",
    index = False
)

pair_counts.loc[lambda df: df["count"] >= 50]
```

Out[16]:

	candidate_x	candidate_y	count
0	Kamala Harris	Pete Buttigieg	170
1	Cory Booker	Kamala Harris	166
2	Beto O'Rourke	Pete Buttigieg	144
3	Amy Klobuchar	Kamala Harris	141
4	Bernie Sanders	Elizabeth Warren	138
5	Kamala Harris	Kirsten Gillibrand	130
6	Beto O'Rourke	Kamala Harris	128
7	Elizabeth Warren	Kamala Harris	121
8	Amy Klobuchar	Pete Buttigieg	112
9	Cory Booker	Kirsten Gillibrand	80
10	Elizabeth Warren	Pete Buttigieg	75
11	Amy Klobuchar	Kirsten Gillibrand	72
12	Bernie Sanders	Tulsi Gabbard	65
13	Amy Klobuchar	Beto O'Rourke	65
14	Amy Klobuchar	Elizabeth Warren	52

Identify the most common three-candidate combinations

In [17]:
```
triplet_counts = (
    candidate_triplets
    .groupby([
        "candidate_x",
        "candidate_y",
        "candidate_z",
    ])
    .size()
    .to_frame("count")
    .sort_values("count", ascending = False)
    .reset_index()
)

triplet_counts.to_csv(
    "../outputs/candidate-triplet-counts.csv",
    index = False
)

triplet_counts.loc[lambda df: df["count"] >= 10]
```

Out[17]:

	candidate_x	candidate_y	candidate_z	count
0	Cory Booker	Kamala Harris	Kirsten Gillibrand	33
1	Amy Klobuchar	Kamala Harris	Kirsten Gillibrand	27
2	Beto O'Rourke	Kamala Harris	Pete Buttigieg	26
3	Amy Klobuchar	Kamala Harris	Pete Buttigieg	22
4	Amy Klobuchar	Beto O'Rourke	Pete Buttigieg	19
5	Elizabeth Warren	Kamala Harris	Pete Buttigieg	17
6	Amy Klobuchar	Elizabeth Warren	Kamala Harris	17
7	Amy Klobuchar	Cory Booker	Kamala Harris	17
8	Elizabeth Warren	Kamala Harris	Kirsten Gillibrand	16
9	Cory Booker	Kamala Harris	Pete Buttigieg	14
10	Amy Klobuchar	Beto O'Rourke	Kamala Harris	10
11	Cory Booker	Elizabeth Warren	Kamala Harris	10
12	Amy Klobuchar	Elizabeth Warren	Kirsten Gillibrand	10
13	Amy Klobuchar	Cory Booker	Kirsten Gillibrand	10
14	Amy Klobuchar	Jay Inslee	Pete Buttigieg	10

Count number of donors who gave to at least three female candidates

In [18]:
```
(
    contributor_totals
    .loc[lambda df: df["Candidate Name"].isin(
        candidates
        .loc[lambda df: df["Candidate Sex"] == "F"]
        ["Candidate Name"]
    )]
    .groupby([ "donor_id" ])
    ["Candidate Name"]
    .nunique()
    .loc[lambda x: x >= 3]
    .pipe(len)
)
```

Out[18]:
```
44
```

Calculate number of donors, per candidate, that gave to multiple campaigns

In [19]:
```
(
    candidate_pairs
    .melt(
        id_vars = [ "donor_id" ],
        value_name = "candidate"
    )
    .groupby([ "candidate" ])
    ["donor_id"]
    .nunique()
    .sort_values(ascending = False)
    .to_frame("Multiple-Candidate Donors")
    .join(
        distinct_donor_counts,
        how = "left"
    )
    .assign(**{
        "Per 1k": lambda df: (
            df["Multiple-Candidate Donors"] * 1000 /
            df["Distinct Donor IDs"]
        ).round(1)
    })
)
```

Out[19]:

	Multiple-Candidate Donors	Distinct Donor IDs	Per 1k
candidate			
Kamala Harris	722	7489	96.4
Pete Buttigieg	512	4045	126.6
Elizabeth Warren	420	3177	132.2
Beto O'Rourke	394	4879	80.8
Amy Klobuchar	386	2867	134.6
Cory Booker	313	3063	102.2
Bernie Sanders	296	9321	31.8
Kirsten Gillibrand	277	1723	160.8
Jay Inslee	128	1214	105.4
Tulsi Gabbard	79	801	98.6
John Hickenlooper	68	1090	62.4
Julián Castro	52	576	90.3
Andrew Yang	22	659	33.4
John Delaney	18	372	48.4
Wayne Messam	2	33	60.6

Calculate total number of donors observed giving to multiple campaigns

In [20]:
```
donor_candidate_counts = (
    contributor_totals
    .groupby(["donor_id"])
    .size()
    .to_frame("num_candidates")
    .reset_index()
)

(
    donor_candidate_counts
    ["num_candidates"]
    .value_counts()
    .sort_index()
)
```

Out[20]:
```
1    37620
2     1401
3      203
4       55
5        8
6        3
Name: num_candidates, dtype: int64
```

... and as a proportion of the total:

In [21]:
```
(
    donor_candidate_counts
    ["num_candidates"]
    .value_counts(normalize = True)
    .sort_index()
)
```

Out[21]:
```
1    0.957496
2    0.035658
3    0.005167
4    0.001400
5    0.000204
6    0.000076
Name: num_candidates, dtype: float64
```