In [1]:
```
import io
import json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys
import seaborn as sb
from scipy.stats import skew
from sklearn import preprocessing
from IPython.display import Javascript
```

In [2]:
```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

In [3]:
```
"""
Cell Description:
This is reading in the three samhsa dataframes outputted from the EDA-pt1-sa
"""
main_samhsa_path = "/content/gdrive/MyDrive/L123-Project-Opiates/EDA/EDA-Dat
ill_dist_multi_path = "/content/gdrive/MyDrive/L123-Project-Opiates/EDA/EDA-
ill_dist_bin_path = "/content/gdrive/MyDrive/L123-Project-Opiates/EDA/EDA-Da

print("Begin Download...")

main_samhsa_df = pd.read_csv(main_samhsa_path)
ill_dist_multi_df = pd.read_csv(ill_dist_multi_path)
ill_dist_bin_df = pd.read_csv(ill_dist_bin_path)

print("SAMHSA Downloaded")
```

```
Begin Download...
SAMHSA Downloaded
```

In [4]:
```
print("main_samhsa_df.shape: ", main_samhsa_df.shape)
# main_samhsa_df.head()
```

```
main_samhsa_df.shape:  (369518, 150)
```

In [5]:
```
print("************************************\n" * 20)
```

```
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
************************************
```

In [6]:
```
print("ill_dist_multi_df.shape: ", ill_dist_multi_df.shape)
# ill_dist_multi_df.head()
```

```
ill_dist_multi_df.shape:  (369518, 9)
```

In [7]:
```
print("ill_dist_multi_df number of nulls: ", ill_dist_multi_df.isna().sum().
# ill_dist_multi_df.info()
```

```
ill_dist_multi_df number of nulls:  0
```

In [8]:
```
# ill_dist_multi_df.describe()
```

In [9]:
```
"""
Cell Description:
This cell attempts to normalize the ill-distributed multinomial data
by determining which type of transformation reduces skew the best.
"""
multi_df = ill_dist_multi_df.head(0)
for col_name in ill_dist_multi_df.columns.values:
  x_sq, x_cub = np.square(ill_dist_multi_df[col_name]), np.power(ill_dist_mu
  x_half, x_log = np.power(ill_dist_multi_df[col_name], .5), np.log(ill_dist
  skews = [abs(skew(x_sq)), abs(skew(x_cub)), abs(skew(x_half)), abs(skew(x_
  best_index = np.argmin(np.array(skews))
  print(best_index)
  if best_index == 0:
    multi_df[col_name] = x_sq
  elif best_index == 1:
    multi_df[col_name] = x_cub
  elif best_index == 2:
    multi_df[col_name] = x_half
  else:
    multi_df[col_name] = x_log

multi_df.head()
```

```
3
2
2
3
3
3
3
3
3
```

Out[9]:

	A_PCT	B_PCT	D_PCT	OPBUPNUM	OPVIVNUM	OPDISNUM	OPNALNUM	OPACA
0	1.609438	1.414214	1.414214	0.000000	0.000000	0.000000	0.000000	0.(
1	1.098612	1.732051	1.732051	0.619039	0.000000	0.000000	0.000000	0.
2	0.000000	1.000000	2.236068	0.000000	0.000000	0.000000	0.000000	0.(
3	0.800119	1.548933	1.750576	0.459777	0.222213	0.060898	0.216611	0.(
4	0.800119	1.548933	1.750576	0.459777	0.222213	0.060898	0.216611	0.(

In [10]:
```
print("**************************************\n" * 20)
```

```
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
**************************************
```

In [11]:
```
print("ill_dist_bin_df.shape: ", ill_dist_bin_df.shape)
# ill_dist_bin_df.head()
```

```
ill_dist_bin_df.shape:  (369518, 20)
```

In [12]:
```
"""
Cell Description:
This cell combines all three of the samhsa datasets into one, large samhsa d
"""
full_samhsa_df = pd.concat([multi_df, ill_dist_bin_df], axis=1)
full_samhsa_df = pd.concat([main_samhsa_df, full_samhsa_df], axis=1)
```

```python
print("full_samhsa_df.shape: ", full_samhsa_df.shape)
# full_samhsa_df.head()
```

```
full_samhsa_df.shape:  (369518, 179)
```

In [13]:
```python
# full_samhsa_df.info()
```

In [14]:
```python
# full_samhsa_df.describe()
```

In [15]:
```python
print("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&\n" * 20)
```

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

In [16]:
```python
"""
Cell Description:
This cell defines the compute_correlation function will be called in
subsequent cells. It applies correlation calculations dependent on the
type of data we're dealing with (numerical/continuous, ordinal, nominal).
This function returns the a correlation matrix of features that
have atleast one correlation value with another feature between [0.5, 1)
"""
def compute_correlation(corr_type:str, df:pd.DataFrame):
  print("Correlation Type: ", corr_type)

  if df.shape[1] == 1:
    return df

  corr_cat = None
  if corr_type == "numericals":
    corr_cat = 'pearson'
  elif corr_type == "ordinals":
    corr_cat = 'kendall'
  else:
    corr_cat = 'spearman'

  df_corr = df.corr(method=corr_cat)
```

```
  abs_df_corr = df_corr.abs()
  np.fill_diagonal(abs_df_corr.values, -2)
  abs_df_corr = abs_df_corr.unstack().sort_values(kind="quicksort")
  abs_df_corr = abs_df_corr[(abs_df_corr >= .5) & (abs_df_corr < 1)].index.t
  print("Top Correlations: ", abs_df_corr)
  top_columns = list(set([item for t in abs_df_corr for item in t]))
  # assert 'STATE' in top_columns
  top_df = df[top_columns]
  print("New Shape: ", top_df.shape)

  plt.figure(figsize=(20, 20))
  mask = np.triu(np.ones_like(top_df.corr()))
  dataplot = sb.heatmap(top_df.corr(), cmap="cividis", annot=False, mask=mas
  plt.show()

  return top_df
```

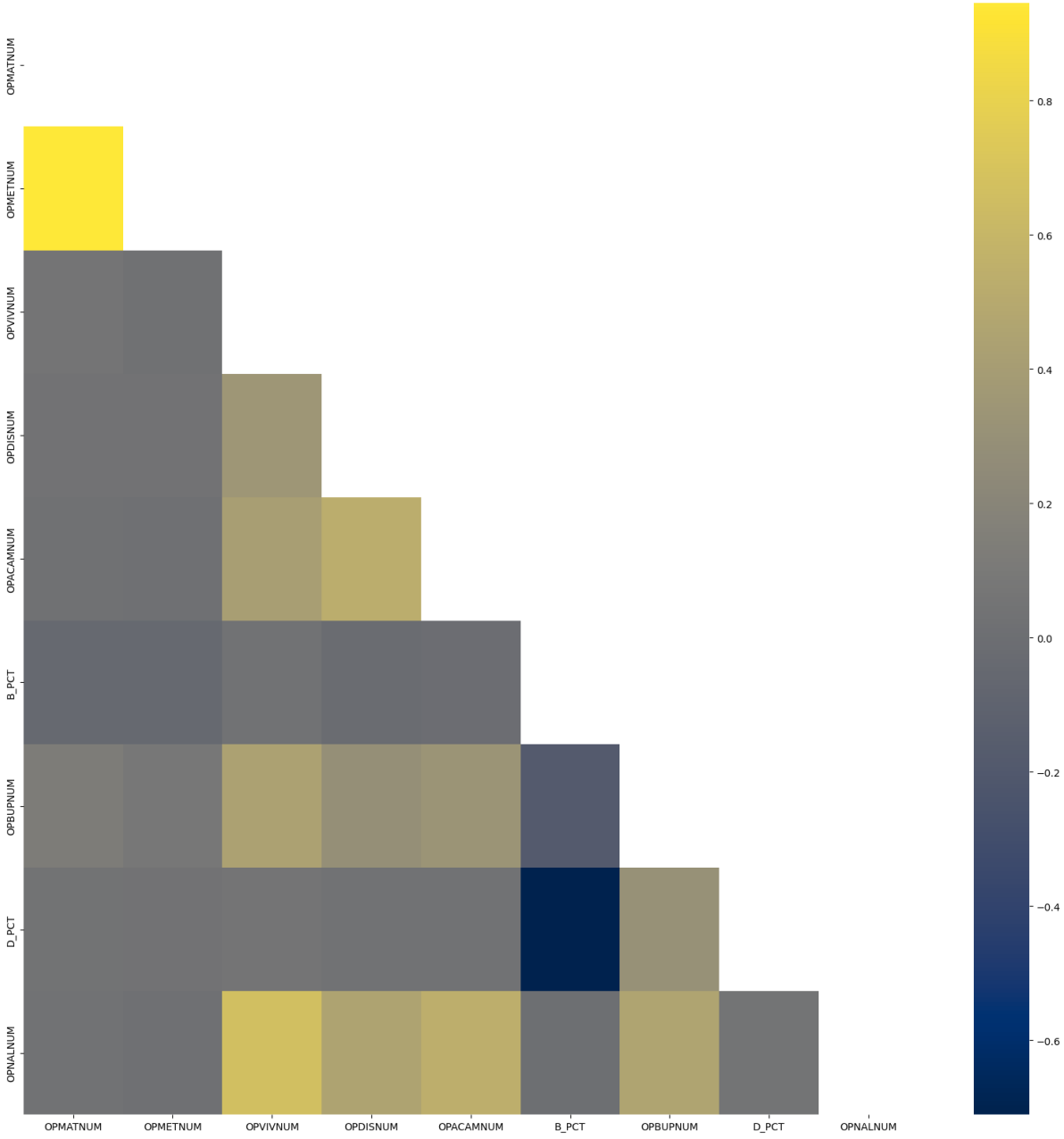
In [17]:
```
print("# Features: ", full_samhsa_df.columns.values.size)
# full_samhsa_df.columns.values
```

```
# Features:  179
```

In [18]:
```
"""
Cell Description:
This cell establishes and calls the function find_correlations,
which first establishes the type of each feature (numerical/continuous, ordi
then calls the above-defined function compute_correlation on the different
dataframe subsets, and then combines all of the features deemed to be releva
from the compute_correlation to create a new subset of features we can
now move forward with. This is a feature selection step.

"""
def find_correlations():
  red_samhsa_df = pd.DataFrame()
  red_samhsa_df['CASEID'] = full_samhsa_df['CASEID']
  red_samhsa_df['STATE'] = full_samhsa_df['STATE']
  numerical_cols = ['Year']

  ordinal_cols = ['HIBUPNUM', 'HIVIVNUM', 'OPBUPNUM', 'OPMETNUM', 'OPVIVNUM'
                  'RCBUPNUM', 'RCMETNUM', 'RCVIVNUM', 'ASSESSMENT', 'OTHER_S
                  'ANCILLARY', 'TESTING', 'TELEMED', 'TRANSITION', 'OPMATNUM
                  'A_PCT', 'B_PCT', 'D_PCT', 'OPDISNUM', 'OPNALNUM', 'OPACAM
                  'O_AGE1', 'RECOVERY', 'EDUCATION', 'T_CLIOP_X']

  nominal_cols = ['OWNERSHP', 'EARMARK', 'SMISEDSUD', 'SRVCODED', 'OTP', 'ON
                  'OPIOIDDETOX', 'OPIOIDDLOFE', 'OPIOIDMAINT', 'OPIOIDNAL',
                  'OPIOIDWDRAW', 'CTYPEML', 'SRVC85', 'REVCHK1', 'SRVC30',
                  'LICEN', 'REVCHK15', 'PAYASST', 'REVCHK5', 'PHARMACOTHERAP
                  'SRVC33', 'SIGNLANG', 'SRVC34', 'SRVC115', 'SRVC11', 'STAT
                  'SRVC31', 'REVCHK17', 'CTYPE3', 'SRVC71', 'SRVC63',
                  'DUI_DWI', 'SRVC10', 'DETOX', 'CTYPE1', 'REVCHK2', 'CTYPE7
                  'SRVC114', 'SRVC32', 'LICENSED', 'SRVC62', 'CTYPE4', 'CTYP
                  'SRVC116', 'SRVC108', 'CTYPERC4', 'LANG', 'SRVC87', 'SRVC6
                  'LOC5', 'CTYPE2', 'ACCRED', 'FEESCALE', 'SRVC61', 'SRVC86'
                  'CTYPE6', 'REVCHK8', 'MEDICAL', 'SRVC113', 'REVCHK10', 'OT
                  'CTYPEOP', 'TREATMT', 'HOSPITAL', 'SRVC6', 'SRVC75', 'REVC
```

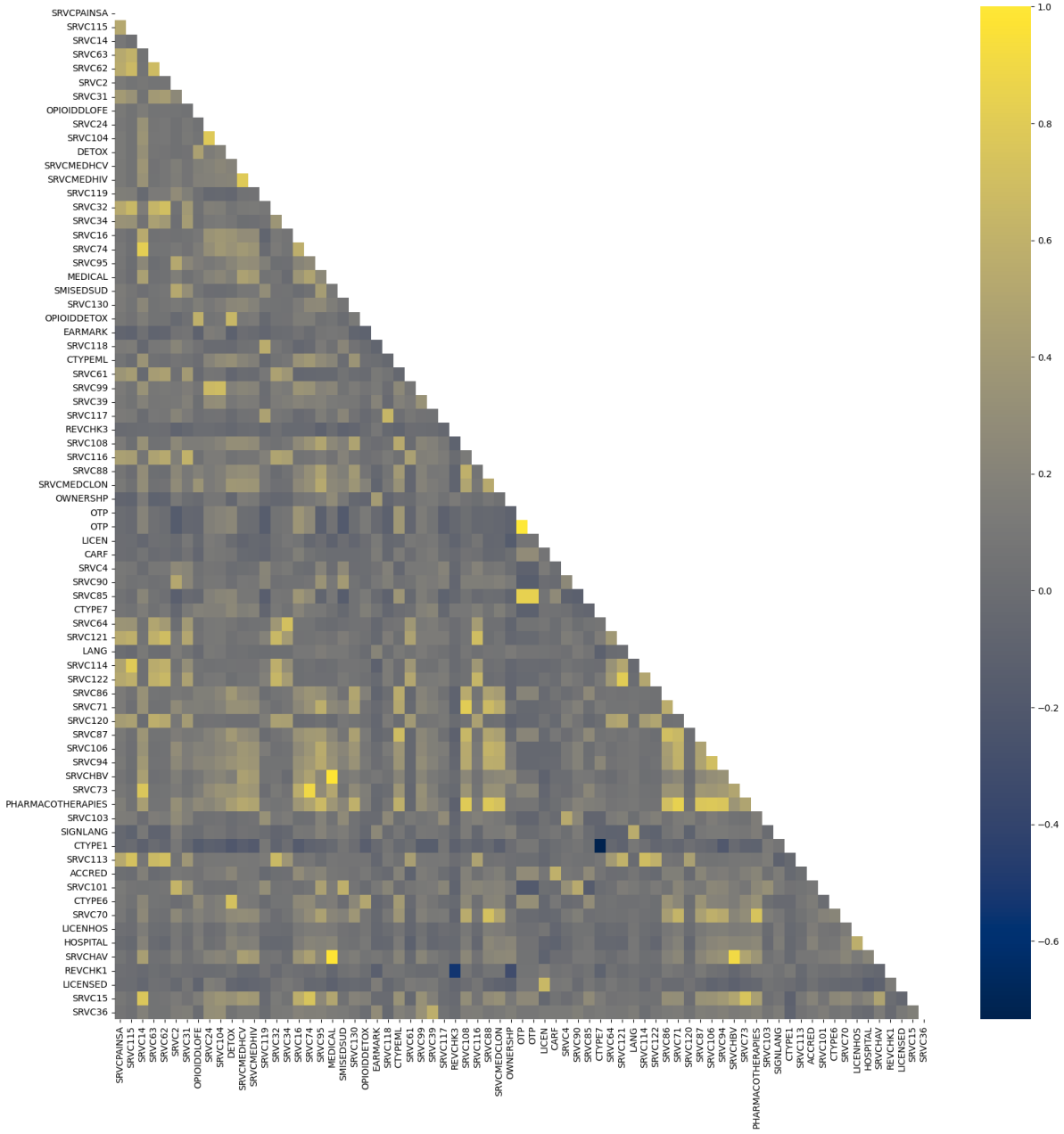
```
                  'NCQA', 'COA', 'HFAP', 'CARF', 'JCAHO','SRVC97', 'SRVC38',
                  'SRVCCOACH', 'SRVC130', 'SRVC95', 'OPIOIDOTH', 'LICENMH',
                  'LICENHOS', 'SRVC94', 'SRVC103', 'SRVC99', 'SRVCMEDHCV', '
                  'SRVC52', 'SRVC90', 'SRVCVOCED', 'SRVC49', 'SRVC88', 'SRVC
                  'SRVCORAL', 'SRVC118', 'SRVCEDCON', 'SRVC35', 'SRVC105', '
                  'SRVC120', 'SRVC96', 'SRVC48', 'SRVCHAV', 'SRVCOUTCM',
                  'SRVC73', 'SRVCMETA', 'SRVC5', 'SRVC121', 'SRVC106', 'SRVC
                  'SRVC39', 'SRVCPAINSA', 'SRVC102', 'SRVC117', 'SRVCMEDHIV'
                  'SRVC119', 'SRVCMEDCLON', 'SRVC70', 'SRVC14', 'SRVC50', 'S
                  'SRVCMEDLOFE', 'SRVC122', 'SRVC16', 'SRVC98', 'SRVC4', 'SR
                  'SRVC59', 'SRVC93', 'SRVC1', 'SRVC74', 'SRVC89', 'SRVC37',
                  'SRVC100', 'SRVC129', 'NOAPPRCH', 'DIALBT', 'BRIEF', 'APPR
                  'REBTHER', 'MATRIXM', 'CONMGMT', '_12STEP', 'SRVC2', 'COGB
                  'SACOUN', 'ANGERM', 'TRAUMAC', 'MOTIVATE', 'RELPREV', 'CRV

  assert len(set(ordinal_cols)) + len(set(nominal_cols)) + len(numerical_col

  numericals_df = full_samhsa_df[numerical_cols]
  ordinals_df = full_samhsa_df[ordinal_cols]
  nominals_df = full_samhsa_df[nominal_cols]
  dfs = [numericals_df, ordinals_df, nominals_df]
  corr_types = ['numericals', 'ordinals', 'nominals']
  for i in range(3):
    result_df = compute_correlation(corr_types[i], dfs[i])
    red_samhsa_df = pd.concat([red_samhsa_df, result_df], axis=1)

  red_samhsa_df['A_PCT'] = full_samhsa_df['A_PCT']
  return red_samhsa_df


simple_samhsa_df = find_correlations()
print("Simple Shape: ", simple_samhsa_df.shape)
print("Simple Features: ", simple_samhsa_df.columns.values)
simple_samhsa_df = simple_samhsa_df.round(0).astype(int)
# simple_samhsa_df.head()
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeigh
```

```
Correlation Type:  numericals
Correlation Type:  ordinals
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:5278: Runti
meWarning: overflow encountered in long_scalars
  (2 * xtie * ytie) / m + x0 * y0 / (9 * m * (size - 2)))
Top Correlations:  [('OPNALNUM', 'OPBUPNUM'), ('OPBUPNUM', 'OPNALNUM'), ('O
PBUPNUM', 'OPVIVNUM'), ('OPVIVNUM', 'OPBUPNUM'), ('OPVIVNUM', 'OPDISNUM'),
('OPDISNUM', 'OPVIVNUM'), ('OPVIVNUM', 'OPACAMNUM'), ('OPACAMNUM', 'OPVIVNU
M'), ('D_PCT', 'B_PCT'), ('B_PCT', 'D_PCT'), ('OPNALNUM', 'OPDISNUM'), ('OP
DISNUM', 'OPNALNUM'), ('OPACAMNUM', 'OPNALNUM'), ('OPNALNUM', 'OPACAMNUM'),
('OPNALNUM', 'OPVIVNUM'), ('OPVIVNUM', 'OPNALNUM'), ('OPACAMNUM', 'OPDISNU
M'), ('OPDISNUM', 'OPACAMNUM'), ('OPMATNUM', 'OPMETNUM'), ('OPMETNUM', 'OPM
ATNUM')]
New Shape:  (369518, 9)
```


OPMATNUM
OPMETNUM
OPVIVNUM
OPDISNUM
OPACAMNUM
B_PCT
OPBUPNUM
D_PCT
OPNALNUM
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
Correlation Type:  nominals
Top Correlations:  [('SRVC130', 'SRVC86'), ('SRVC86', 'SRVC130'), ('SRVC1
5', 'MEDICAL'), ('MEDICAL', 'SRVC15'), ('SRVC73', 'SRVCHBV'), ('SRVCHBV',
'SRVC73'), ('SRVC70', 'SRVCMEDCLON'), ('SRVCMEDCLON', 'SRVC70'), ('SRVC63',
'SRVC64'), ('SRVC64', 'SRVC63'), ('SRVC88', 'SRVC87'), ('SRVC87', 'SRVC8
8'), ('SRVC31', 'SRVC121'), ('SRVC121', 'SRVC31'), ('CTYPEML', 'SRVC71'),
('SRVC71', 'CTYPEML'), ('SRVC95', 'SRVC108'), ('SRVC108', 'SRVC95'), ('SRVC
73', 'MEDICAL'), ('MEDICAL', 'SRVC73'), ('SRVC122', 'SRVC61'), ('SRVC61',
'SRVC122'), ('SRVC122', 'SRVC114'), ('SRVC114', 'SRVC122'), ('SRVC120', 'SR
VC121'), ('SRVC121', 'SRVC120'), ('SRVC121', 'SRVC114'), ('SRVC114', 'SRVC1
21'), ('SRVC90', 'SRVC2'), ('SRVC2', 'SRVC90'), ('SRVCPAINSA', 'SRVC122'),
('SRVC122', 'SRVCPAINSA'), ('PHARMACOTHERAPIES', 'SRVC130'), ('SRVC130', 'P
HARMACOTHERAPIES'), ('SRVCPAINSA', 'SRVC115'), ('SRVC115', 'SRVCPAINSA'),
('SRVC120', 'SRVC62'), ('SRVC62', 'SRVC120'), ('SMISEDSUD', 'SRVC2'), ('SRV
C2', 'SMISEDSUD'), ('SRVC2', 'SRVC95'), ('SRVC95', 'SRVC2'), ('SRVCMEDCLO
N', 'SRVC95'), ('SRVC95', 'SRVCMEDCLON'), ('SRVCPAINSA', 'SRVC63'), ('SRVC6
3', 'SRVCPAINSA'), ('SRVC108', 'CTYPEML'), ('CTYPEML', 'SRVC108'), ('SRVC11
6', 'SRVC62'), ('SRVC62', 'SRVC116'), ('SRVCPAINSA', 'SRVC113'), ('SRVC11
3', 'SRVCPAINSA'), ('SRVC108', 'SRVC94'), ('SRVC94', 'SRVC108'), ('REVCHK
1', 'REVCHK3'), ('REVCHK3', 'REVCHK1'), ('SRVCMEDCLON', 'SRVC88'), ('SRVC8
8', 'SRVCMEDCLON'), ('SRVC108', 'SRVC106'), ('SRVC106', 'SRVC108'), ('SRVC9
4', 'SRVC70'), ('SRVC70', 'SRVC94'), ('SRVC4', 'SRVC103'), ('SRVC103', 'SRV
C4'), ('SRVC32', 'SRVCPAINSA'), ('SRVCPAINSA', 'SRVC32'), ('SRVC94', 'SRVC9
5'), ('SRVC95', 'SRVC94'), ('SRVC86', 'SRVC108'), ('SRVC108', 'SRVC86'),
('SRVC86', 'CTYPEML'), ('CTYPEML', 'SRVC86'), ('SRVC101', 'SRVC90'), ('SRVC
90', 'SRVC101'), ('SRVC14', 'SRVC16'), ('SRVC16', 'SRVC14'), ('SRVC121', 'S
RVCPAINSA'), ('SRVCPAINSA', 'SRVC121'), ('SRVC106', 'SRVC95'), ('SRVC95',
'SRVC106'), ('SRVCPAINSA', 'SRVC62'), ('SRVC62', 'SRVCPAINSA'), ('SRVC108',
'SRVC70'), ('SRVC70', 'SRVC108'), ('SIGNLANG', 'LANG'), ('LANG', 'SIGNLAN
G'), ('SRVC114', 'SRVC63'), ('SRVC63', 'SRVC114'), ('SRVC94', 'SRVCMEDCLO
N'), ('SRVCMEDCLON', 'SRVC94'), ('SRVC119', 'SRVC118'), ('SRVC118', 'SRVC11
9'), ('SRVC16', 'SRVC73'), ('SRVC73', 'SRVC16'), ('SRVC86', 'SRVC71'), ('SR
VC71', 'SRVC86'), ('EARMARK', 'OWNERSHP'), ('OWNERSHP', 'EARMARK'), ('SRVCM
EDCLON', 'SRVC71'), ('SRVC71', 'SRVCMEDCLON'), ('SRVC94', 'SRVC88'), ('SRVC
88', 'SRVC94'), ('LICENHOS', 'HOSPITAL'), ('HOSPITAL', 'LICENHOS'), ('SRVC7
1', 'SRVC95'), ('SRVC95', 'SRVC71'), ('SRVC16', 'SRVC15'), ('SRVC15', 'SRVC
16'), ('SRVC63', 'SRVC120'), ('SRVC120', 'SRVC63'), ('SRVC16', 'SRVC74'),
('SRVC74', 'SRVC16'), ('SRVC106', 'SRVCMEDCLON'), ('SRVCMEDCLON', 'SRVC10
6'), ('SRVC39', 'SRVC36'), ('SRVC36', 'SRVC39'), ('SRVC115', 'SRVC122'),
('SRVC122', 'SRVC115'), ('SMISEDSUD', 'SRVC101'), ('SRVC101', 'SMISEDSUD'),
('SRVC32', 'SRVC114'), ('SRVC114', 'SRVC32'), ('SRVC121', 'SRVC115'), ('SRV
C115', 'SRVC121'), ('SRVC70', 'SRVC106'), ('SRVC106', 'SRVC70'), ('SRVC63',
'SRVC122'), ('SRVC122', 'SRVC63'), ('SRVC121', 'SRVC63'), ('SRVC63', 'SRVC1
21'), ('SRVC88', 'SRVC108'), ('SRVC108', 'SRVC88'), ('OPIOIDDLOFE', 'OPIOID
DETOX'), ('OPIOIDDETOX', 'OPIOIDDLOFE'), ('SRVC62', 'SRVC114'), ('SRVC114',
'SRVC62'), ('SRVC113', 'SRVC122'), ('SRVC122', 'SRVC113'), ('CARF', 'ACCRE
D'), ('ACCRED', 'CARF'), ('SRVC94', 'SRVC71'), ('SRVC71', 'SRVC94'), ('SRVC
116', 'SRVC31'), ('SRVC31', 'SRVC116'), ('SRVC63', 'SRVC115'), ('SRVC115',
'SRVC63'), ('SRVC71', 'SRVC106'), ('SRVC106', 'SRVC71'), ('SRVC121', 'SRVC1
13'), ('SRVC113', 'SRVC121'), ('SRVC106', 'SRVC88'), ('SRVC88', 'SRVC106'),
('SRVC101', 'SRVC2'), ('SRVC2', 'SRVC101'), ('PHARMACOTHERAPIES', 'CTYPEM
L'), ('CTYPEML', 'PHARMACOTHERAPIES'), ('SRVC71', 'SRVC70'), ('SRVC70', 'SR
VC71'), ('SRVC32', 'SRVC63'), ('SRVC63', 'SRVC32'), ('SRVC122', 'SRVC116'),
('SRVC116', 'SRVC122'), ('SRVC117', 'SRVC118'), ('SRVC118', 'SRVC117'), ('S
RVC122', 'SRVC32'), ('SRVC32', 'SRVC122'), ('SRVC87', 'SRVC108'), ('SRVC10
8', 'SRVC87'), ('LICENSED', 'LICEN'), ('LICEN', 'LICENSED'), ('SRVC62', 'SR
```

```
VC122'), ('SRVC122', 'SRVC62'), ('SRVC71', 'SRVC87'), ('SRVC87', 'SRVC71'),
('SRVC32', 'SRVC115'), ('SRVC115', 'SRVC32'), ('SRVC121', 'SRVC32'), ('SRVC
32', 'SRVC121'), ('SRVC63', 'SRVC113'), ('SRVC113', 'SRVC63'), ('SRVC32',
'SRVC113'), ('SRVC113', 'SRVC32'), ('SRVC115', 'SRVC62'), ('SRVC62', 'SRVC1
15'), ('SRVC99', 'SRVC24'), ('SRVC24', 'SRVC99'), ('SRVC88', 'SRVC71'), ('S
RVC71', 'SRVC88'), ('SRVC63', 'SRVC62'), ('SRVC62', 'SRVC63'), ('SRVC87',
'CTYPEML'), ('CTYPEML', 'SRVC87'), ('SRVC70', 'PHARMACOTHERAPIES'), ('PHARM
ACOTHERAPIES', 'SRVC70'), ('SRVC86', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAP
IES', 'SRVC86'), ('SRVC99', 'SRVC104'), ('SRVC104', 'SRVC99'), ('SRVC116',
'SRVC121'), ('SRVC121', 'SRVC116'), ('SRVC62', 'SRVC121'), ('SRVC121', 'SRV
C62'), ('SRVCMEDCLON', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIES', 'SRVCMED
CLON'), ('SRVC114', 'SRVC113'), ('SRVC113', 'SRVC114'), ('SRVC87', 'SRVC8
6'), ('SRVC86', 'SRVC87'), ('SRVC106', 'SRVC94'), ('SRVC94', 'SRVC106'),
('DETOX', 'CTYPE6'), ('CTYPE6', 'DETOX'), ('SRVC113', 'SRVC62'), ('SRVC62',
'SRVC113'), ('SRVC32', 'SRVC62'), ('SRVC62', 'SRVC32'), ('PHARMACOTHERAPIE
S', 'SRVC88'), ('SRVC88', 'PHARMACOTHERAPIES'), ('SRVC64', 'SRVC34'), ('SRV
C34', 'SRVC64'), ('SRVC106', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIES', 'S
RVC106'), ('SRVC88', 'SRVC70'), ('SRVC70', 'SRVC88'), ('SRVC113', 'SRVC11
5'), ('SRVC115', 'SRVC113'), ('CTYPE7', 'CTYPE1'), ('CTYPE1', 'CTYPE7'),
('SRVC95', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIES', 'SRVC95'), ('SRVC9
4', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIES', 'SRVC94'), ('SRVC15', 'SRVC
73'), ('SRVC73', 'SRVC15'), ('SRVC74', 'SRVC15'), ('SRVC15', 'SRVC74'), ('S
RVC15', 'SRVC14'), ('SRVC14', 'SRVC15'), ('SRVC14', 'SRVC73'), ('SRVC73',
'SRVC14'), ('SRVC87', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIES', 'SRVC8
7'), ('PHARMACOTHERAPIES', 'SRVC108'), ('SRVC108', 'PHARMACOTHERAPIES'),
('SRVC104', 'SRVC24'), ('SRVC24', 'SRVC104'), ('SRVC114', 'SRVC115'), ('SRV
C115', 'SRVC114'), ('SRVCMEDHCV', 'SRVCMEDHIV'), ('SRVCMEDHIV', 'SRVCMEDHC
V'), ('SRVC71', 'SRVC108'), ('SRVC108', 'SRVC71'), ('SRVC121', 'SRVC122'),
('SRVC122', 'SRVC121'), ('SRVC71', 'PHARMACOTHERAPIES'), ('PHARMACOTHERAPIE
S', 'SRVC71'), ('SRVC74', 'SRVC14'), ('SRVC14', 'SRVC74'), ('SRVC85', 'OT
P'), ('OTP', 'SRVC85'), ('OTP', 'SRVC85'), ('SRVC85', 'OTP'), ('SRVC73', 'S
RVC74'), ('SRVC74', 'SRVC73'), ('SRVCHBV', 'SRVCHAV'), ('SRVCHAV', 'SRVCHB
V'), ('SRVCHBV', 'MEDICAL'), ('MEDICAL', 'SRVCHBV'), ('SRVCHAV', 'MEDICA
L'), ('MEDICAL', 'SRVCHAV')]
New Shape:  (369518, 73)
```




```
Simple Shape:  (369518, 86)
Simple Features:  ['CASEID' 'STATE' 'Year' 'OPMATNUM' 'OPMETNUM' 'OPVIVNUM'
'OPDISNUM'
 'OPACAMNUM' 'B_PCT' 'OPBUPNUM' 'D_PCT' 'OPNALNUM' 'SRVCPAINSA' 'SRVC115'
 'SRVC14' 'SRVC63' 'SRVC62' 'SRVC2' 'SRVC31' 'OPIOIDDLOFE' 'SRVC24'
 'SRVC104' 'DETOX' 'SRVCMEDHCV' 'SRVCMEDHIV' 'SRVC119' 'SRVC32' 'SRVC34'
 'SRVC16' 'SRVC74' 'SRVC95' 'MEDICAL' 'SMISEDSUD' 'SRVC130' 'OPIOIDDETOX'
 'EARMARK' 'SRVC118' 'CTYPEML' 'SRVC61' 'SRVC99' 'SRVC39' 'SRVC117'
 'REVCHK3' 'SRVC108' 'SRVC116' 'SRVC88' 'SRVCMEDCLON' 'OWNERSHP' 'OTP'
 'OTP' 'LICEN' 'CARF' 'SRVC4' 'SRVC90' 'SRVC85' 'CTYPE7' 'SRVC64'
 'SRVC121' 'LANG' 'SRVC114' 'SRVC122' 'SRVC86' 'SRVC71' 'SRVC120' 'SRVC87'
 'SRVC106' 'SRVC94' 'SRVCHBV' 'SRVC73' 'PHARMACOTHERAPIES' 'SRVC103'
 'SIGNLANG' 'CTYPE1' 'SRVC113' 'ACCRED' 'SRVC101' 'CTYPE6' 'SRVC70'
 'LICENHOS' 'HOSPITAL' 'SRVCHAV' 'REVCHK1' 'LICENSED' 'SRVC15' 'SRVC36'
 'A_PCT']
```

In [19]:
```
# simple_samhsa_df.info()
```

In [20]:
```python
simple_samhsa_df.describe()
```

Out[20]:

	CASEID	STATE	Year	OPMATNUM	OPMETNUM	
count	369518.000000	369518.000000	369518.000000	369518.000000	369518.000000	369!
mean	8033.500000	23.231358	2009.260870	3.066365	2.916862	
std	4637.860979	14.744807	6.562202	0.881035	0.881636	
min	1.000000	0.000000	1997.000000	1.000000	1.000000	
25%	4017.000000	10.000000	2004.000000	2.000000	2.000000	
50%	8033.500000	22.000000	2009.000000	3.000000	3.000000	
75%	12050.000000	35.000000	2015.000000	4.000000	4.000000	
max	16066.000000	52.000000	2020.000000	5.000000	5.000000	

8 rows × 86 columns

In [21]:
```python
print("GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW
```

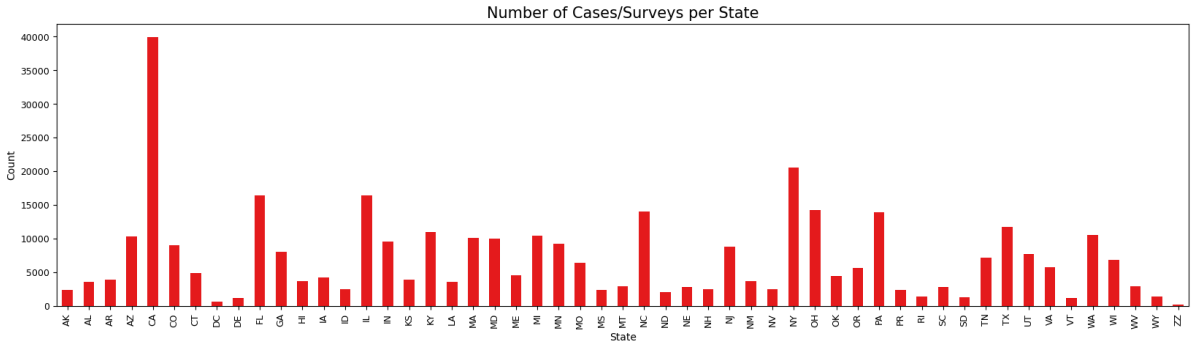
```
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
GRAPHING BELOW!    GRAPHING BELOW    GRAPHING BELOW    GRAPHING BELOW!
```

In [22]:
```python
"""
Cell Description:
This calls in the label_encoder for the STATE column that was created and sa
earlier, and does a reverse transform on the column so that we can have mean
labels for the graphing below (ie we want a graph showing UTAH instead of it
arbitrary label of 6 or whatever it is)
"""
le = preprocessing.LabelEncoder()
le.classes_ = np.load('/content/gdrive/MyDrive/L123-Project-Opiates/EDA/stat
simple_samhsa_df['STATE'] = le.inverse_transform(simple_samhsa_df['STATE'])
```

In [23]:
```python
"""
```

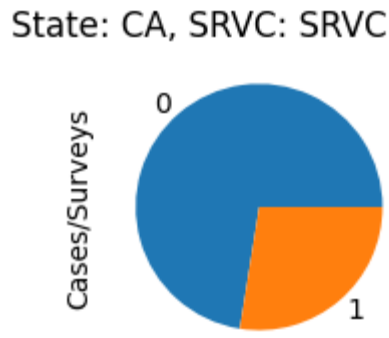
```
Cell Description:
This graphs the number of records per state.
"""
simple_samhsa_df['STATE'].value_counts().sort_index().plot(kind='bar', xlabe
plt.title("Number of Cases/Surveys per State", fontsize=15)
```

Out[23]: Text(0.5, 1.0, 'Number of Cases/Surveys per State')

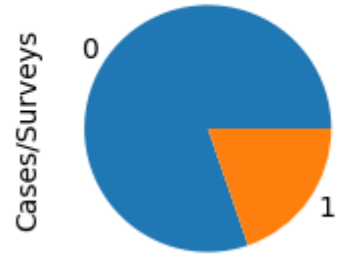


In [24]:

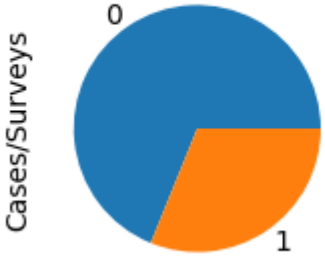
```
"""
Cell Description:
This plots multiple pie graphs based on what proportion of yes and no respon
per SRVC feature.
"""
def plot_state_vs_srvc(state:str):
  srvc_boolean_cols = [feature for feature in simple_samhsa_df.columns.value
  state_df = simple_samhsa_df[simple_samhsa_df['STATE'] == state]
  for srvc_col in srvc_boolean_cols:
    srvc_df = state_df.groupby(srvc_col).count().plot(kind="pie", y='CASEID'
    plt.title("State: {state}, SRVC: {srvc_col}".format(state=state, srvc_co
    plt.ylabel("Cases/Surveys")
    plt.show()

"""To plot ALL states vs ALL SRVCs"""
# states = simple_samhsa_df['STATE'].unique()
# for state in states:
#   plot_state_vs_srvc(state)

"""To plot specific state vs ALL SRVCs.
Note: Input arg must be string State Acronym"""
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeigh
plot_state_vs_srvc('CA')
```



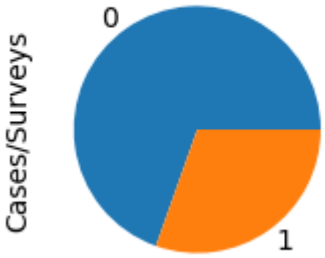
State: CA, SRVC: SRVC115



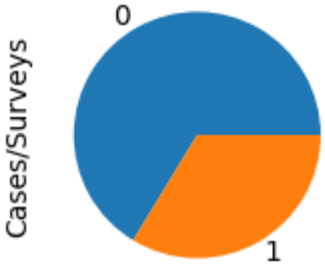
State: CA, SRVC: SRVC14



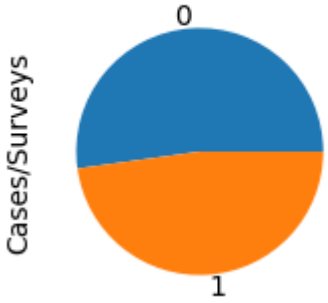
State: CA, SRVC: SRVC63



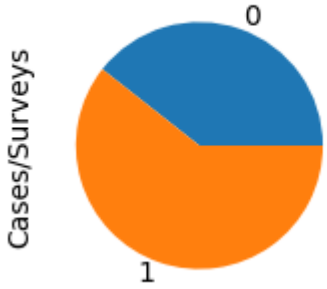
State: CA, SRVC: SRVC62



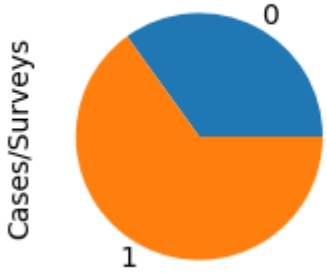
State: CA, SRVC: SRVC2



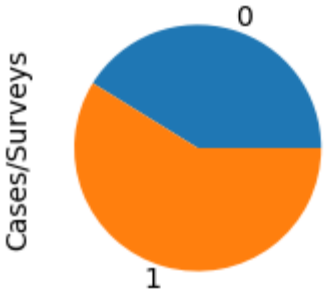
State: CA, SRVC: SRVC31



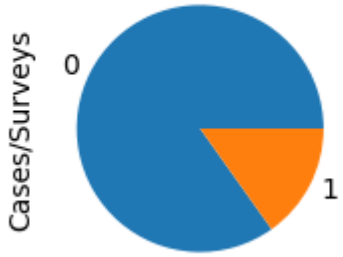
State: CA, SRVC: SRVC24



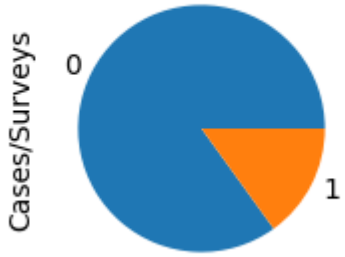
State: CA, SRVC: SRVC104



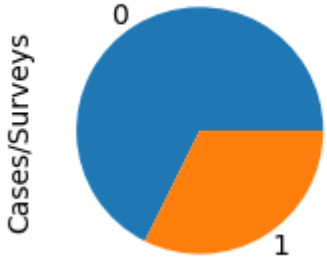
State: CA, SRVC: SRVCMEDHCV



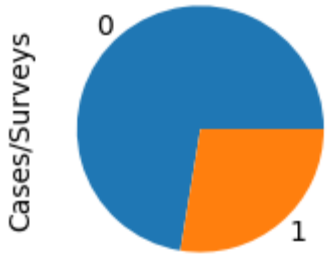
State: CA, SRVC: SRVCMEDHIV



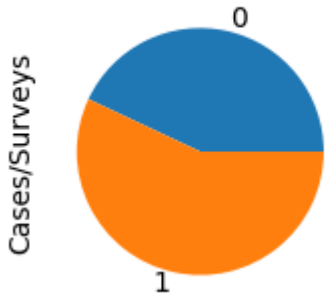
State: CA, SRVC: SRVC119



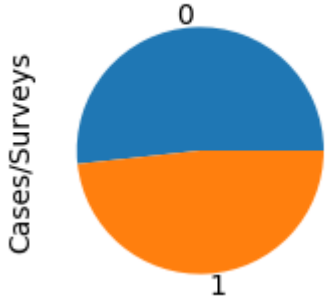
State: CA, SRVC: SRVC32



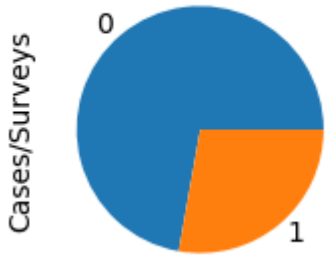
State: CA, SRVC: SRVC34



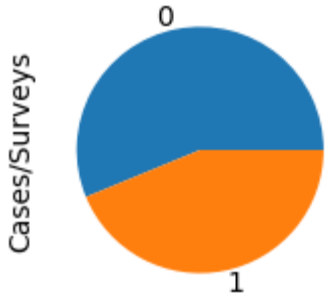
State: CA, SRVC: SRVC16



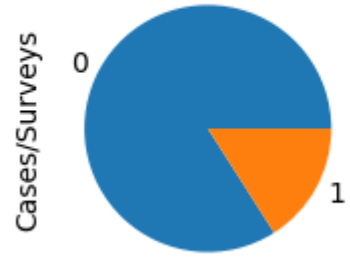
State: CA, SRVC: SRVC74



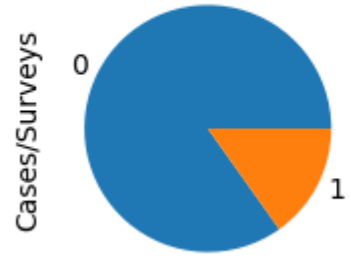
State: CA, SRVC: SRVC95



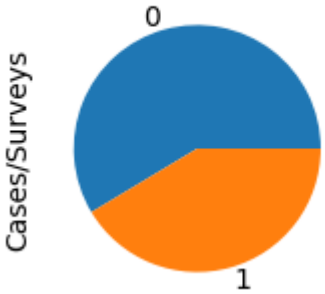
State: CA, SRVC: SRVC130



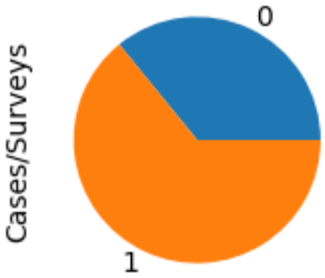
State: CA, SRVC: SRVC118



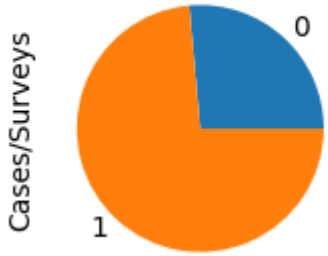
State: CA, SRVC: SRVC61



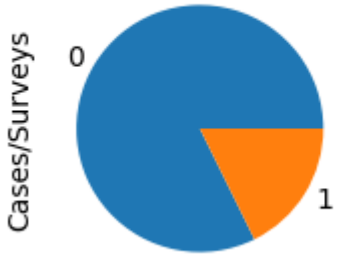
State: CA, SRVC: SRVC99



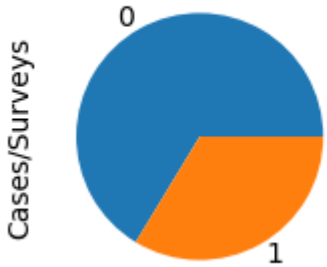
State: CA, SRVC: SRVC39



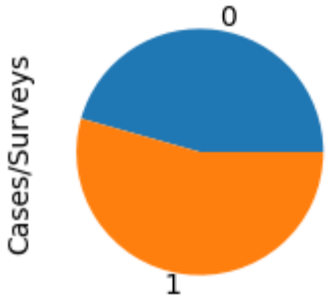
State: CA, SRVC: SRVC117



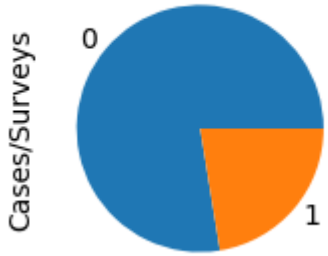
State: CA, SRVC: SRVC108



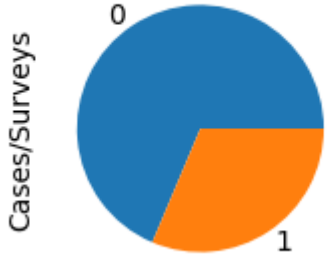
State: CA, SRVC: SRVC116



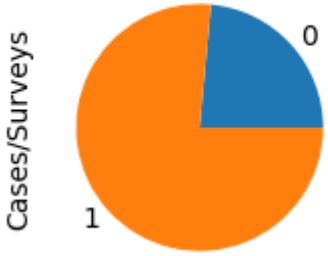
State: CA, SRVC: SRVC88



State: CA, SRVC: SRVCMEDCLON

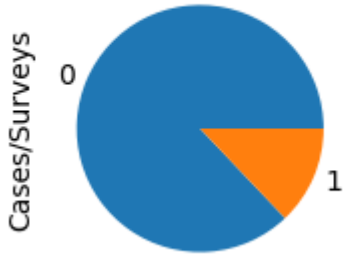


State: CA, SRVC: SRVC4

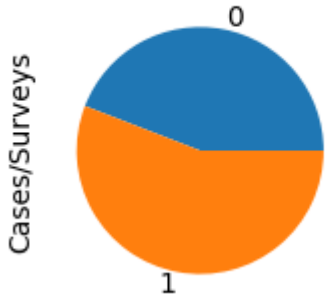


State: CA, SRVC: SRVC90

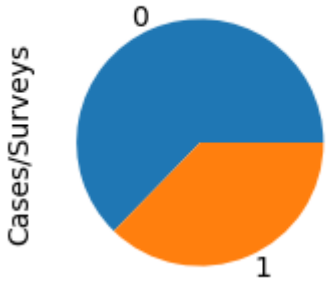
State: CA, SRVC: SRVC85



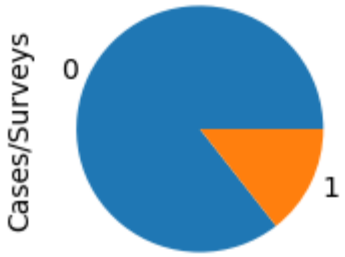
State: CA, SRVC: SRVC64



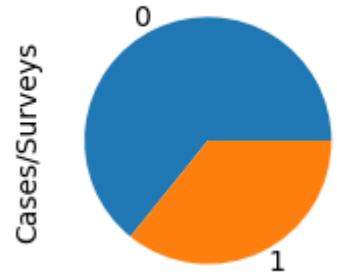
State: CA, SRVC: SRVC121



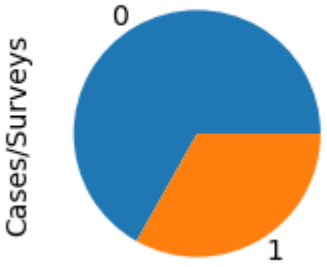
State: CA, SRVC: SRVC114



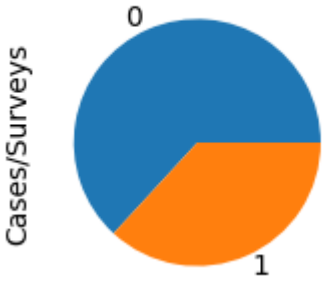
State: CA, SRVC: SRVC122



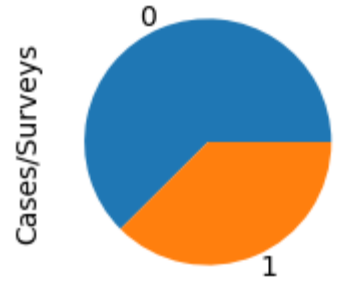
State: CA, SRVC: SRVC86



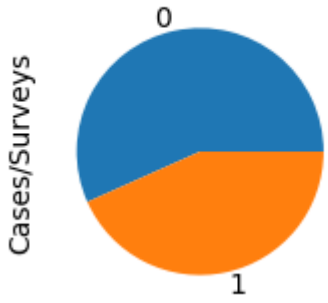
State: CA, SRVC: SRVC71



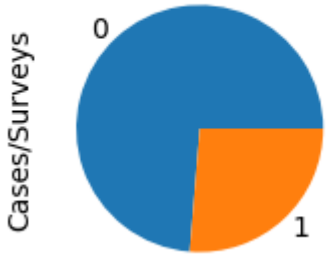
State: CA, SRVC: SRVC120



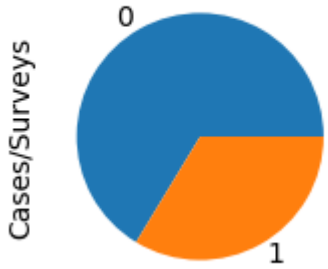
State: CA, SRVC: SRVC87



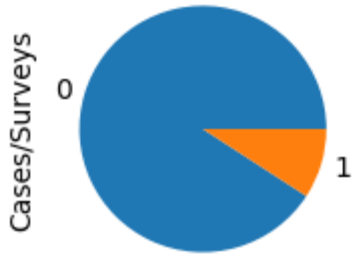
State: CA, SRVC: SRVC106



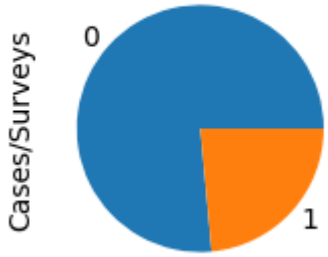
State: CA, SRVC: SRVC94



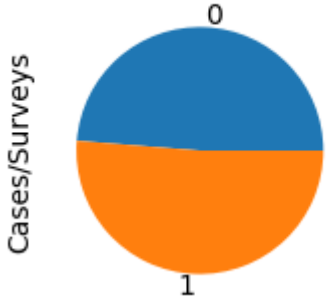
State: CA, SRVC: SRVCHBV



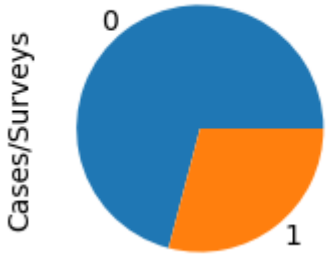
State: CA, SRVC: SRVC73



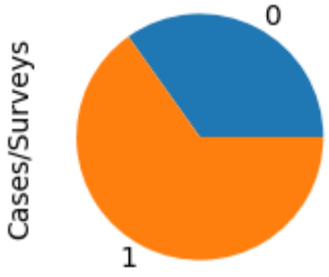
State: CA, SRVC: SRVC103



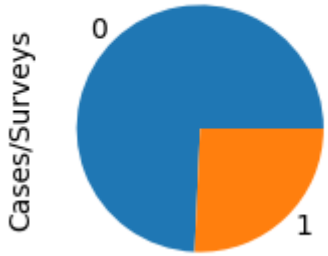
State: CA, SRVC: SRVC113



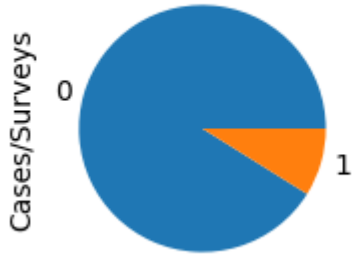
State: CA, SRVC: SRVC101



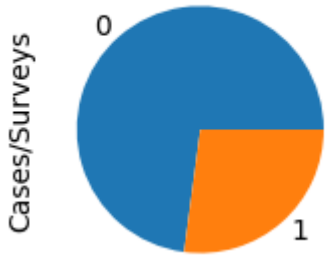
State: CA, SRVC: SRVC70



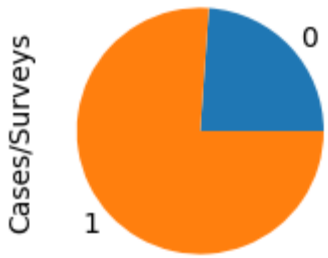
State: CA, SRVC: SRVCHAV



State: CA, SRVC: SRVC15



State: CA, SRVC: SRVC36



In [25]:
```
#Spacer Cell
```

<==>
<==>
<==>

Below is Screenshot Example of what's described below:

<==========>

<==========>

<==========>

OPMETNUM: Total methadone outpatients - For OTP facilities

31c. How many of the clients from the OUTPATIENT TOTAL BOX received:

1. Methadone dispensed at this facility

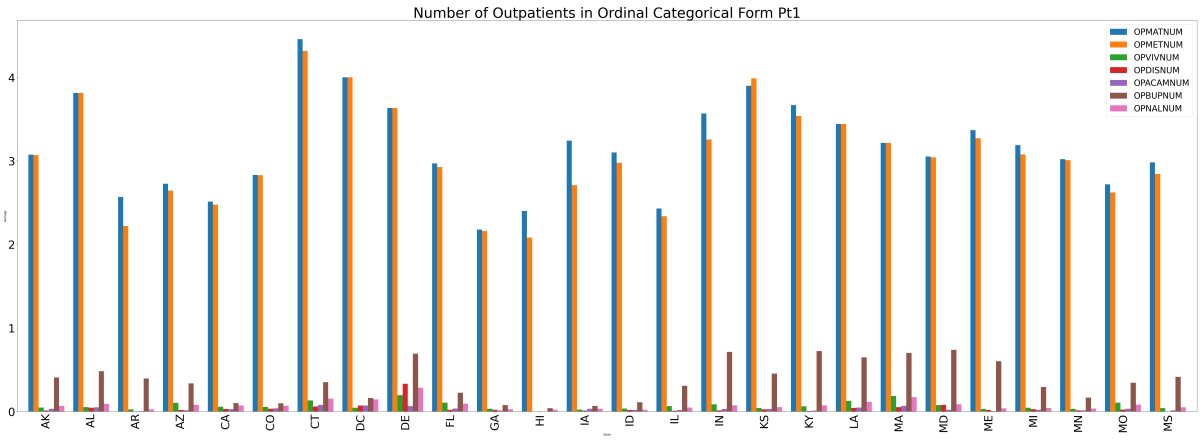
Enter a number for each (if none, enter "0")

Value	Label	Unweighted Frequency	%
1	0-114	299	22.2%
2	115-196	208	15.4%
3	197-293	209	15.5%
4	294-435	219	16.2%
5	436+	272	20.2%

In [26]:
```
print("\nThe below two graphs show averages based on ordinal categorical var
"""
Note 1: Example of what that means is above!

Very Important Note 2: The following features have already been centered fro
[A_PCT  B_PCT   D_PCT   OPBUPNUM        OPVIVNUM        OPDISNUM        OPNA
Some of these features are plotted in the two plots below!!!!!
"""


outpatient_freq_cols = [feature for feature in simple_samhsa_df.columns.valu
grpd = simple_samhsa_df.groupby("STATE").mean()[outpatient_freq_cols]
half_index = int((grpd.shape[0])/2)
grpd[:half_index].plot(kind='bar', xlabel='State', ylabel='Mean', rot=90, fo
plt.title("Number of Outpatients in Ordinal Categorical Form Pt1", fontsize=
plt.xlabel('State')
plt.ylabel('Average')
plt.legend(fontsize=30)
```

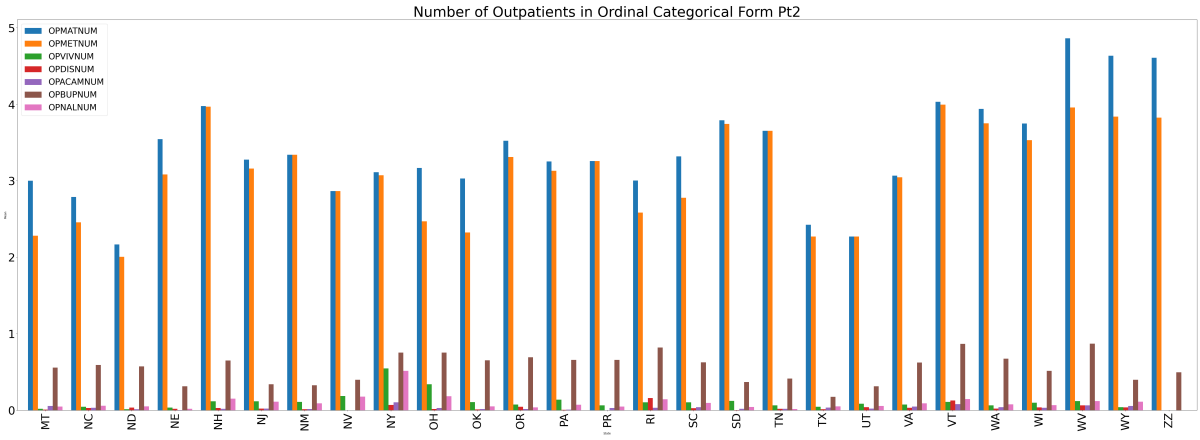
```
The below two graphs show averages based on ordinal categorical variables r
elated to Outpatient care
```

Out[26]:
```
<matplotlib.legend.Legend at 0x7fe3ef3cfdf0>
```



In [27]:
```
grpd[half_index:].plot(kind='bar', xlabel='State', ylabel='Mean', rot=90, fo
plt.title("Number of Outpatients in Ordinal Categorical Form Pt2", fontsize=
plt.legend(fontsize=30)
```

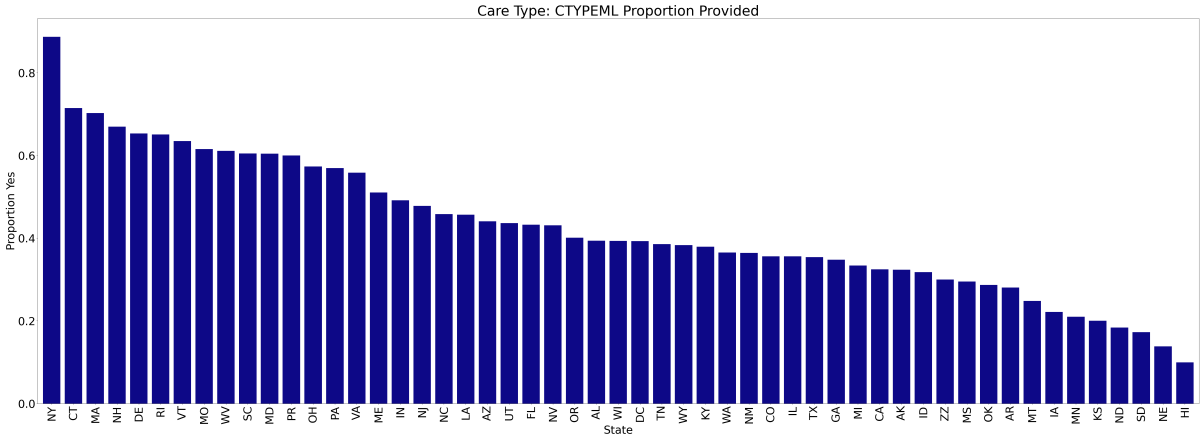
Out[27]:
```
<matplotlib.legend.Legend at 0x7fe3e8557910>
```

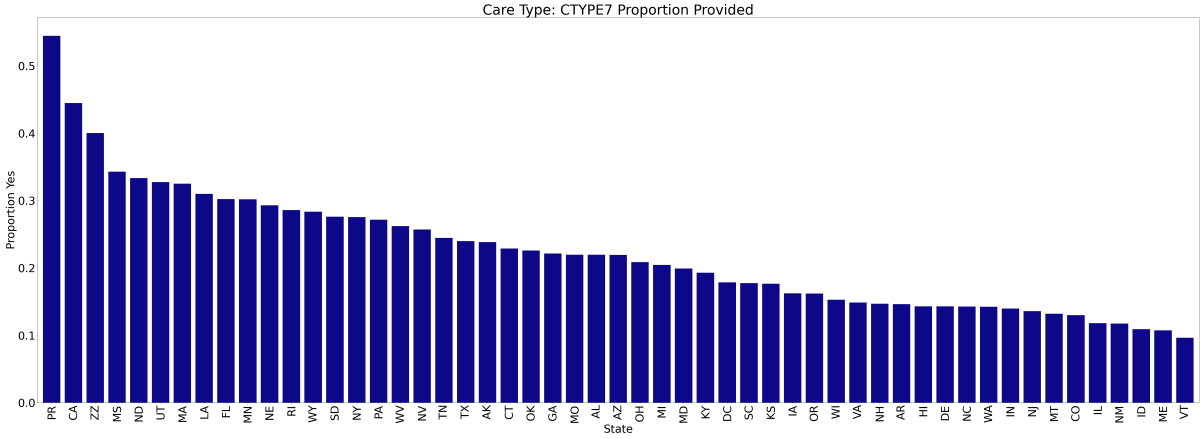


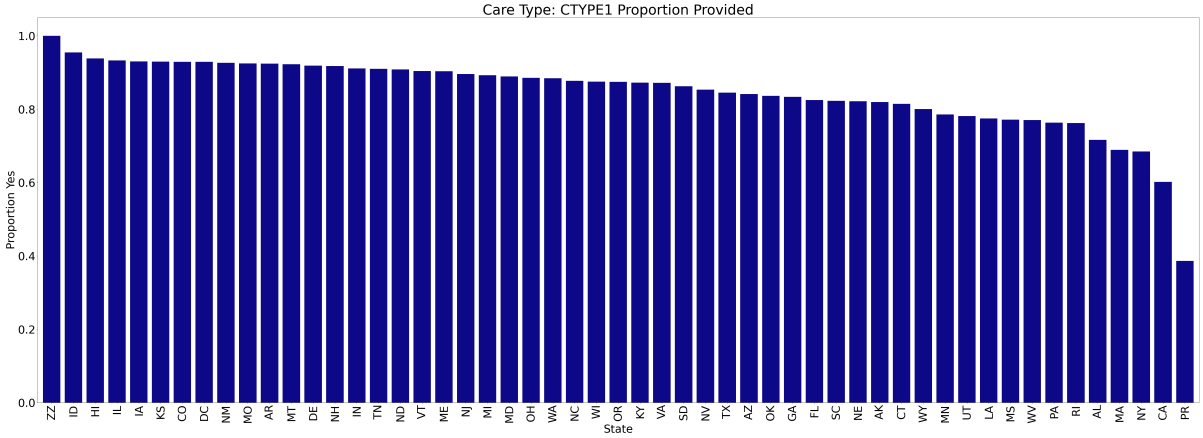
In [28]:
```
ak_sample_df = simple_samhsa_df[simple_samhsa_df['STATE'] == 'AK']
ak_sample_df['CTYPE7'].value_counts()
```

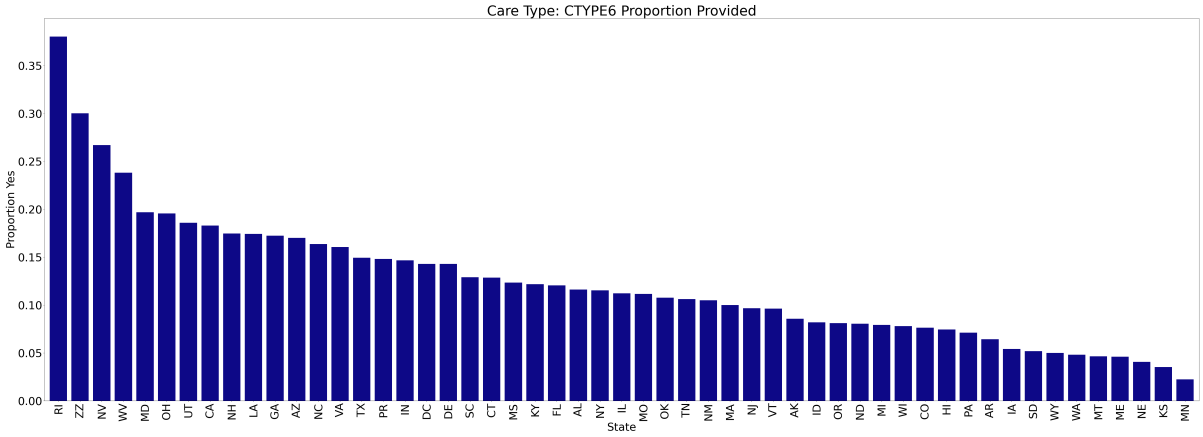
Out[28]:
```
0    1840
1     575
Name: CTYPE7, dtype: int64
```

In [29]:
```
"""
Cell Description:
This plots multiple graphs of the proportion of yes and no respones for each
which is sorted by order of most yes responses, grouped by state.
"""
ctype_cols = [feature for feature in simple_samhsa_df.columns.values if 'CTY
for ctype_col in ctype_cols:
  ctype_series = simple_samhsa_df.groupby('STATE').mean()[ctype_col].sort_va
  ctype_series.plot(kind='bar', xlabel='State', ylabel='Proportion Yes', rot
  plt.title("Care Type: {care_type} Proportion Provided".format(care_type=ct
  plt.xlabel("State", fontsize=40)
  plt.ylabel("Proportion Yes", fontsize=40)
  plt.show()
```


Care Type: CTYPEML Proportion Provided
Proportion Yes
State
NY CT MA NH DE RI VT MO WV SC MD PR OH PA VA ME IN NJ NC LA AZ UT FL NV OR AL WI DC TN WY KY WA NM CO IL TX GA MI CA AK ID ZZ MS OK AR MT IA MN KS ND SD NE HI


Care Type: CTYPE7 Proportion Provided
Proportion Yes
State
PR CA ZZ MS ND UT MA LA FL MN NE RI WY SD NY PA WV NV TN TX AK CT OK GA MO AL AZ OH MI MD KY DC SC KS IA OR WI VA NH AR HI DE NC WA IN NJ MT CO IL NM ID ME VT


Care Type: CTYPE1 Proportion Provided
Proportion Yes
State
ZZ ID HI IL IA KS CO DC NM MO AR MT DE NH IN TN ND VT ME NJ MI MD OH WA NC WI OR KY VA SD NV TX AZ OK GA FL SC NE AK CT WY MN UT LA MS WV PA RI AL MA NY CA PR

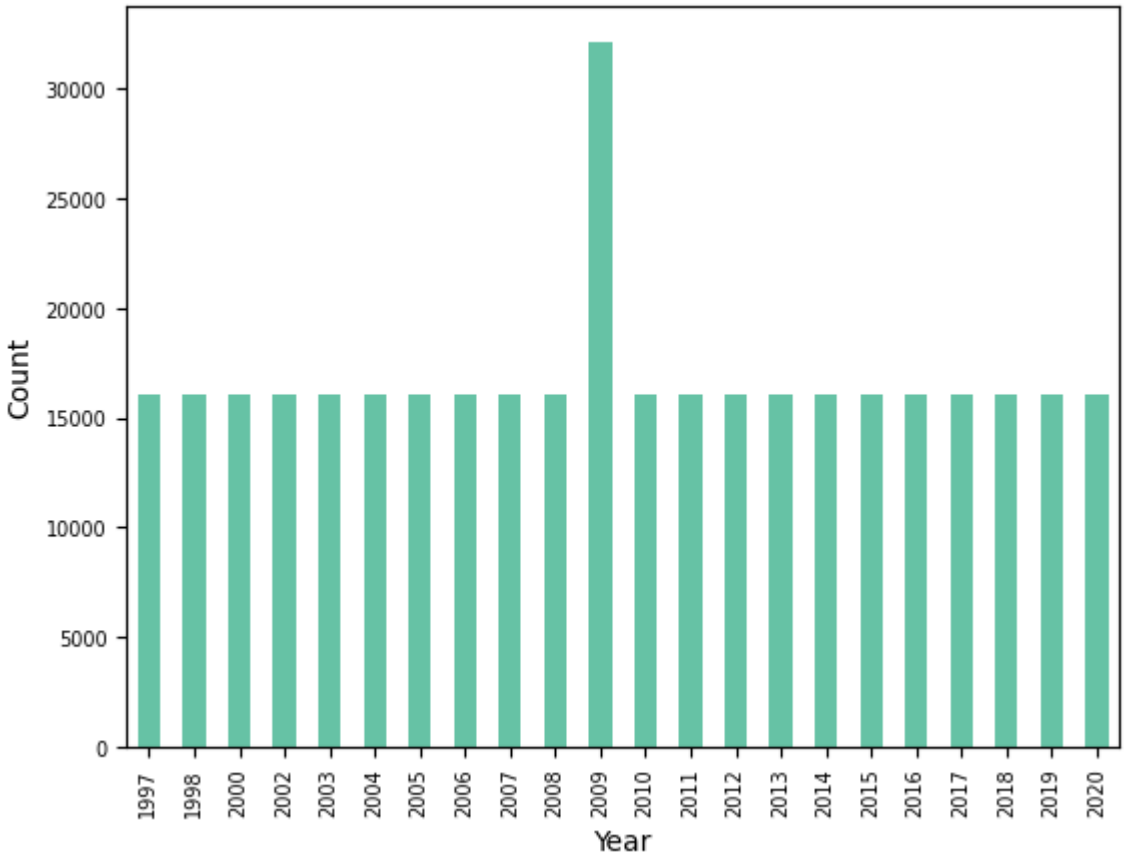

Care Type: CTYPE6 Proportion Provided
Proportion Yes
State
RI ZZ NV WV MD OH UT CA NH LA GA AZ NC VA TX PR IN DC DE SC CT MS KY FL AL NY IL MO OK TN NM MA NJ VT AK ID OR ND MI WI CO HI PA AR IA SD WY WA MT ME NE KS MN

In [29]:

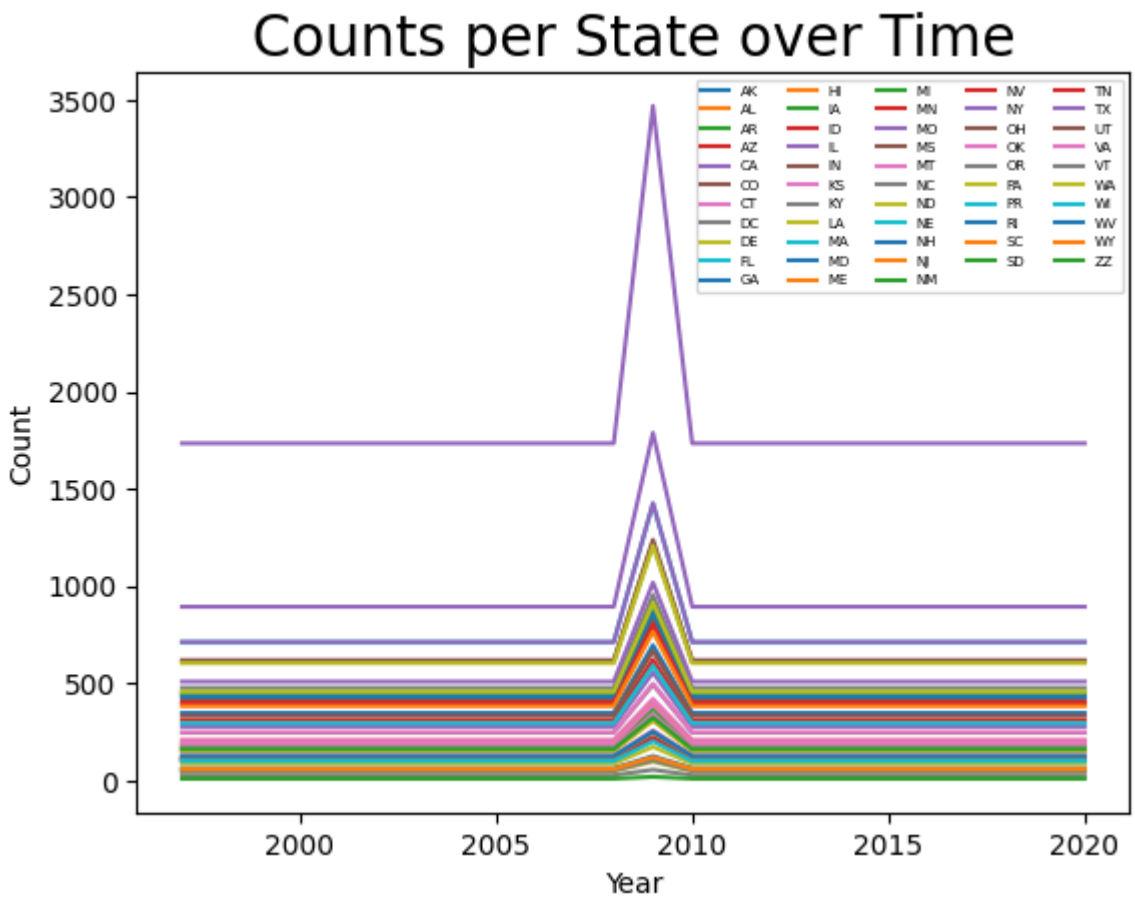
In [30]: """

```
Cell Description:
This graphs the number of records grouped by year
"""
simple_samhsa_df['Year'].value_counts().sort_index().plot(kind='bar', xlabel
```

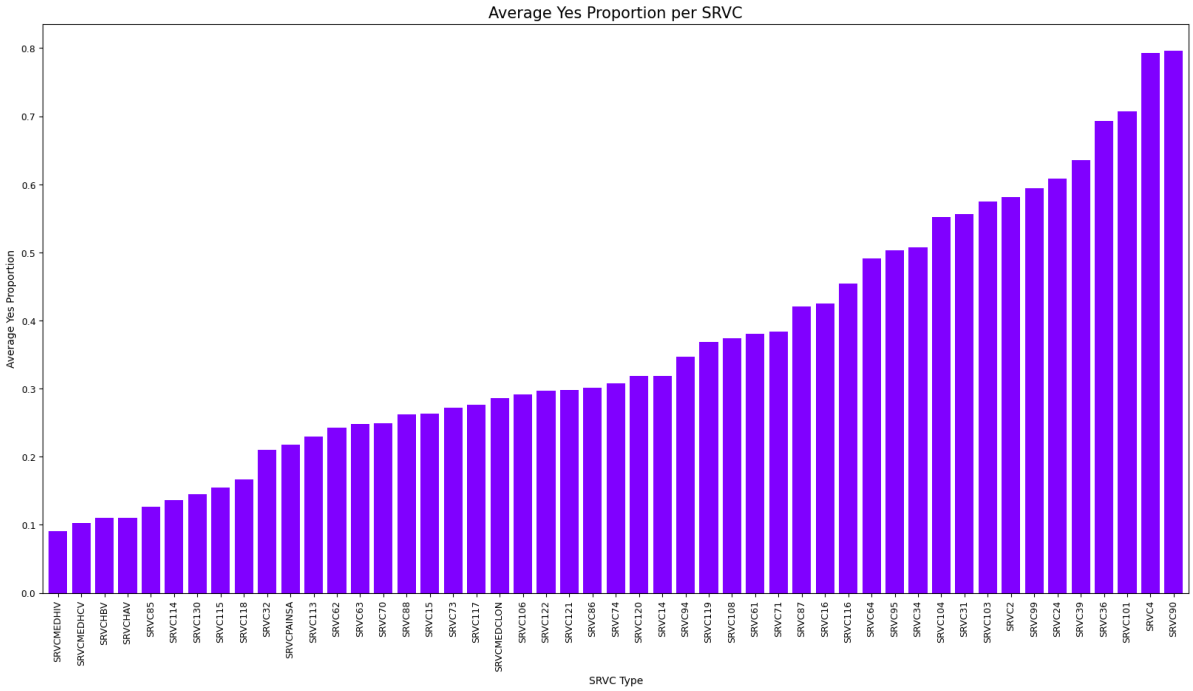
Out[30]: <Axes: xlabel='Year', ylabel='Count'>



In [31]:
```
"""
Cell Description:
This graphs the same as above, but now separated/grouped by state over time.
"""
for state in simple_samhsa_df['STATE'].unique():
  state_df = simple_samhsa_df[simple_samhsa_df['STATE'] == state]
  state_srs = state_df.groupby('Year').count()['CASEID']
  plt.plot(state_srs, label=state)
  #state_srs.plot(kind='line', xlabel='Year', ylabel='Count', rot=90, fontsi
plt.title("Counts per State over Time", fontsize=20)
plt.xlabel("Year", fontsize=10)
plt.ylabel("Count", fontsize=10)
plt.legend(loc="best", fontsize=5, ncols=5)
plt.show()
```

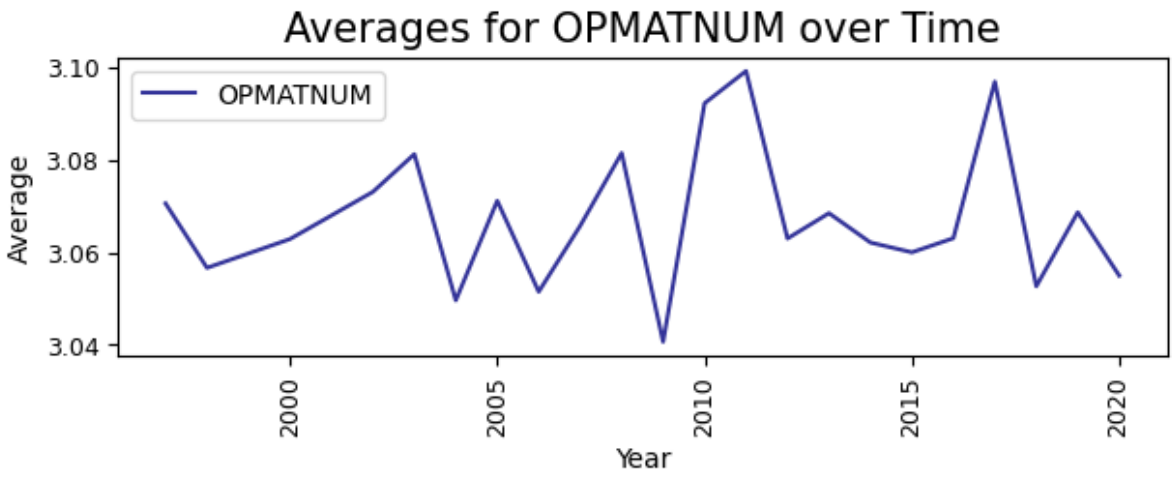


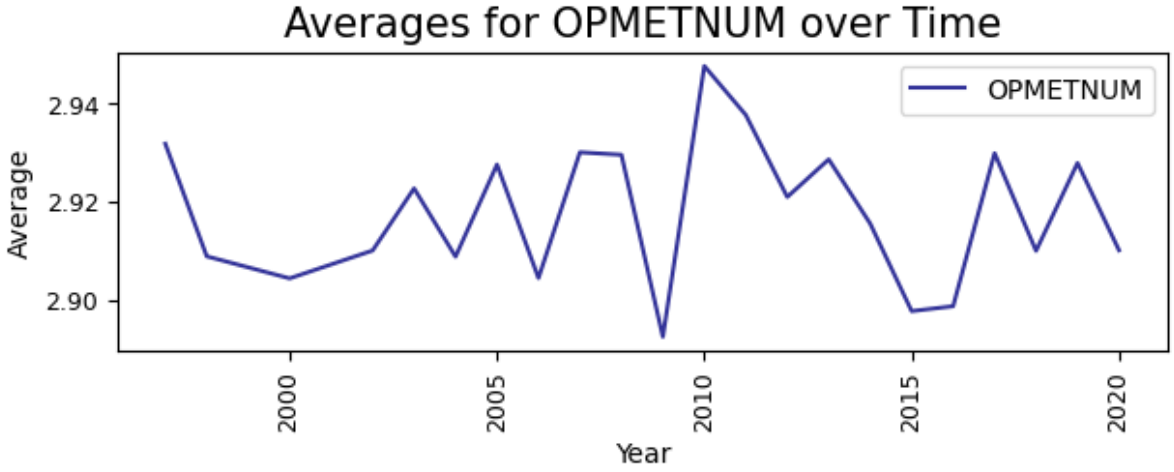
In [32]:
```
"""
Cell Description:
This plots the average proportions of yes repsonses per SRVC feature. The av
is over all years in the dataset.
"""
def plot_year_vs_srvc():
  srvc_boolean_cols = [feature for feature in simple_samhsa_df.columns.value
  srvc_df = simple_samhsa_df[['Year'] + srvc_boolean_cols]
  srvc_srs = srvc_df.groupby('Year').mean().mean()
  srvc_srs.sort_values(ascending=True, inplace=True)
  srvc_srs.plot(kind='bar', xlabel='SRVC', ylabel='Average Yes Proportion',
  plt.title("Average Yes Proportion per SRVC", fontsize=15)
  plt.xlabel("SRVC Type", fontsize=10)
  plt.ylabel("Average Yes Proportion", fontsize=10)
  plt.show()

plot_year_vs_srvc()
```

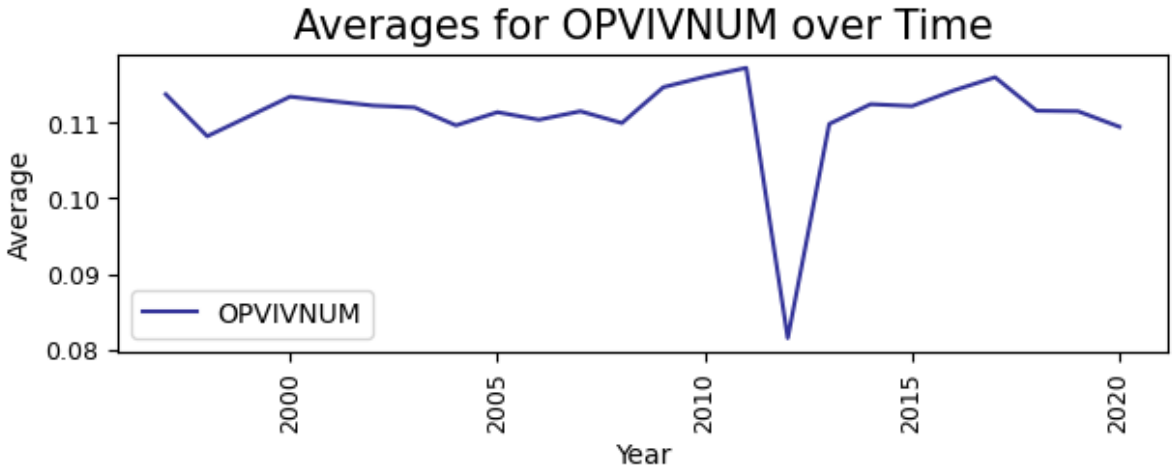


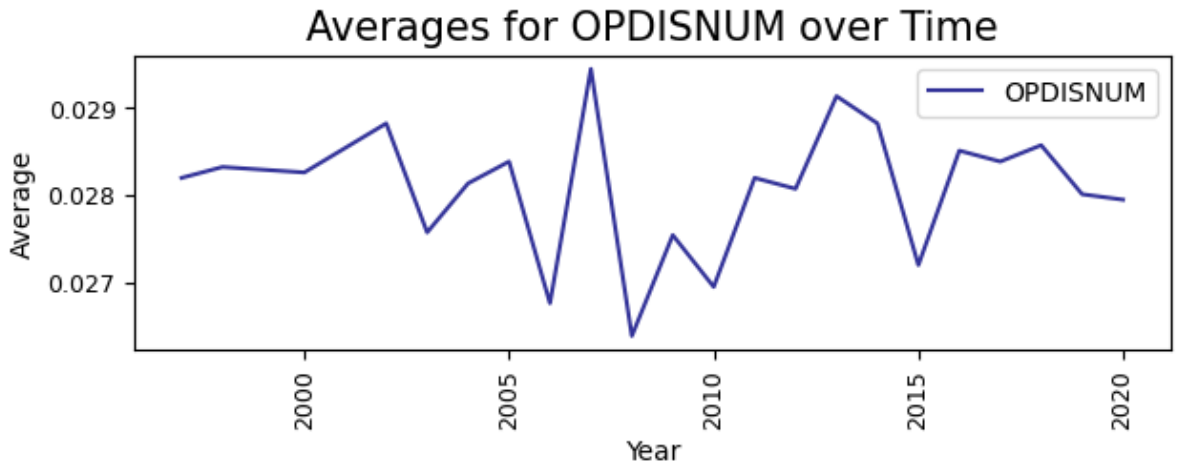
In [33]:

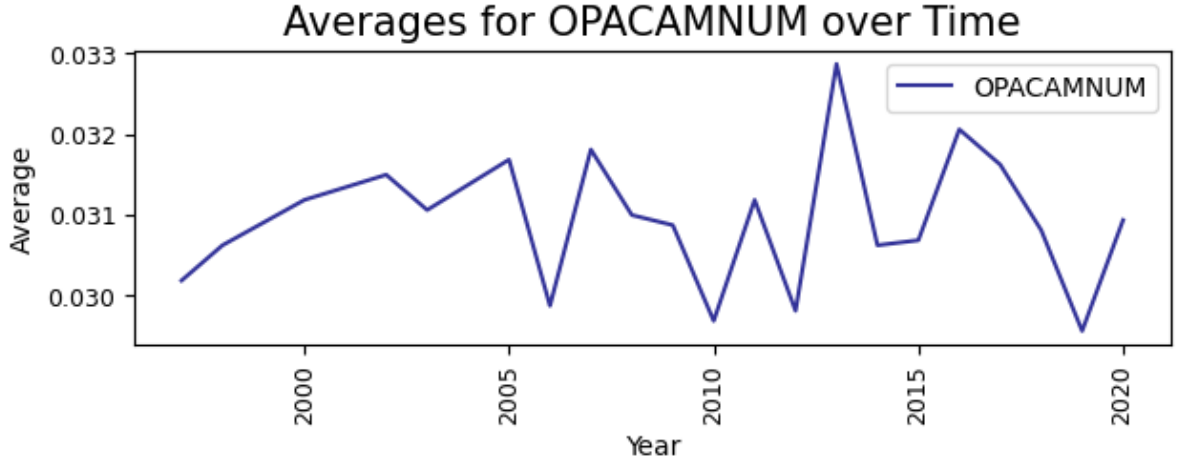
```
"""
Important Note: Just like in a previous cell, some of the features being gra
below have already been log transformed to bring to normal distribution!!!!!
These are the features that have been transformed:
[A_PCT  B_PCT   D_PCT   OPBUPNUM        OPVIVNUM        OPDISNUM        OPNA

These are graphs of the OPMATNUM ordinal features over time, some of which
have alredy been log-transformed as descibed above.
"""
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeigh

outpatient_freq_cols = [feature for feature in simple_samhsa_df.columns.valu
for out_col in outpatient_freq_cols:
  year_srs = simple_samhsa_df[['Year', out_col]].groupby("Year").mean()
  year_srs.plot(kind='line', xlabel='Year', ylabel='(Potentially Log Transfo
  plt.title("Averages for {out_col} over Time".format(out_col=out_col), font
  plt.xlabel("Year", fontsize=10)
  plt.ylabel("Average", fontsize=10)
  plt.show()
```

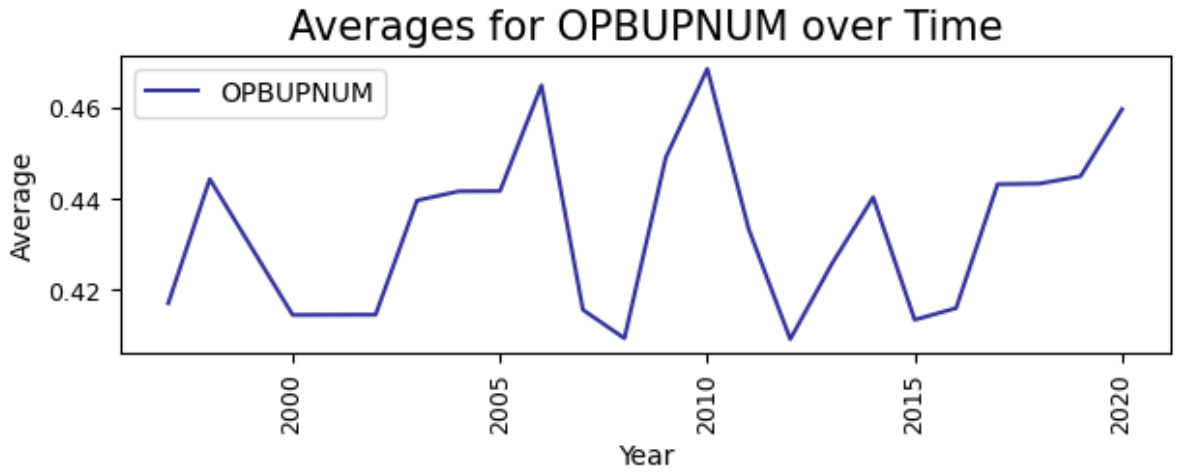

Averages for OPMATNUM over Time
OPMATNUM
Average
3.10
3.08
3.06
3.04
2000
2005
2010
2015
2020
Year

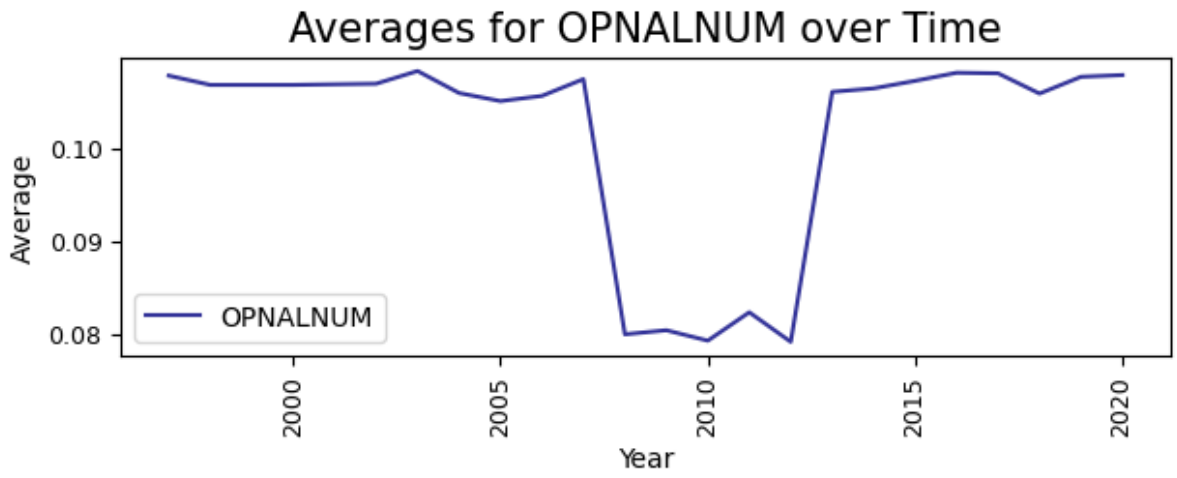

Averages for OPMETNUM over Time
OPMETNUM
Average
2.94
2.92
2.90
2000
2005
2010
2015
2020
Year


Averages for OPVIVNUM over Time
OPVIVNUM
Average
0.11
0.10
0.09
0.08
2000
2005
2010
2015
2020
Year


Averages for OPDISNUM over Time
OPDISNUM
Average
0.029
0.028
0.027
2000
2005
2010
2015
2020
Year

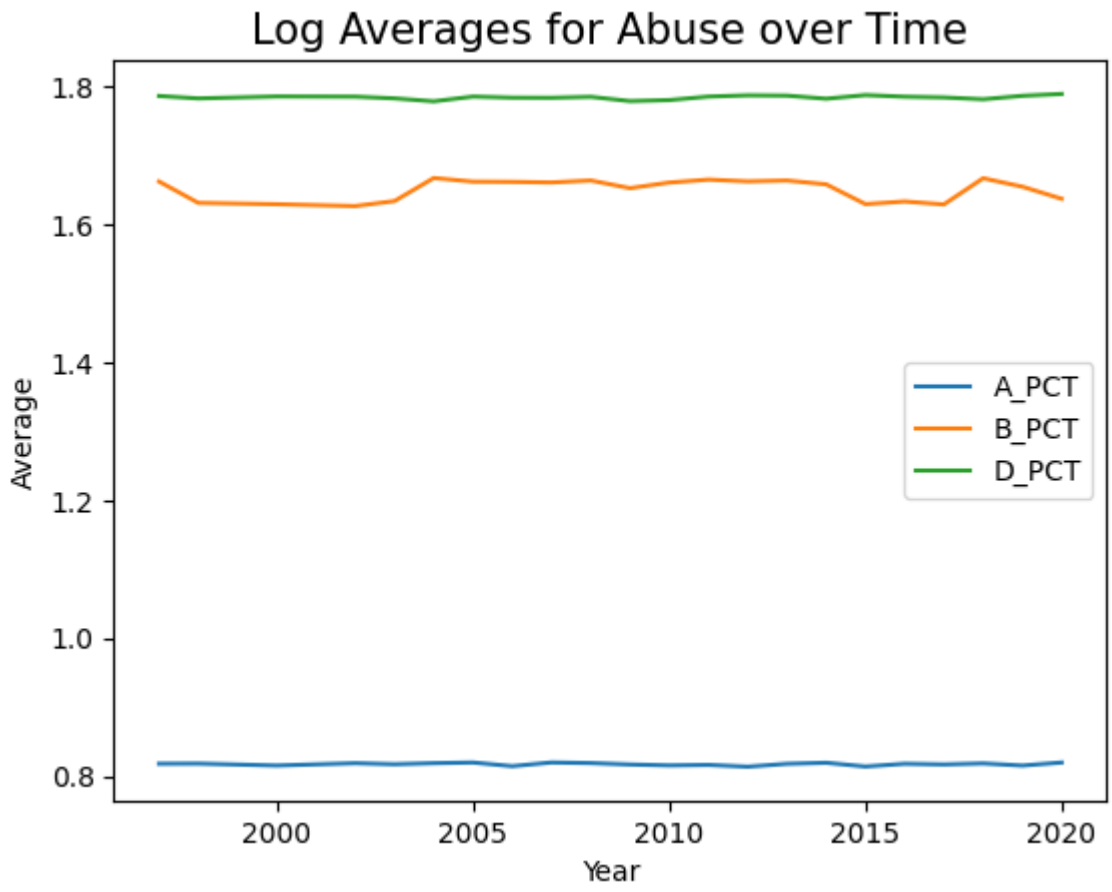

Averages for OPACAMNUM over Time
OPACAMNUM
Average
0.033
0.032
0.031
0.030
2000
2005
2010
2015
2020
Year


Averages for OPBUPNUM over Time
OPBUPNUM
Average
0.46
0.44
0.42
2000
2005
2010
2015
2020
Year



In [34]:

```
"""
Important Note: All of the features being graphed
below have already been log transformed to bring to normal distribution!!!!!

This is a plot of the log average of different types of drug abuse over time
for the relevant columns.
"""
drug_abuse_cols = ["A_PCT", "B_PCT","D_PCT"]
for abuse_col in drug_abuse_cols:
  year_srs = simple_samhsa_df[['Year', abuse_col]].groupby("Year").mean()
  plt.plot(year_srs, label=abuse_col)

plt.title("Log Averages for Abuse over Time", fontsize=15)
plt.xlabel("Year", fontsize=10)
plt.ylabel("Average", fontsize=10)
plt.legend(drug_abuse_cols)
plt.show()
```


Log Averages for Abuse over Time
1.8
1.6
1.4
1.2
1.0
0.8
Average
2000
2005
2010
2015
2020
Year
A_PCT
B_PCT
D_PCT

In [34]: