In [106…

```
import pandas as pd
import time
from datetime import datetime
import seaborn
```

Loading CSV file with race data and converting it into a Pandas dataframe

In [106…

```
rd = pd.read_csv('/Volumes/GoogleDrive/My Drive/PythonFun/TheRutAnalysis/rut_202
```

Exploring data and converting data types

In [107…

```
rd.dtypes
```

Out[107…

```
Place            int64
Name            object
City            object
State           object
Gender          object
Age              int64
Clock Time      object
Chip Time       object
Pace            object
dtype: object
```

In [107…

```
rd.describe
```

Out[107…

```
<bound method NDFrame.describe of      Place                Name            City  
State Gender  Age Clock Time  \
0        1            Cam Smith  Crested Butte    CO      M   25    3:09:00
1        2         Dakota Jones        Bozeman    MT      M   30    3:15:42
2        3         Aaron Robson         Lander    WY      M   34    3:23:08
3        4          Ryan Becker        Boulder    CO      M   27    3:27:18
4        5     Patrick Caldwell         Denver    CO      M   27    3:30:35
..     ...                  ...            ...   ...    ...  ...        ...
447    448         Kelly Meeker        Bozeman    MT      F   40    8:40:03
448    449         Emily Culkin       Missoula    MT      F   32    8:40:41
449    450      Eric Kitzmiller        Bozeman    MT      M   53    8:42:58
450    451         Nicole Marsh        Bozeman    MT      F   28    8:48:05
451    452  Christina Callender       Billings    MT      F   41    8:52:04

    Chip Time      Pace
0     3:09:00  10:51:00
1     3:15:42  11:14:00
2     3:23:08  11:40:00
3     3:27:18  11:54:00
4     3:30:33  12:06:00
..        ...       ...
447   8:33:42  29:30:00
448   8:33:50  29:31:00
449   8:35:48  29:38:00
450   8:41:48  29:58:00
451   8:46:51  30:16:00

[452 rows x 9 columns]>
```

*After reviewing the default data types and example data rows, we'll need to convert the relevant string fields ("Clock Time", "Chip Time" and "Pace") into timedelta objects. Timedeltas are absolute differences in times, expressed in difference units (e.g., hours or minutes).

Regarding our string fields being categorized as object data types for fields such as name, this is due to variable string lengths.

In [107…
```
rd['Clock Time (hrs)'] = pd.to_timedelta(rd['Clock Time'])
rd['Chip Time (hrs)'] = pd.to_timedelta(rd['Chip Time'])
```

*Since the source data for the "Pace" field is formatted the same way as the total duration fields ("Clock Time" & "Chip Time"), we'll need to do some string manipulation before converting it to a timedelta.

In [107…
```
rd['Pace (min/mile)'] = '00:' + rd['Pace'].str[:-3]
rd['Pace (min/mile)'] = pd.to_timedelta(rd['Pace (min/mile)'])
```

In [107…
```
rd.dtypes
```

Out[107…
```
Place                          int64
Name                          object
City                          object
State                         object
Gender                        object
Age                            int64
Clock Time                    object
Chip Time                     object
Pace                          object
Clock Time (hrs)     timedelta64[ns]
Chip Time (hrs)      timedelta64[ns]
Pace (min/mile)      timedelta64[ns]
dtype: object
```

In [107…
```
rd.describe()
```

Out[107…

	Place	Age	Clock Time (hrs)	Chip Time (hrs)	Pace (min/mile)
count	452.000000	452.000000	452	452	452
mean	226.500000	36.933628	0 days 06:24:56.159292035	0 days 06:21:12.825221238	0 days 00:21:53.778761061
std	130.625419	9.075198	0 days 01:11:32.058724470	0 days 01:09:54.229937643	0 days 00:04:00.900729736
min	1.000000	17.000000	0 days 03:09:00	0 days 03:09:00	0 days 00:10:51
25%	113.750000	30.000000	0 days 05:38:33.500000	0 days 05:35:30.750000	0 days 00:19:16
50%	226.500000	36.000000	0 days 06:25:53.500000	0 days 06:20:44.500000	0 days 00:21:52
75%	339.250000	42.000000	0 days 07:19:29.250000	0 days 07:17:37.750000	0 days 00:25:08.250000
max	452.000000	68.000000	0 days 08:52:04	0 days 08:46:51	0 days 00:30:16

*Above, we're now able to see summary statistics about the time duration fields. This serves as a good jumping off point for analysis.

However, to make the timedelta fields more readible and easier to visualize, let's convert them to floats in hour units.

In [107…

```
rd['Clock Time (hrs)'] = (rd['Clock Time (hrs)'].astype('timedelta64[m]') / 60).
rd['Chip Time (hrs)'] = (rd['Chip Time (hrs)'].astype('timedelta64[m]') / 60).as
rd['Pace (min/mile)'] = (rd['Pace (min/mile)'].astype('timedelta64[s]') / 60).as
```

In [107…

```
rd.describe()
```

Out[107…

	Place	Age	Clock Time (hrs)	Chip Time (hrs)	Pace (min/mile)
count	452.000000	452.000000	452.000000	452.000000	452.000000
mean	226.500000	36.933628	6.407448	6.345133	21.896313
std	130.625419	9.075198	1.192295	1.164882	4.015012
min	1.000000	17.000000	3.150000	3.150000	10.850000
25%	113.750000	30.000000	5.629167	5.583333	19.266667
50%	226.500000	36.000000	6.425000	6.333333	21.866667
75%	339.250000	42.000000	7.316667	7.283333	25.137500
max	452.000000	68.000000	8.866667	8.766667	30.266667

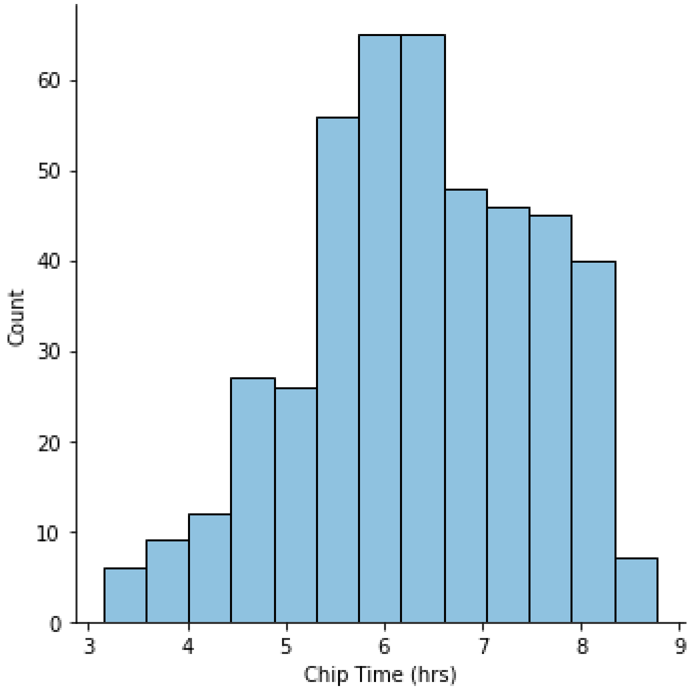
*Finally, here are few plots exploring Chip Time and Pace by gender, age and state.

In [111…

```
seaborn.displot(data=rd['Chip Time (hrs)'], x=rd['Chip Time (hrs)'])
```

Out[111…

```
<seaborn.axisgrid.FacetGrid at 0x7f95f22aa790>
```

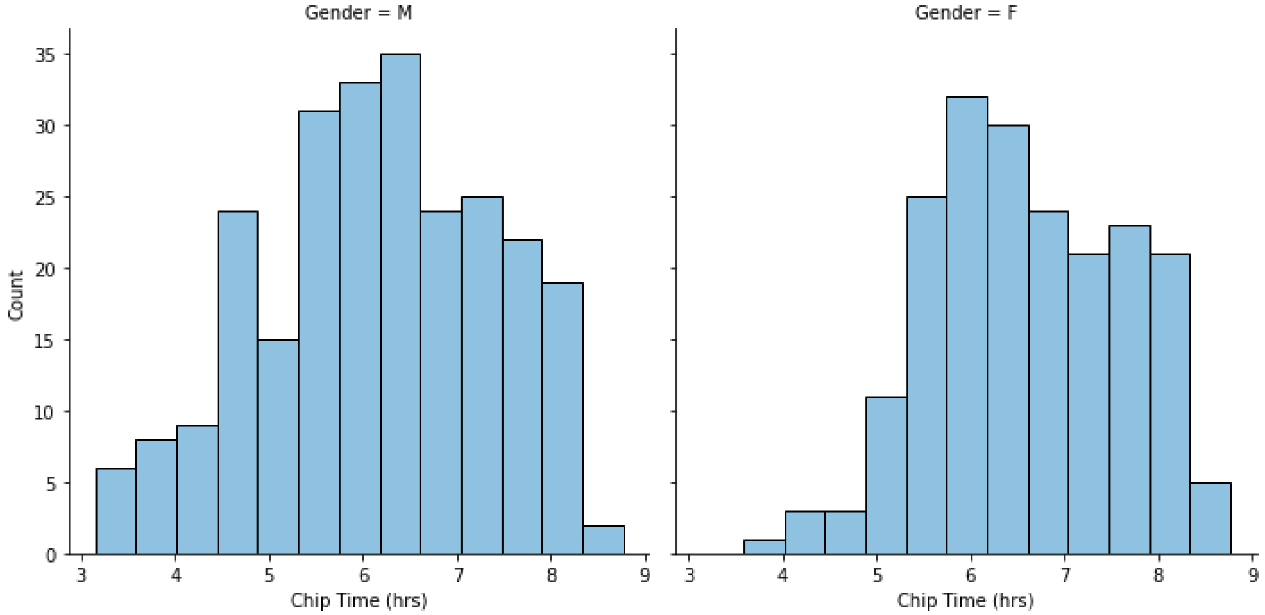


In [107…

```
seaborn.displot(data=rd, x=rd['Chip Time (hrs)'], col='Gender')
```

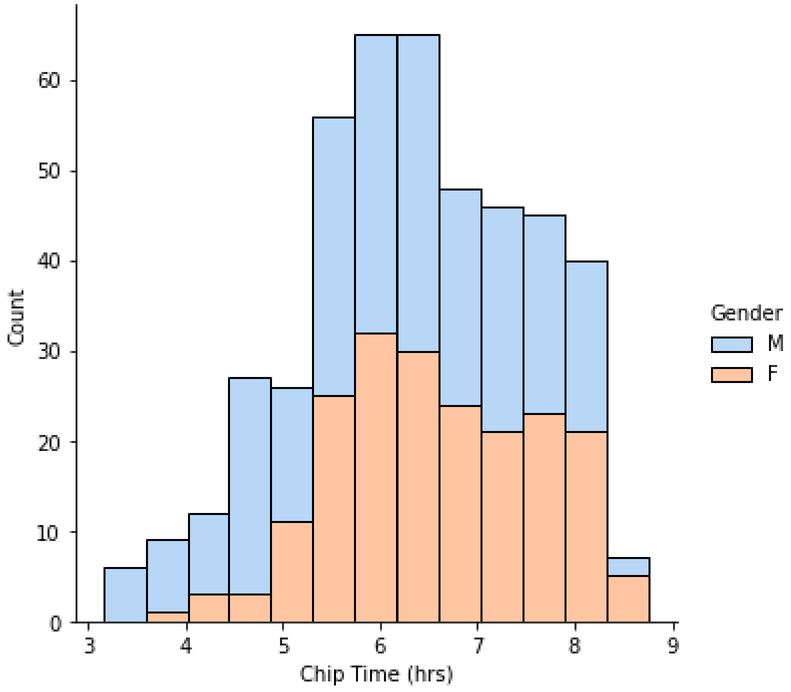
Out[107…

```
<seaborn.axisgrid.FacetGrid at 0x7f96297d2970>
```



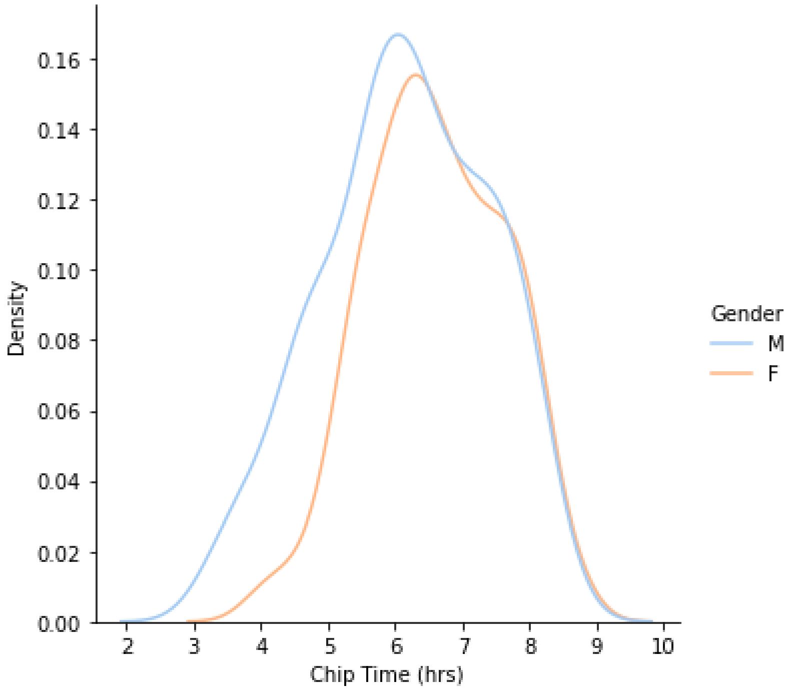
```
In [108…  seaborn.displot(data=rd, x='Chip Time (hrs)', hue='Gender', palette='pastel', mu
```

```
Out[108…  <seaborn.axisgrid.FacetGrid at 0x7f962a946eb0>
```



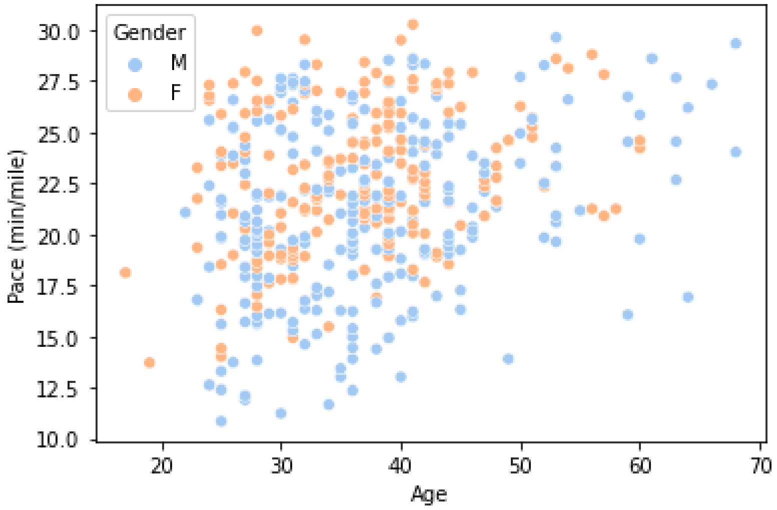
```
In [110…  seaborn.displot(data=rd, x='Chip Time (hrs)', hue='Gender', palette='pastel', ki
```

```
Out[110…  <seaborn.axisgrid.FacetGrid at 0x7f95f80d59a0>
```



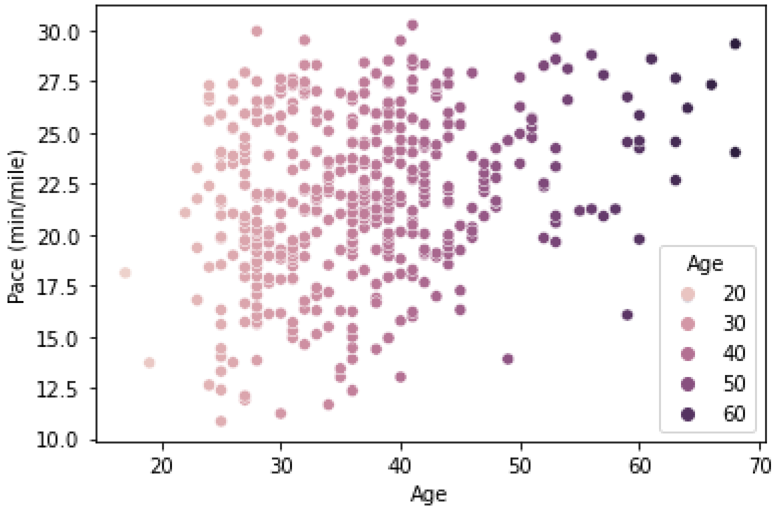
```
In [110…  seaborn.scatterplot(data=rd, x='Age', y='Pace (min/mile)', hue='Gender', palette
```

```
Out[110…  <AxesSubplot:xlabel='Age', ylabel='Pace (min/mile)'>
```



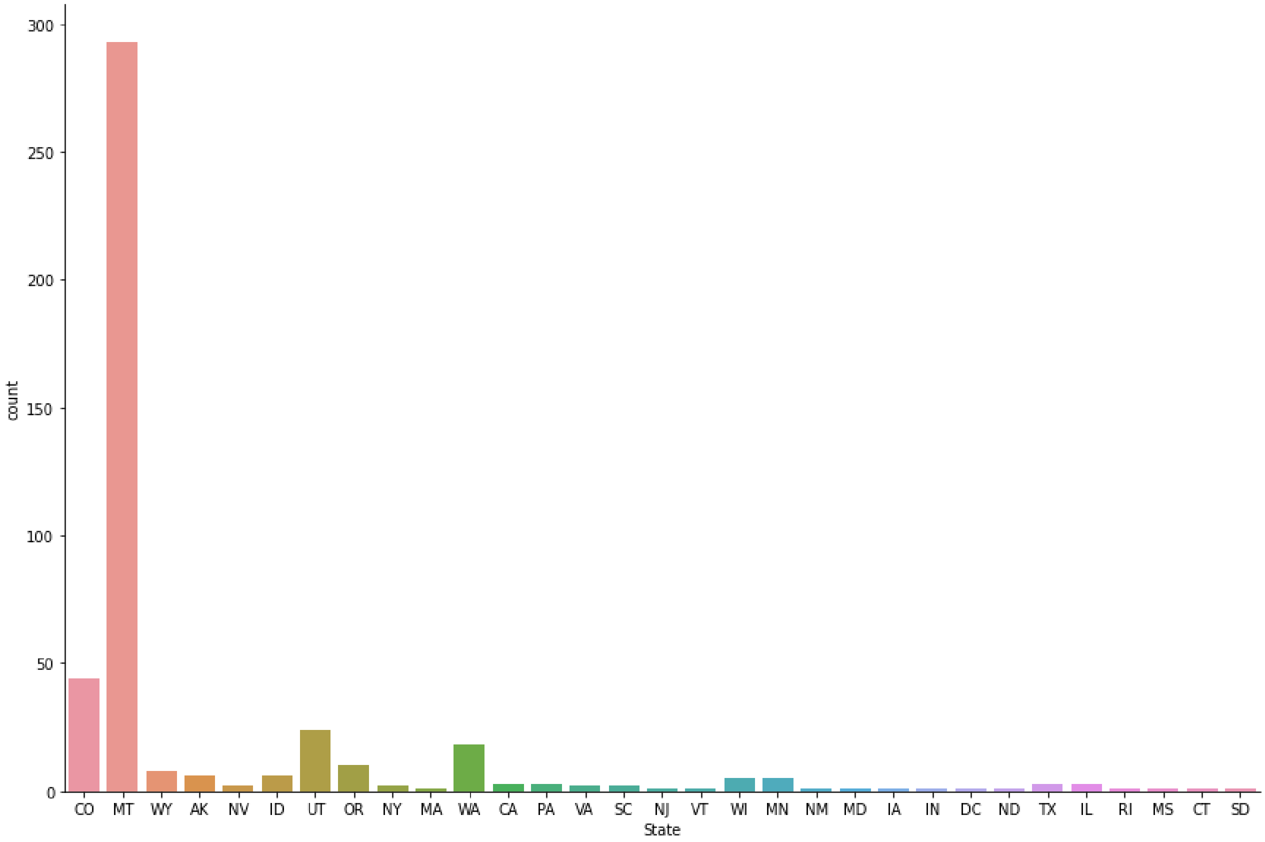
```
In [110…  seaborn.scatterplot(data=rd, x='Age', y='Pace (min/mile)', hue='Age')
```

```
Out[110…  <AxesSubplot:xlabel='Age', ylabel='Pace (min/mile)'>
```



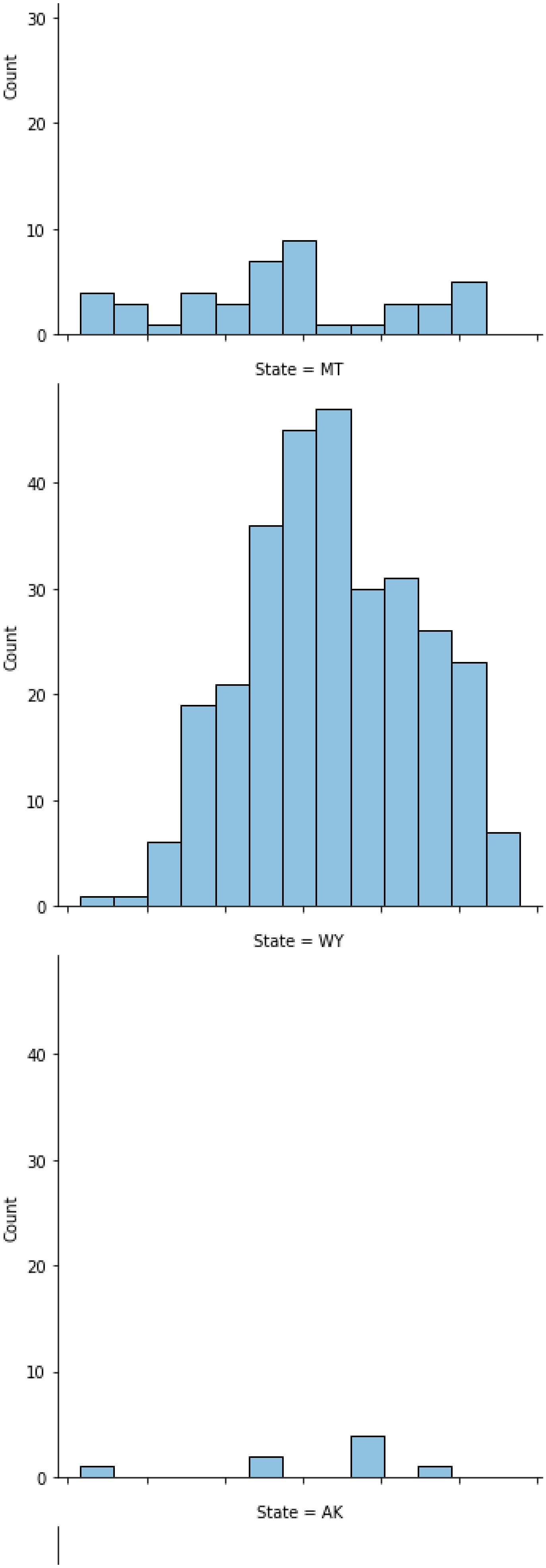
```
In [111…  seaborn.catplot(data=rd, x='State', kind='count', height=8, aspect=1.5 )
```

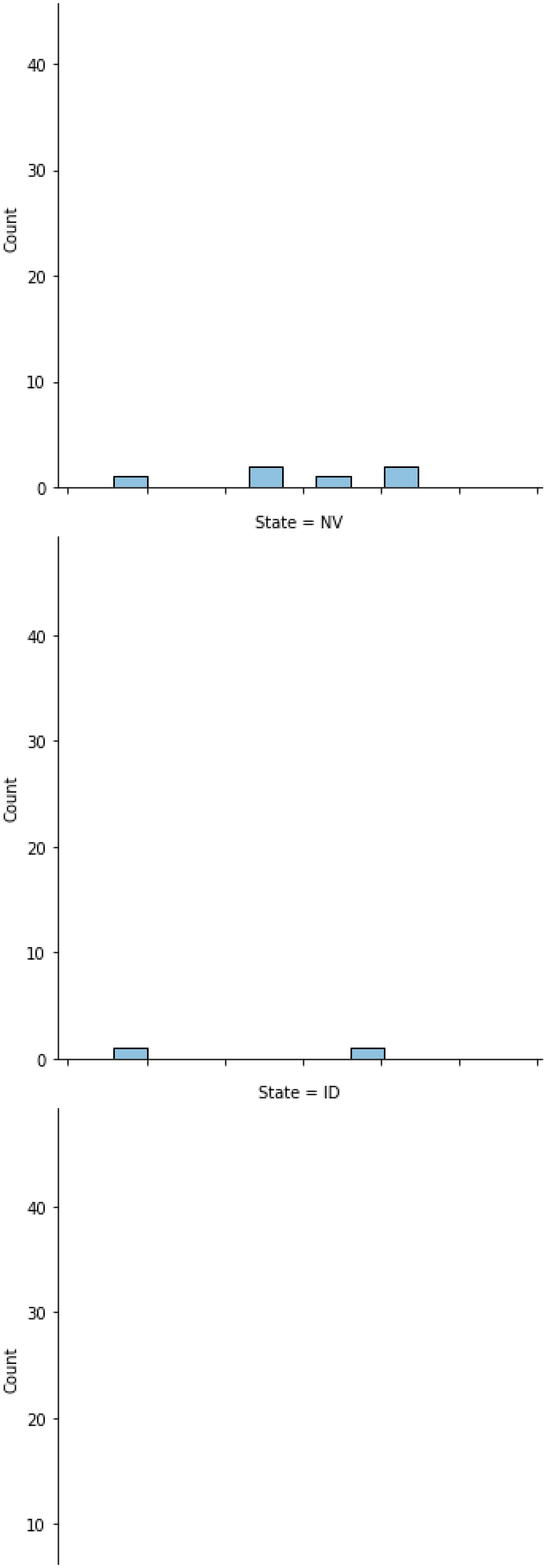
```
Out[111…  <seaborn.axisgrid.FacetGrid at 0x7f95f21d0ee0>
```

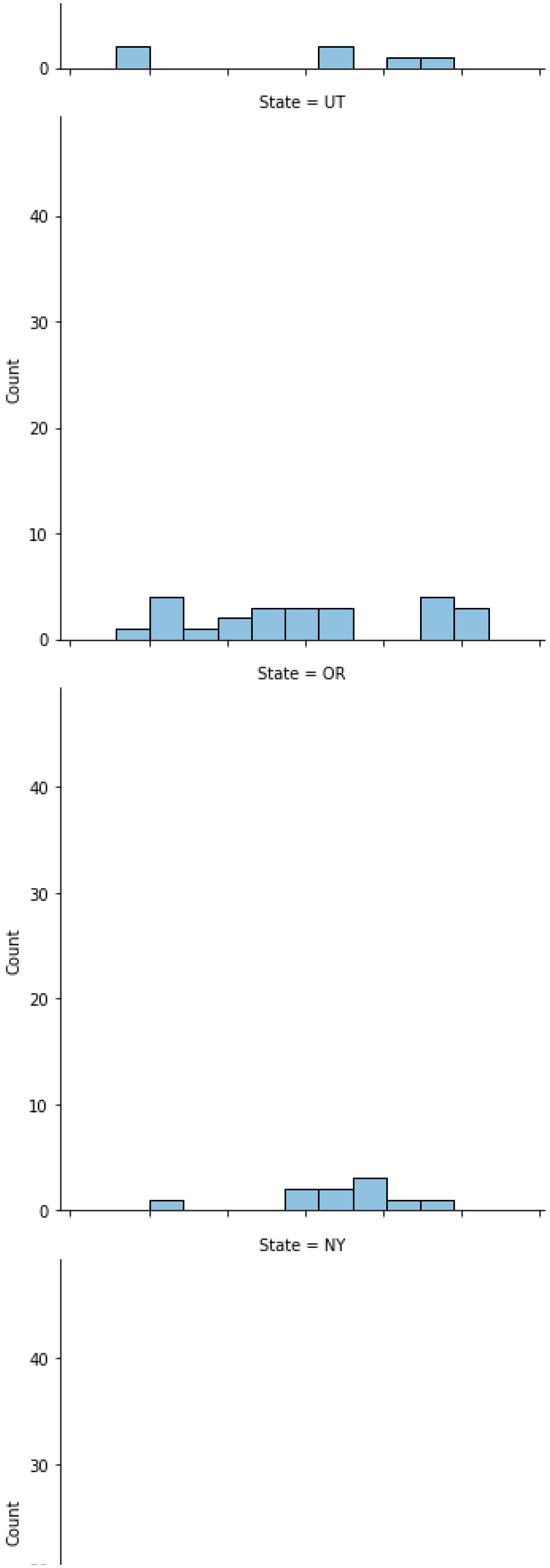


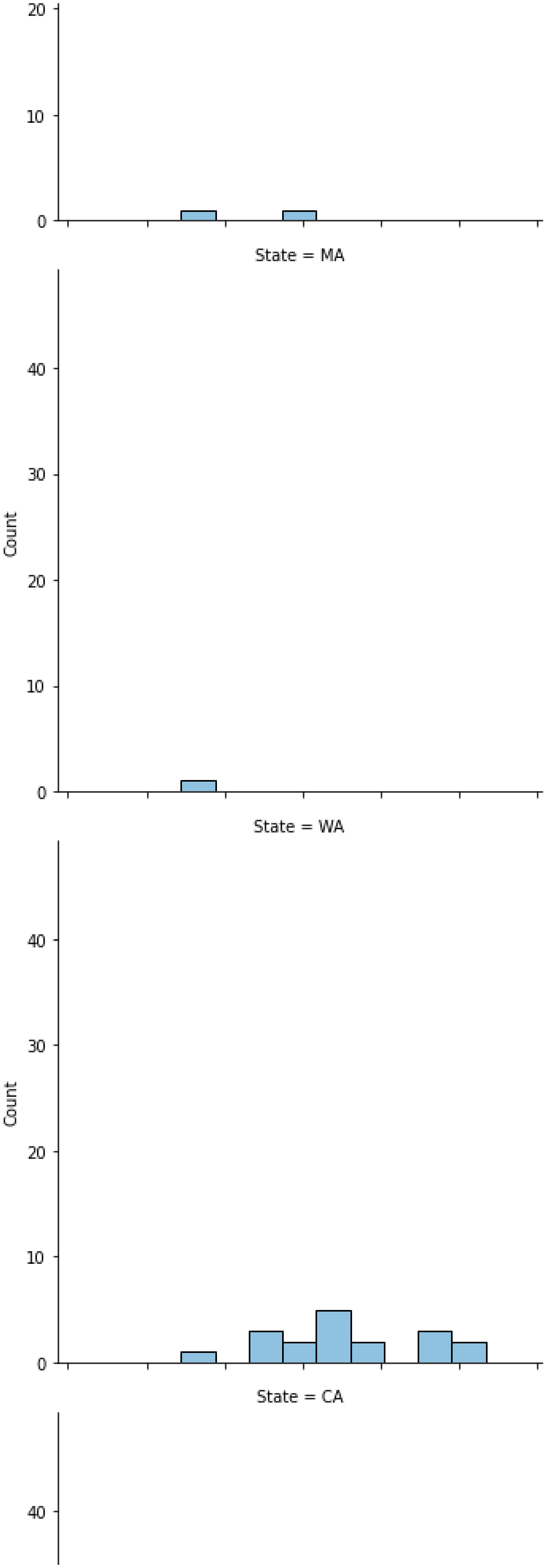
```
In [110…  seaborn.displot(data=rd, x='Chip Time (hrs)', row='State', palette='pastel')
```

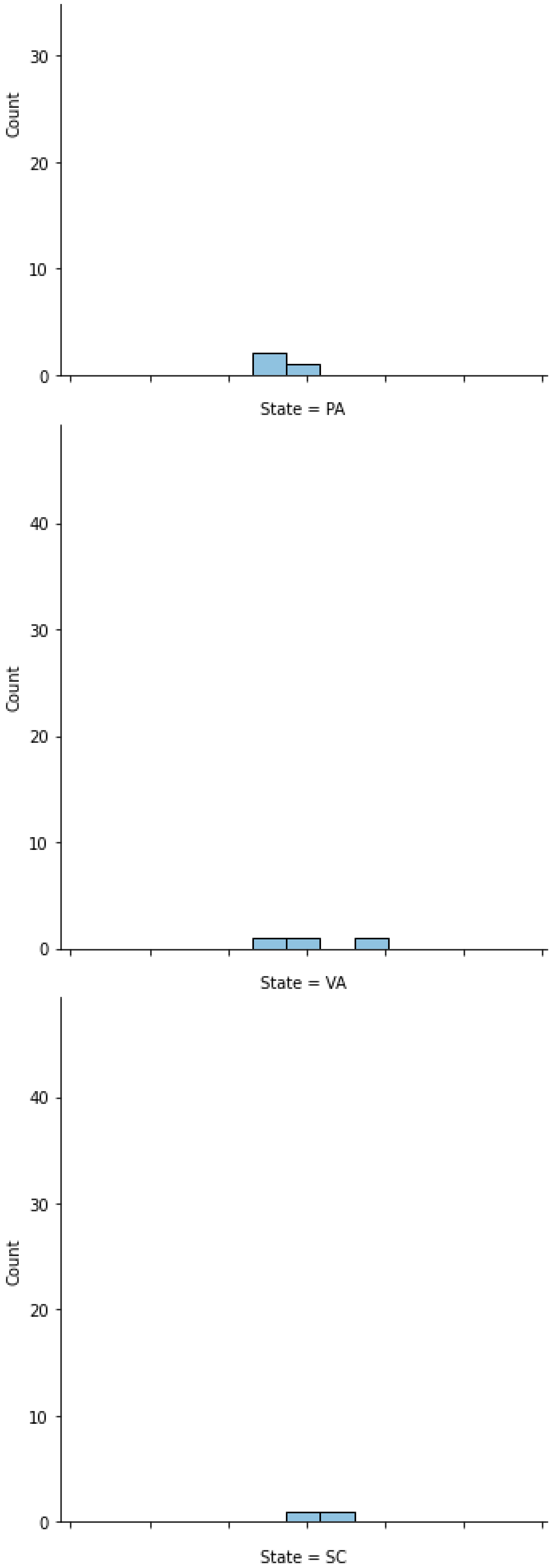
```
Out[110…  <seaborn.axisgrid.FacetGrid at 0x7f95f756d5b0>
```

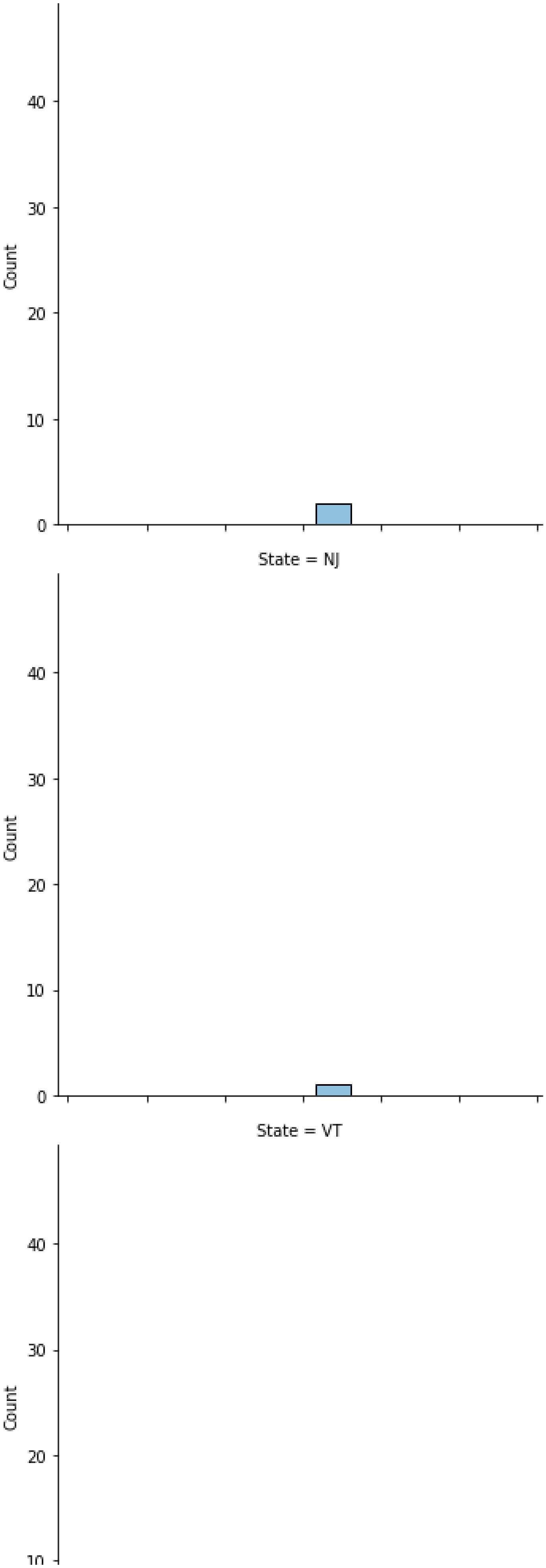

Count
30
20
10
0
State = MT
Count
40
30
20
10
0
State = WY
Count
40
30
20
10
0
State = AK

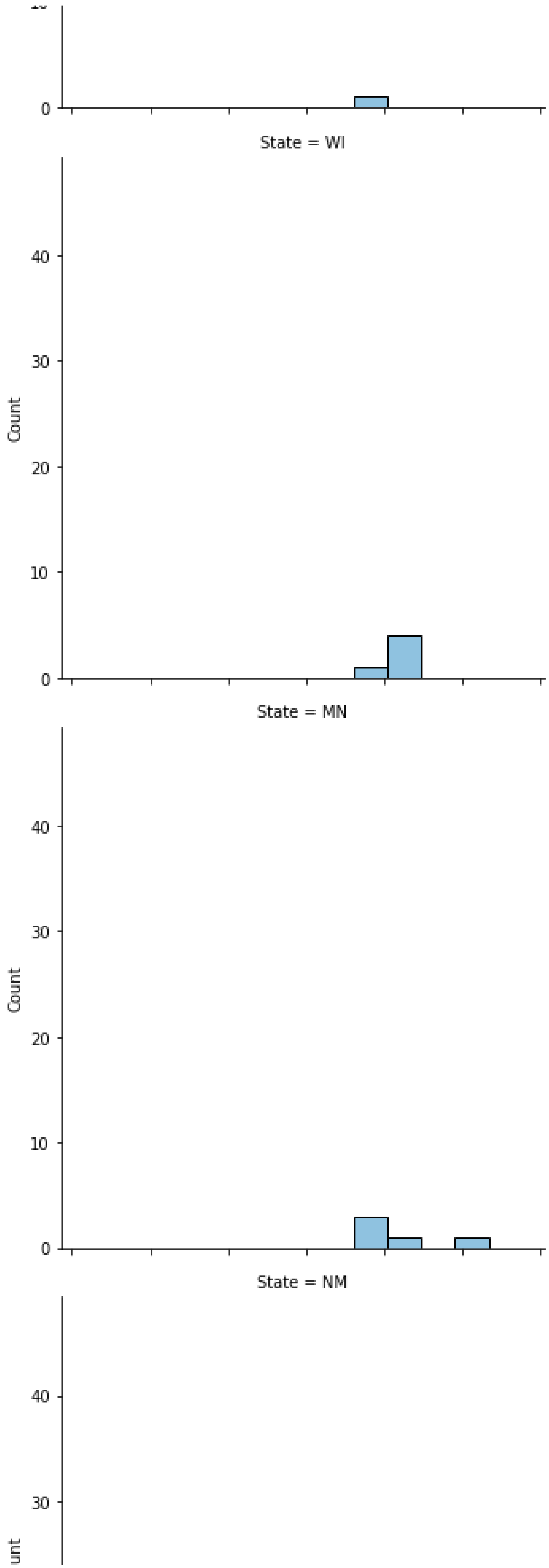

40
30
Count
20
10
0
State = NV
40
30
Count
20
10
0
State = ID
40
30
Count
20
10

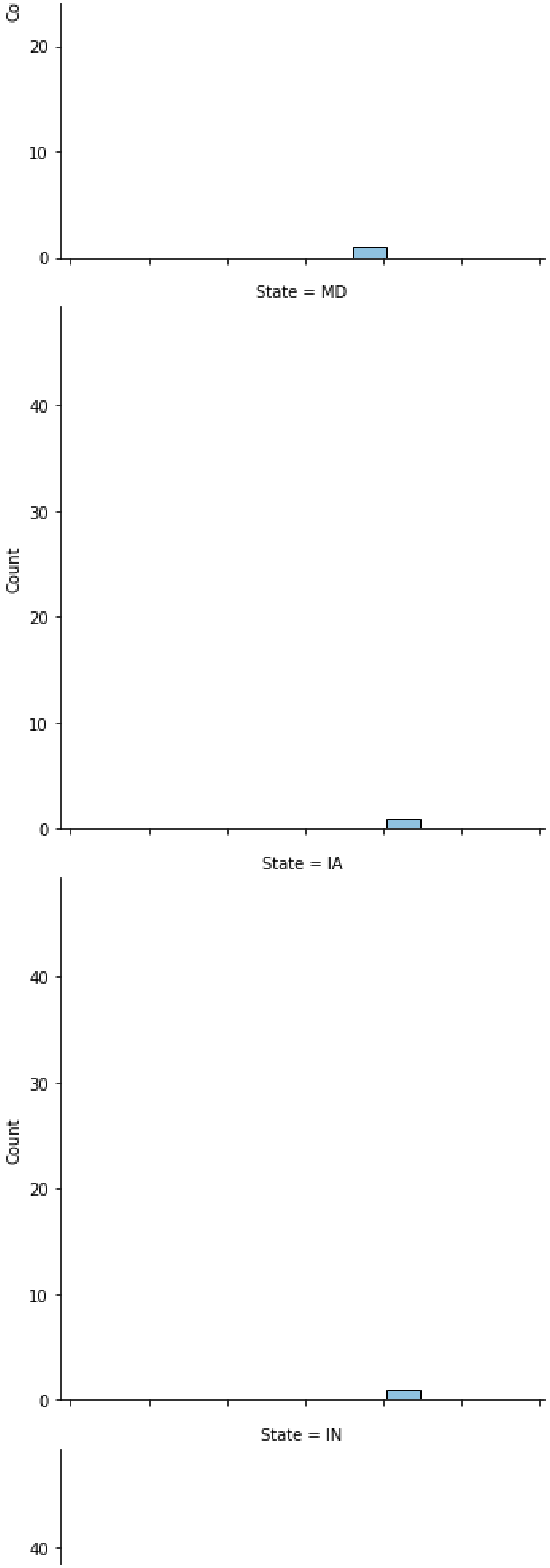

0
State = UT
40
30
20
10
0
Count
State = OR
40
30
20
10
0
Count
State = NY
40
30
Count

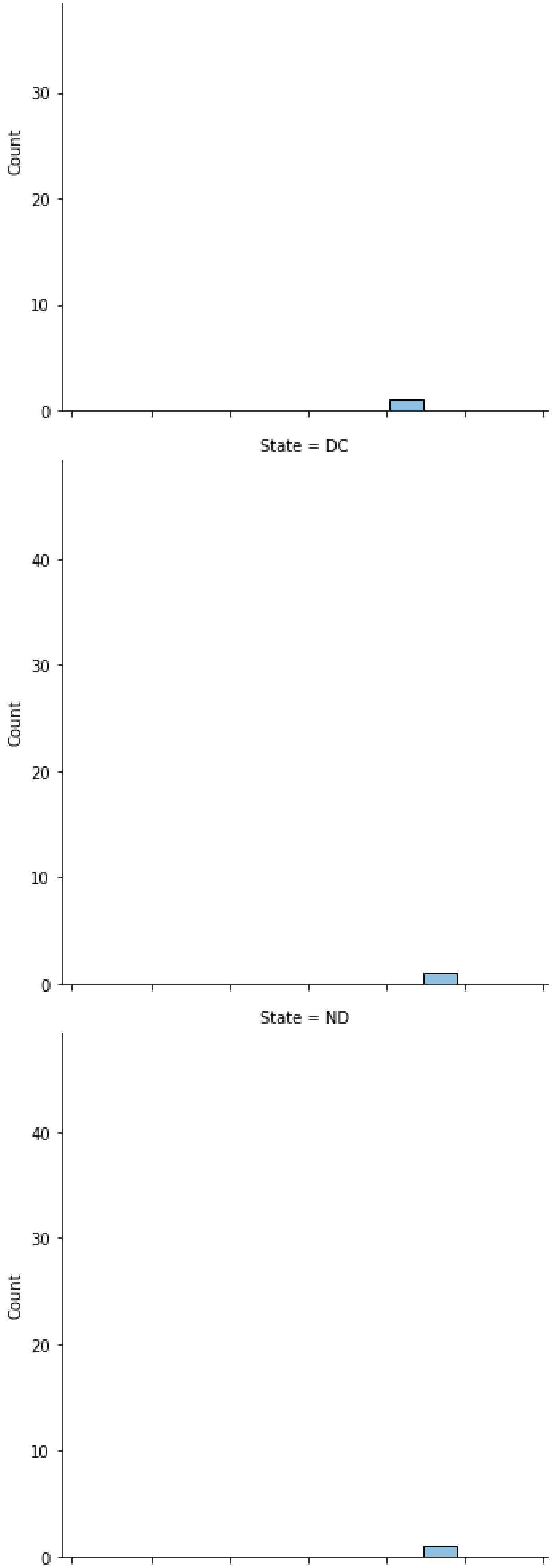

20
10
0
State = MA
40
30
Count
20
10
0
State = WA
40
30
Count
20
10
0
State = CA
40

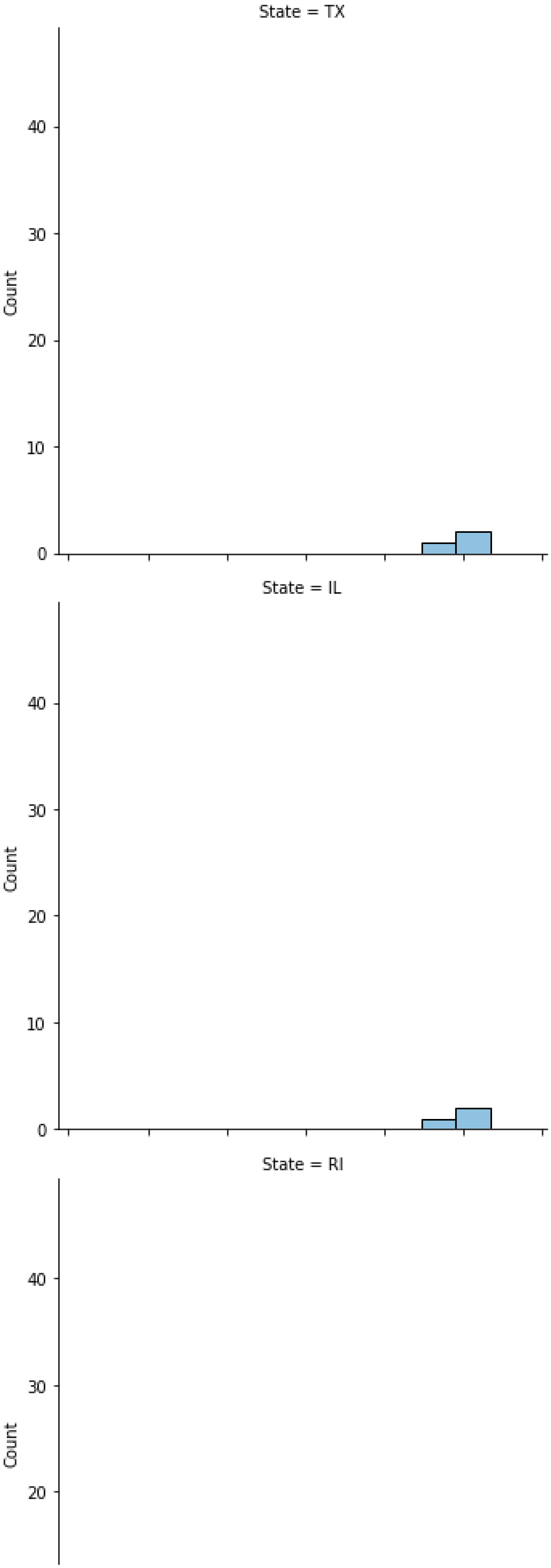

30
Count
20
10
0
State = PA
40
30
Count
20
10
0
State = VA
40
30
Count
20
10
0
State = SC

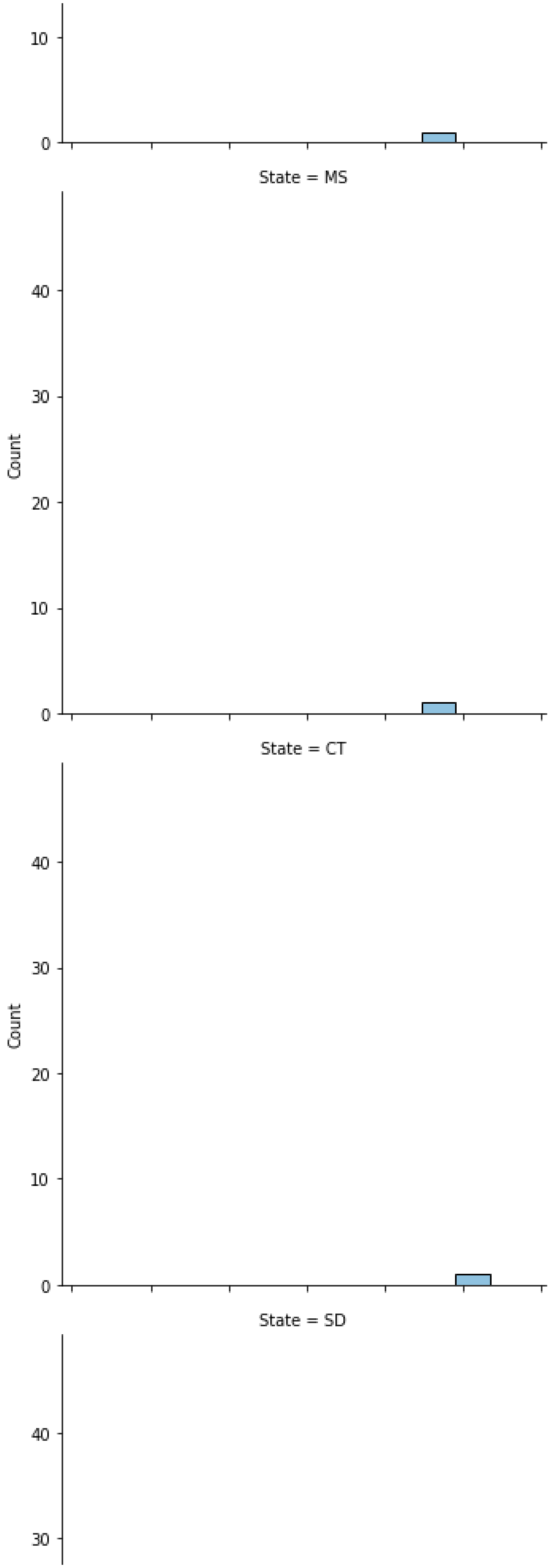

40
30
Count
20
10
0
State = NJ
40
30
Count
20
10
0
State = VT
40
30
Count
20
10

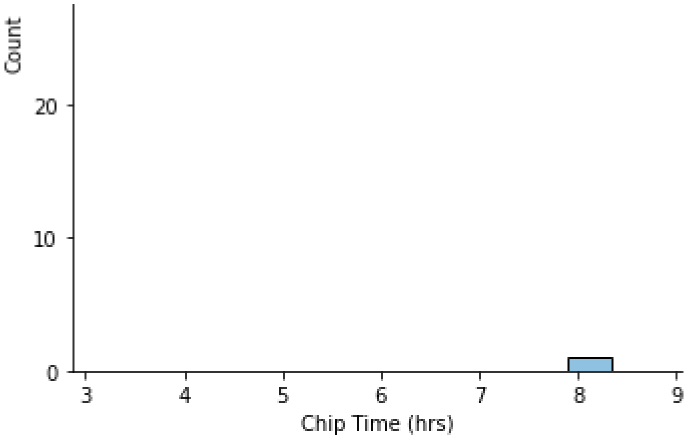

0
State = WI
Count
40
30
20
10
0
State = MN
Count
40
30
20
10
0
State = NM
unt
40
30


Co
20
10
0
State = MD
40
30
Count
20
10
0
State = IA
40
30
Count
20
10
0
State = IN
40


Count
30
20
10
0
State = DC
Count
40
30
20
10
0
State = ND
Count
40
30
20
10
0


State = TX
Count
0
10
20
30
40
State = IL
Count
0
10
20
30
40
State = RI
Count
20
30
40


10
0
State = MS
40
30
20
10
0
Count
State = CT
40
30
20
10
0
Count
State = SD
40
30


Count
20
10
0
3
4
5
6
7
8
9
Chip Time (hrs)