In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import datetime as dt
import calendar
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:
```
df1=pd.read_csv(r"C:\Users\user\Downloads\Unemployment in India.csv")
df2=pd.read_csv(r"C:\Users\user\Downloads\Unemployment_Rate_upto_11_2020.csv")
```

In [3]:
```
df1.head()
```

Out[3]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	Andhra Pradesh	31-05-2019	Monthly	3.65	11999139.0	43.24	Rural
1	Andhra Pradesh	30-06-2019	Monthly	3.05	11755881.0	42.05	Rural
2	Andhra Pradesh	31-07-2019	Monthly	3.75	12086707.0	43.50	Rural
3	Andhra Pradesh	31-08-2019	Monthly	3.32	12285693.0	43.97	Rural
4	Andhra Pradesh	30-09-2019	Monthly	5.17	12256762.0	44.68	Rural

In [4]:
```
df1.columns
```

Out[4]:
```
Index(['Region', ' Date', ' Frequency', ' Estimated Unemployment Rate (%)',
       ' Estimated Employed', ' Estimated Labour Participation Rate (%)',
       'Area'],
      dtype='object')
```

In [5]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   Region                                   740 non-null    object
 1    Date                                    740 non-null    object
 2    Frequency                               740 non-null    object
 3    Estimated Unemployment Rate (%)         740 non-null    float64
 4    Estimated Employed                      740 non-null    float64
 5    Estimated Labour Participation Rate (%) 740 non-null    float64
 6   Area                                     740 non-null    object
dtypes: float64(3), object(4)
memory usage: 42.1+ KB
```

In [6]:
```
df1.describe()
```

Out[6]:

	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)
count	740.000000	7.400000e+02	740.000000
mean	11.787946	7.204460e+06	42.630122
std	10.721298	8.087988e+06	8.111094
min	0.000000	4.942000e+04	13.330000
25%	4.657500	1.190404e+06	38.062500
50%	8.350000	4.744178e+06	41.160000
75%	15.887500	1.127549e+07	45.505000
max	76.740000	4.577751e+07	72.570000

In [7]:
```
df1.isnull().sum()
```

Out[7]:
```
Region                                   28
 Date                                    28
 Frequency                               28
 Estimated Unemployment Rate (%)         28
 Estimated Employed                      28
 Estimated Labour Participation Rate (%) 28
Area                                     28
dtype: int64
```

In [8]:
```
df1.shape
```

Out[8]:
```
(768, 7)
```

In [9]:
```
df1=df1.dropna()
```

In [10]:
```
df1.isnull().sum()
```

Out[10]:
```
Region                                     0
 Date                                      0
 Frequency                                 0
 Estimated Unemployment Rate (%)           0
 Estimated Employed                        0
 Estimated Labour Participation Rate (%)   0
Area                                       0
dtype: int64
```

In [11]:
```
df1.describe()
```

Out[11]:

	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)
count	740.000000	7.400000e+02	740.000000
mean	11.787946	7.204460e+06	42.630122
std	10.721298	8.087988e+06	8.111094
min	0.000000	4.942000e+04	13.330000
25%	4.657500	1.190404e+06	38.062500
50%	8.350000	4.744178e+06	41.160000
75%	15.887500	1.127549e+07	45.505000
max	76.740000	4.577751e+07	72.570000

In [12]:
```
#remove leading and trailing spaces from the column names
df1.columns = df1.columns.str.strip()
```

In [13]:
```
df1["Estimated Unemployment Rate (%)"]
```

Out[13]:
```
0       3.65
1       3.05
2       3.75
3       3.32
4       5.17
       ...
749     7.55
750     6.67
751    15.63
752    15.22
753     9.86
Name: Estimated Unemployment Rate (%), Length: 740, dtype: float64
```

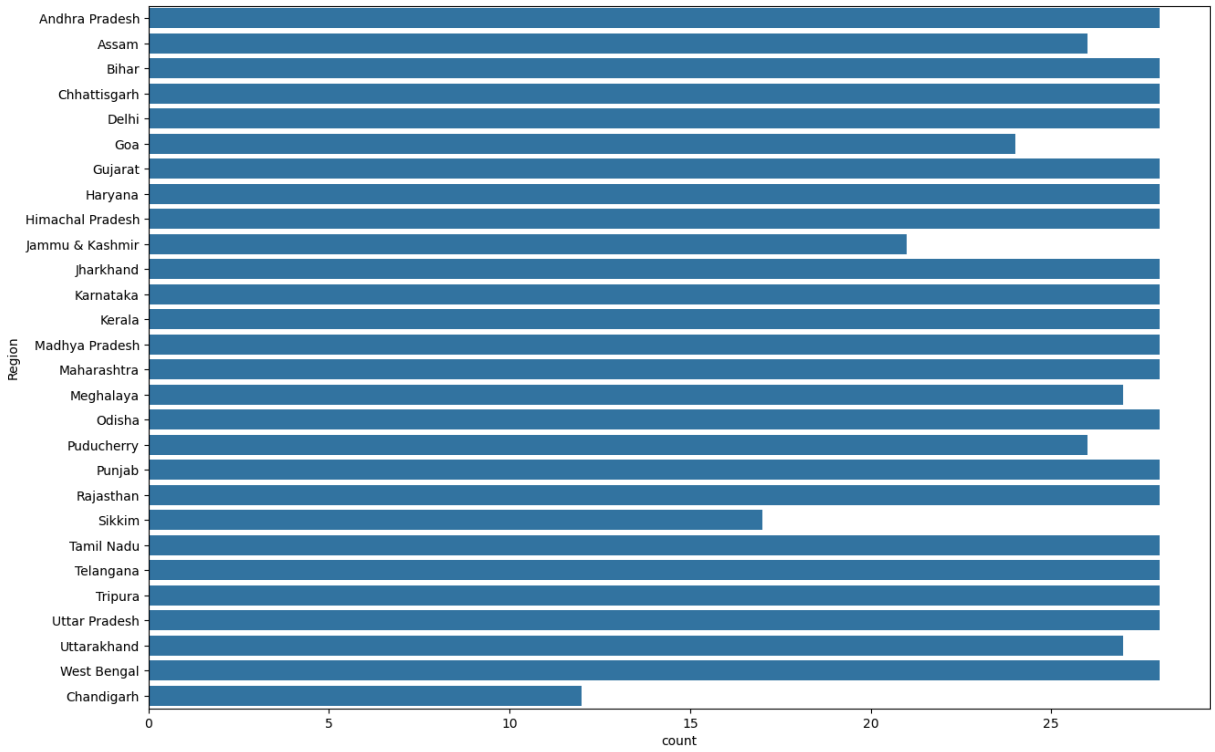
In [14]:
```
df1.duplicated().any()
```

Out[14]:
```
False
```

In [15]:
```
df1.Region.value_counts()
```

Out[15]:
```
Region
Andhra Pradesh      28
Kerala              28
West Bengal         28
Uttar Pradesh       28
Tripura             28
Telangana           28
Tamil Nadu          28
Rajasthan           28
Punjab              28
Odisha              28
Madhya Pradesh      28
Maharashtra         28
Karnataka           28
Jharkhand           28
Himachal Pradesh    28
Haryana             28
Gujarat             28
Delhi               28
Chhattisgarh        28
Bihar               28
Meghalaya           27
Uttarakhand         27
Assam               26
Puducherry          26
Goa                 24
Jammu & Kashmir     21
Sikkim              17
Chandigarh          12
Name: count, dtype: int64
```

In [16]:
```
plt.figure(figsize=(15,10))
sns.countplot(y="Region",data=df1)
plt.show()
```



Analysis: The count plot provides an overview of the number of unemployment rate observations across different regions. Analyzing the distribution reveals whether certain regions are over- or under-represented in the dataset.n.

Conclusion: The data collection appears uneven across regions, which may lead to biases in regional comparisons. Regions with fewer observations might not reflect the true economic conditions, necessitating careful interpretation.

In [19]:
```
#Changing the datatype of 'Date' from object to datetime
df1['Date'] = pd.to_datetime(df1['Date'],dayfirst = True)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 740 entries, 0 to 753
Data columns (total 7 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   Region                                   740 non-null    object
 1   Date                                     740 non-null    datetime64[ns]
 2   Frequency                                740 non-null    object
 3   Estimated Unemployment Rate (%)          740 non-null    float64
 4   Estimated Employed                       740 non-null    float64
 5   Estimated Labour Participation Rate (%)  740 non-null    float64
 6   Area                                     740 non-null    object
dtypes: datetime64[ns](1), float64(3), object(3)
memory usage: 46.2+ KB
```

In [20]:
```
#Extracting month from date attribute
df1['month_int'] = df1['Date'].dt.month
df1.head()
```

Out[20]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area	month_int
0	Andhra Pradesh	2019-05-31	Monthly	3.65	11999139.0	43.24	Rural	5
1	Andhra Pradesh	2019-06-30	Monthly	3.05	11755881.0	42.05	Rural	6
2	Andhra Pradesh	2019-07-31	Monthly	3.75	12086707.0	43.50	Rural	7
3	Andhra Pradesh	2019-08-31	Monthly	3.32	12285693.0	43.97	Rural	8
4	Andhra Pradesh	2019-09-30	Monthly	5.17	12256762.0	44.68	Rural	9

The months are in integer datetype. We need to convert the months into words for better analysis.

In [21]:
```
df1['month'] = df1['month_int'].apply(lambda x: calendar.month_abbr[x])
df1.head()
```

Out[21]:

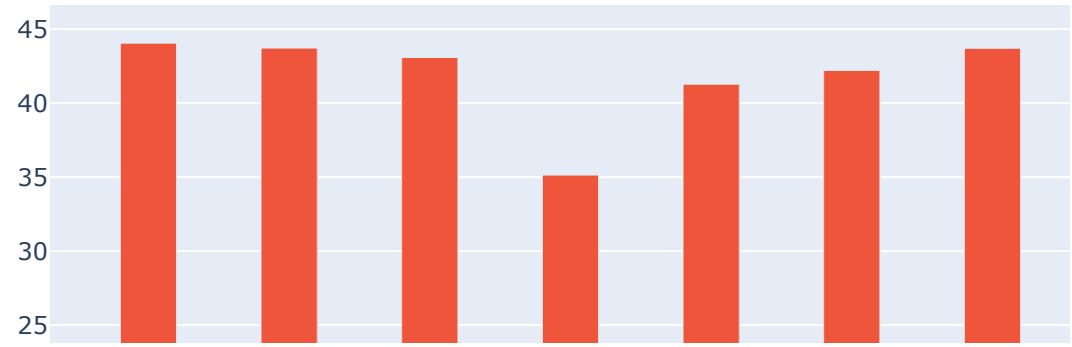
	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area	month_int
0	Andhra Pradesh	2019-05-31	Monthly	3.65	11999139.0	43.24	Rural	5
1	Andhra Pradesh	2019-06-30	Monthly	3.05	11755881.0	42.05	Rural	6
2	Andhra Pradesh	2019-07-31	Monthly	3.75	12086707.0	43.50	Rural	7
3	Andhra Pradesh	2019-08-31	Monthly	3.32	12285693.0	43.97	Rural	8
4	Andhra Pradesh	2019-09-30	Monthly	5.17	12256762.0	44.68	Rural	9

Numeric data grouped by months.

In [25]:
```
data = df1.groupby(['month'])[['Estimated Unemployment Rate (%)','Estimated Employe
data=pd.DataFrame(data).reset_index()
```

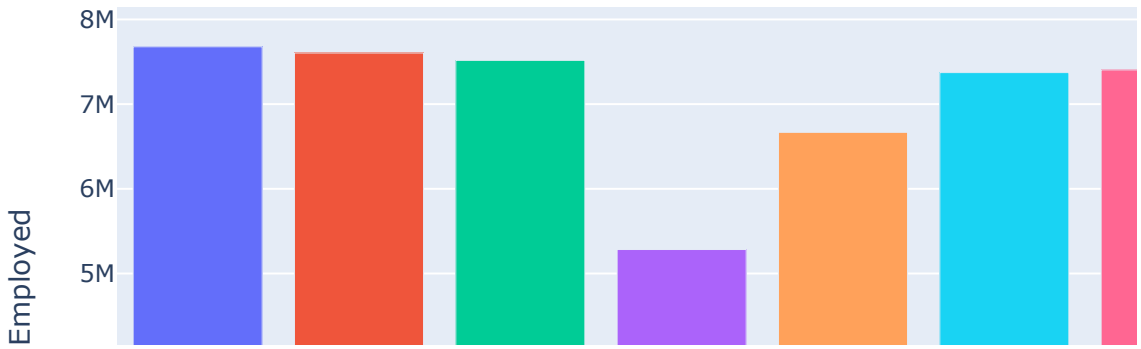
Bar plot of unemployment rate and labour participation rate.

In [31]:
```
month = data.month
unemployment_rate = data['Estimated Unemployment Rate (%)']
labour_participation_rate = data['Estimated Labour Participation Rate (%)']

fig = go.Figure()

fig.add_trace(go.Bar(x = month,y = unemployment_rate,name = 'Unemployment Rate'))
fig.add_trace(go.Bar(x = month,y = labour_participation_rate,name = 'Labour Partici

fig.update_layout(title = 'Unemployment Rate and Labour Participation',
                  xaxis = {'categoryorder':'array','categoryarray':['Jan','Feb',
fig.show()
```



In [95]:
```
#Bar plot of estimated employed citizen in every month
import plotly.express as px
```

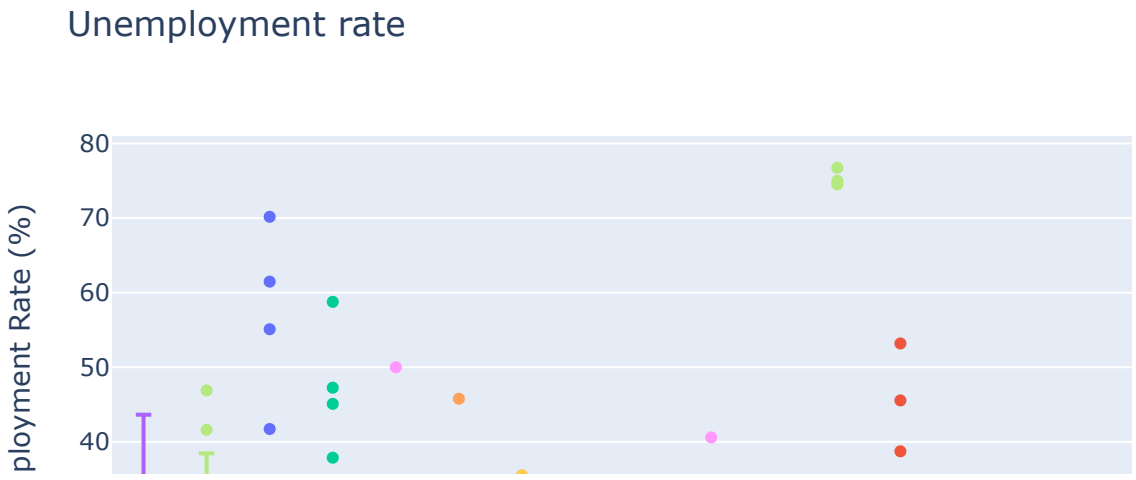
In [37]:
```
fig = px.bar(data,x='month',y='Estimated Employed',color='month',
             category_orders ={'month':['Jan','Feb','Mar','Apr','May','Jun','Jul','A
             title='Estimated employed people from Jan 2020 to Oct 2020')
fig.show()
```



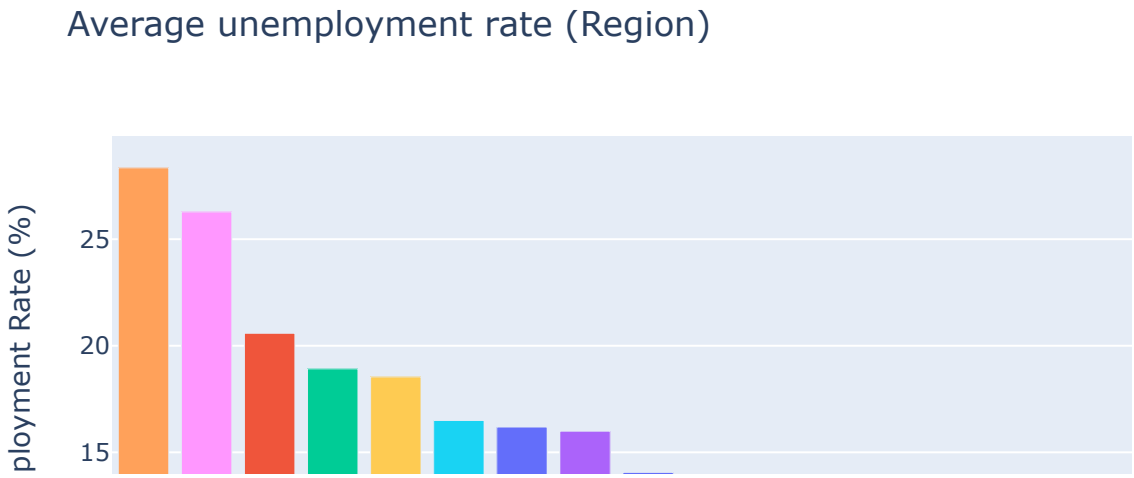
Region(State) wise Analysis

In [45]:
```
Region =  df1.groupby(['Region'])[['Estimated Unemployment Rate (%)','Estimated Emp
Region = pd.DataFrame(Region).reset_index()
```

In [49]:
```
# Box plot
fig = px.box(data_frame=df1,x='Region',y='Estimated Unemployment Rate (%)',color='R
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```



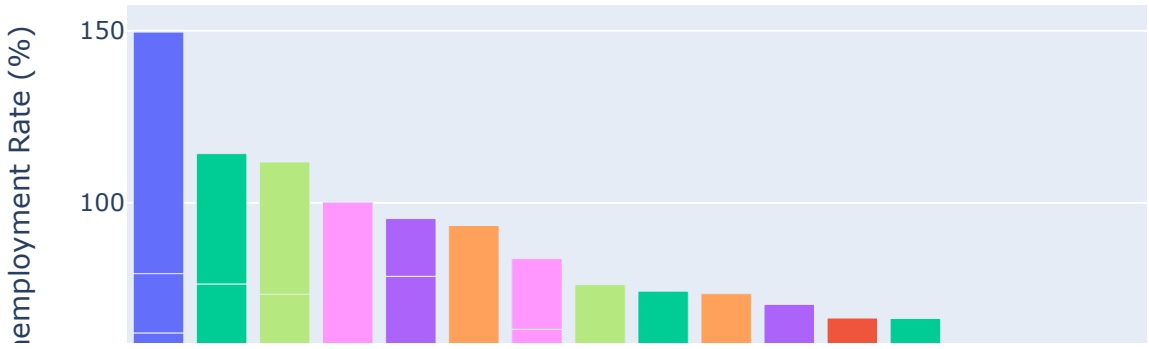
In [55]:
```
# average unemployment rate bar plot

fig = px.bar(Region,x='Region',y='Estimated Unemployment Rate (%)',color='Region',t
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```



Hariyana and Tripura were having the highest average amount of Unemployment rate.

Meghalaya was having the lowest average amount of Unemployment ra.te

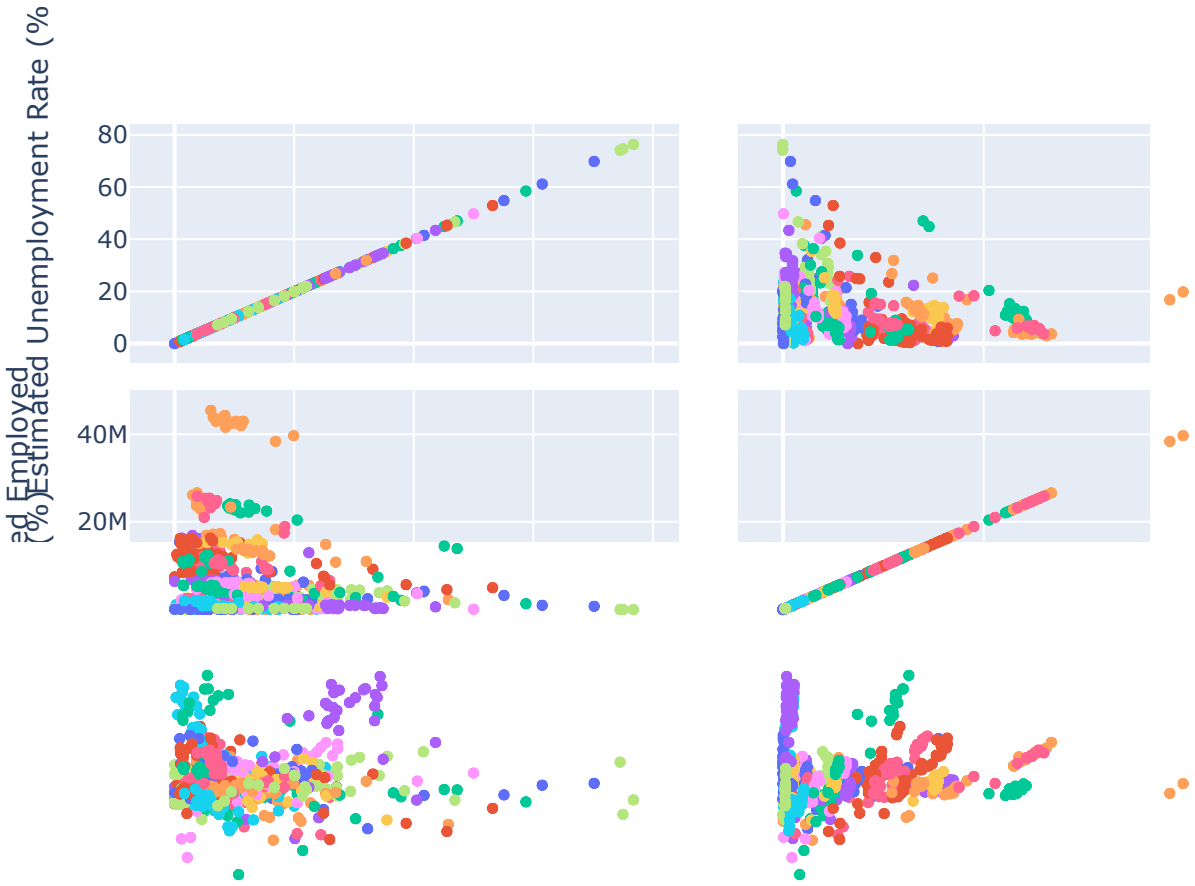
In [58]:
```
# Bar plot Unemployment Rate (monthly)

fig = px.bar(df1,x='Region',y='Estimated Unemployment Rate (%)',animation_frame='mo
             title='Unemployment rate from Jan 2020 to Oct 2020(Region)')

fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```



Monthly unemployment rate

In [66]:
```
# numeric data grouped by region

Region = df1.groupby(['Region'])[['Estimated Unemployment Rate (%)','Estimated Empl
Region = pd.DataFrame(Region).reset_index()
```

In [70]:
```
#Scatter plot
pd.DataFrame.iteritems = pd.DataFrame.items
fig= px.scatter_matrix(df1,dimensions=['Estimated Unemployment Rate (%)','Estimated
fig.show()
```



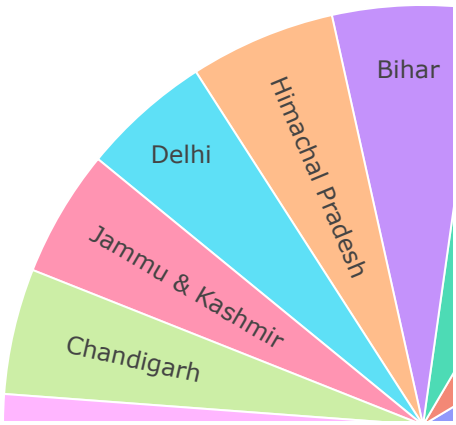
In [74]:
```
unemployment =df1.groupby(['Region'])['Estimated Unemployment Rate (%)'].mean().res
unemployment.head()
```

Out[74]:

	Region	Estimated Unemployment Rate (%)
0	Andhra Pradesh	7.477143
1	Assam	6.428077
2	Bihar	18.918214
3	Chandigarh	15.991667
4	Chhattisgarh	9.240357

In [78]:
```
fig = px.sunburst(unemployment,path=['Region'],values='Estimated Unemployment Rate
                  title ='Unemployment rate in state and region',height=600)
fig.show()
```

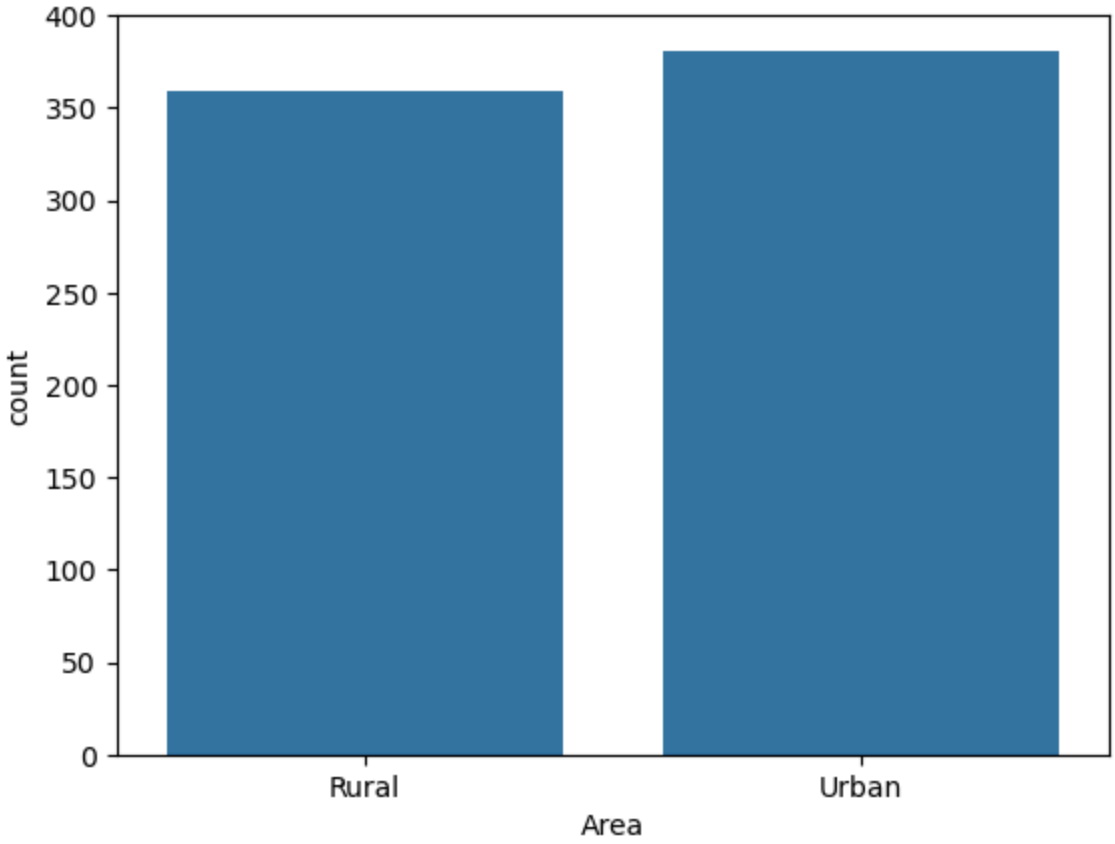
Unemployment rate in state and region



In [107…
```
df1["Area"].value_counts()
```

Out[107…
```
Area
Urban    381
Rural    359
Name: count, dtype: int64
```

In [109…
```
sns.countplot(x="Area",data=df1)
plt.show()
```



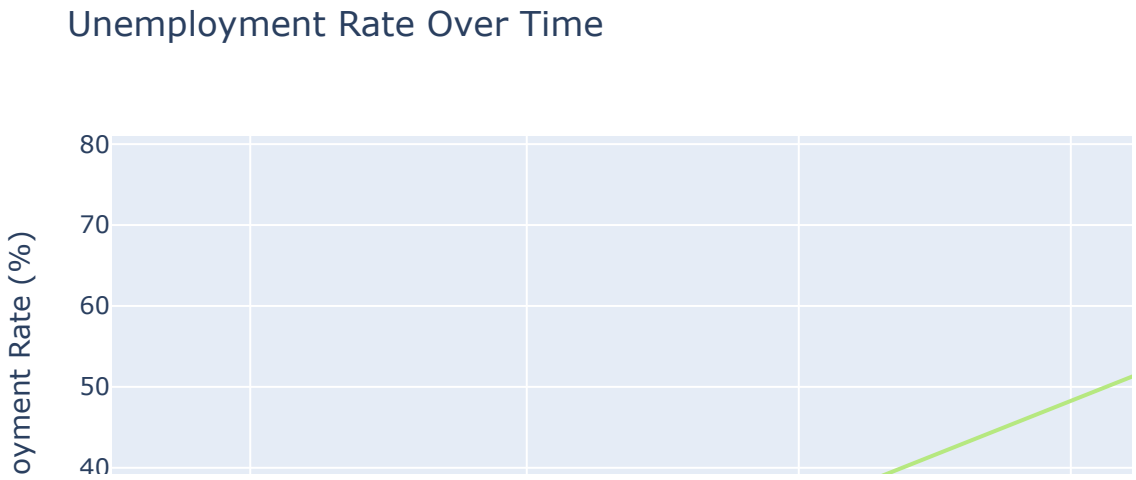
Analysis: The countplot illustrates the distribution of observations across two categories: Rural and Urban. The plot reveals a higher frequency of Urban observations compared to Rural.

Conclusion: A potential bias towards urban areas exists in the data, which may limit the generalizability of findings to rural India and obscure the true nature of unemployment challenges in these regions.

Time Series Line Plot for Unemployment Rate by Region

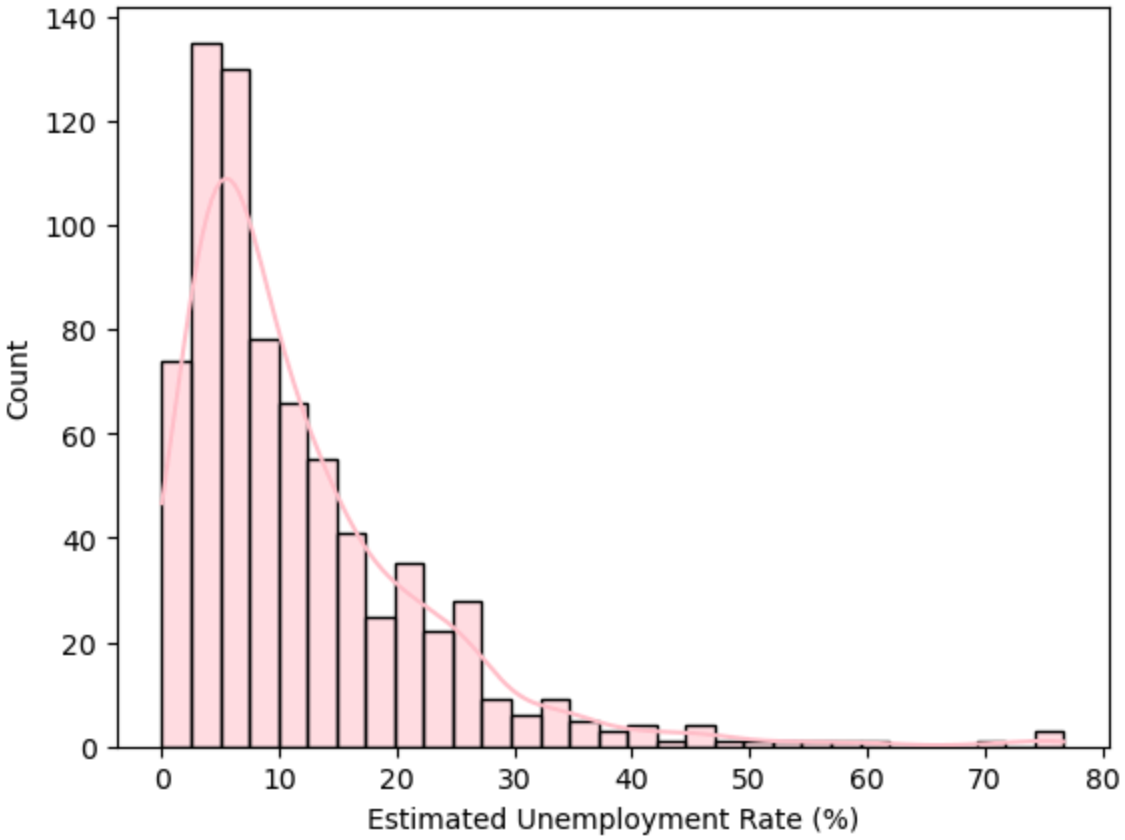
In [116…

```
fig = px.line(df1, x='Date', y="Estimated Unemployment Rate (%)", color='Region',
              title='Unemployment Rate Over Time', template='plotly')
fig.show()
```



In [127…

```
sns.histplot(data=df1, x="Estimated Unemployment Rate (%)", kde=True,color="pink")
plt.show()
```

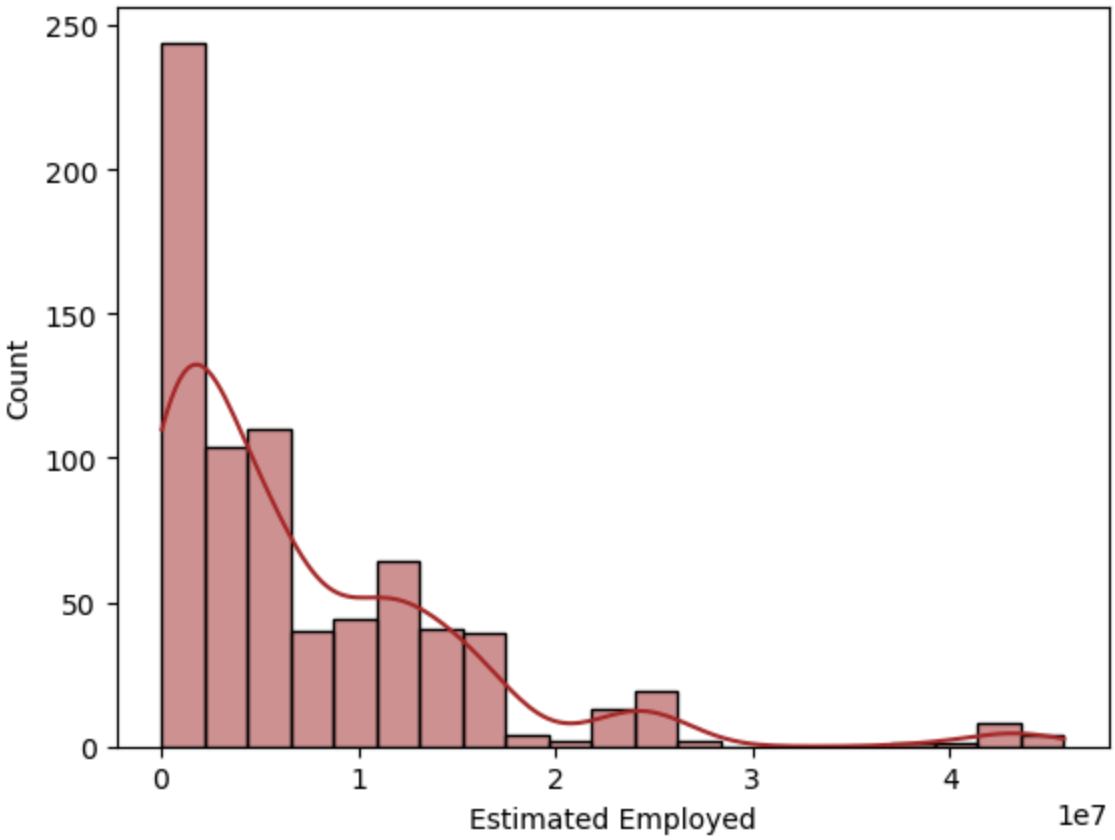


Analysis: The histogram shows the distribution of the estimated unemployment rates across various regions in India. The majority of regions have unemployment rates concentrated at the lower end of the spectrum, with a few regions experiencing much higher rates. The KDE curve overlaid on the histogram provides a smooth estimate of the distribution, highlighting the skewness towards lower unemployment rates.

Conclusion: Most regions have relatively low unemployment rates, with a majority falling below 20%. However, there are a few regions with significantly higher unemployment rates, which may indicate regional economic challenges or structural unemployment issues. The left-skewed distribution suggests that while low unemployment is common, high unemployment, though less frequent, is a concern in certain areas.

In [122…

```
sns.histplot(data=df1, x="Estimated Employed", kde=True,color="brown")
plt.show()
```

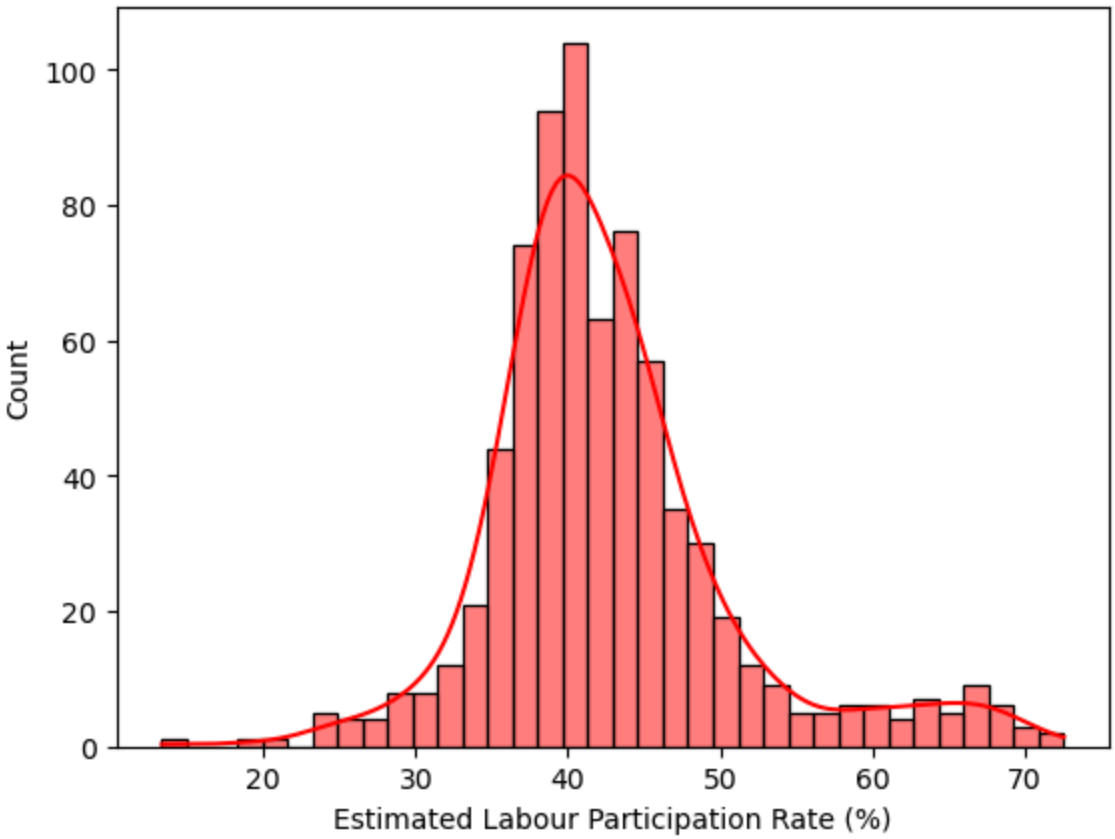


Analysis: The histogram shows that most regions have lower numbers of employed individuals, with a few regions showing much higher employment figures. The right-skewed distribution is evident, with the KDE curve confirming this pattern.

Conclusion: Most regions have relatively low employment numbers, while a few outliers have significantly higher figures, suggesting regional economic or population differences.

In [125…

```
sns.histplot(data=df1, x="Estimated Labour Participation Rate (%)", kde=True,color=
plt.show()
```

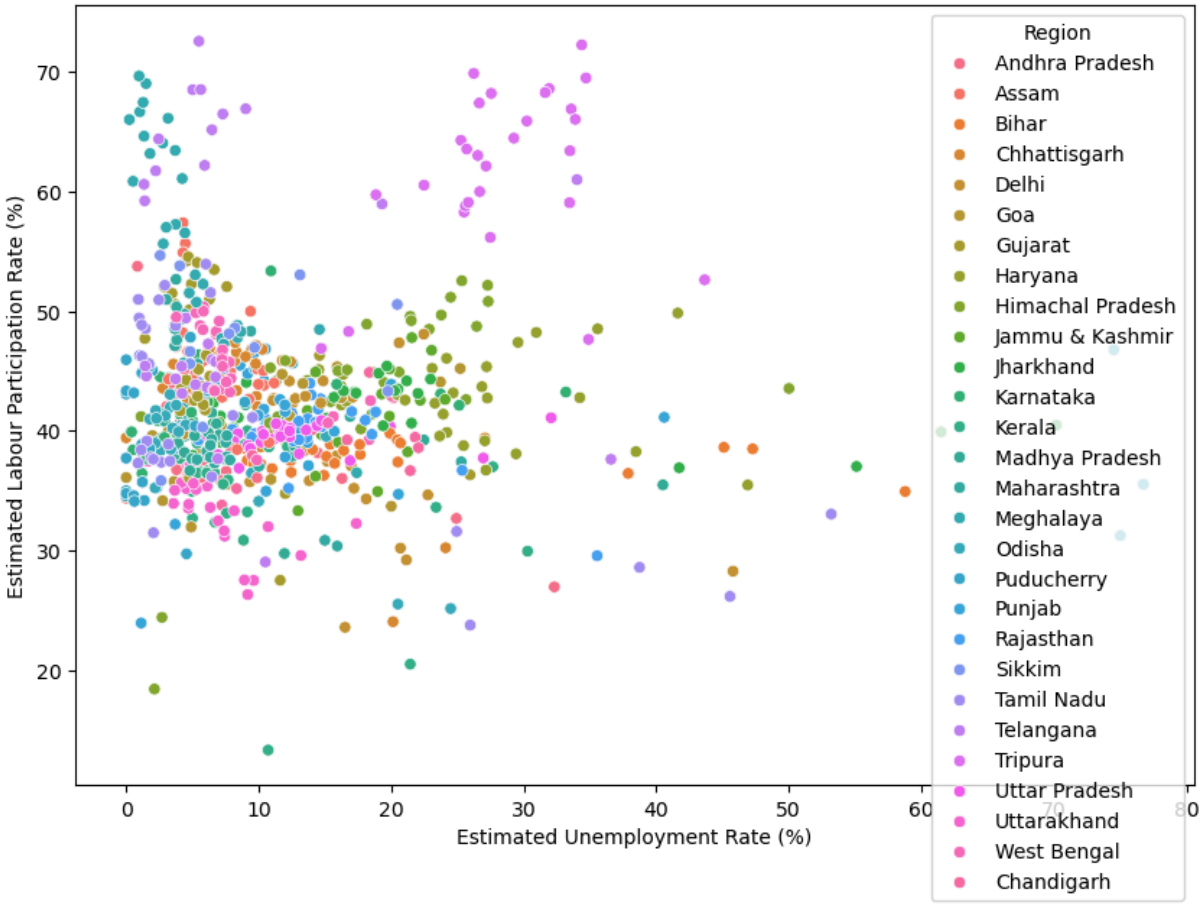


Analysis: The histogram illustrates the distribution of the estimated labor participation rate across different regions. The data is fairly normally distributed, with most regions having a labor participation rate around 40%. The KDE curve confirms this central tendency.

Conclusion: Most regions have a labor participation rate centered around 40%, indicating a typical engagement level. The normal distribution suggests consistent labor participation across regions with fewer deviations towards the extremes.

In [130…

```
plt.figure(figsize=(10,7))
sns.scatterplot(data=df1, x="Estimated Unemployment Rate (%)", y="Estimated Labour
plt.show()
```

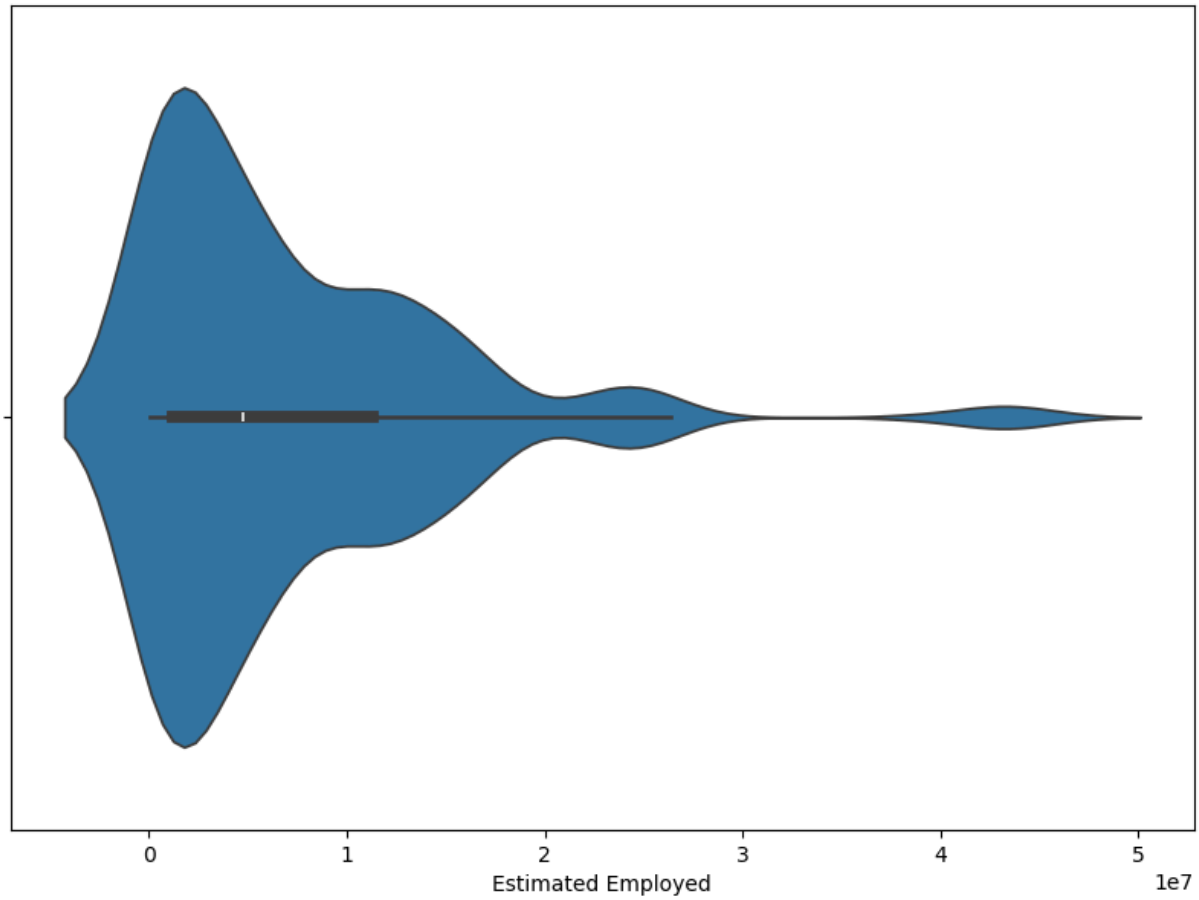


Analysis: The scatter plot illustrates a negative correlation between unemployment and labor participation rates across Indian states. Higher unemployment is generally associated with lower labor force participation.

Conclusion: The data suggests that increasing labor force participation may require addressing unemployment challenges in India. Targeted policies addressing regional disparities could be beneficial.

In [133…

```
plt.figure(figsize=(10,7))
sns.violinplot(x=df1["Estimated Employed"])
plt.show()
```



Analysis: The violin plot shows that most regions have a relatively low estimated number of employed individuals.

Conclusion: Employment numbers are concentrated in the lower range, with fewer regions reporting significantly higher employment figures.

Unemployment rate before and after Lockdown

In [84]:
```
# data representation before and after lockdown

before_lockdown = df1[(df1['month_int']>=1) &(df1['month_int'] <4)]
after_lockdown = df1[(df1['month_int'] >=4) & (df1['month_int'] <=6)]
```

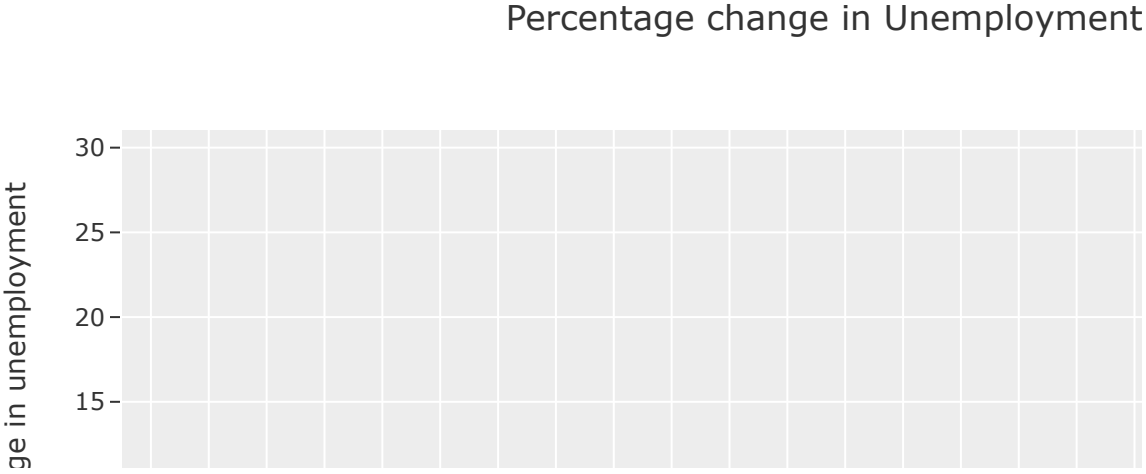
In [86]:
```
af_lockdown = after_lockdown.groupby('Region')['Estimated Unemployment Rate (%)'].m

lockdown = before_lockdown.groupby('Region')['Estimated Unemployment Rate (%)'].mea
lockdown['unemployment rate before lockdown'] = af_lockdown['Estimated Unemployment

lockdown.columns = ['Region','unemployment rate before lockdown','unemployment rate
lockdown.head()
```

Out[86]:

	Region	unemployment rate before lockdown	unemployment rate after lockdown
0	Andhra Pradesh	6.243333	11.126000
1	Assam	6.480000	6.563333
2	Bihar	14.276667	27.459000
3	Chandigarh	19.366667	12.656667
4	Chhattisgarh	8.683333	12.720000

In [88]:
```
# unenployment rate change after lockdown

lockdown['rate change in unemployment'] =round(lockdown['unemployment rate before l
                                                /lockdown['unemployment rate after lo
```

In [90]:
```
fig = px.bar(lockdown,x='Region',y='rate change in unemployment',color='rate change
            title='Percentage change in Unemployment rate in each state after lockd
fig.update_layout(xaxis={'categoryorder':'total ascending'})
fig.show()
```



Highest,Lowest Unemployment rate

In [102…

```
avg_unemployment_rate=df1.groupby('Region')['Estimated Unemployment Rate (%)'].mean
state_with_highest_unemployment=avg_unemployment_rate.idxmax()
high_unemployment_rate=avg_unemployment_rate.max()
state_with_lowest_unemployment=avg_unemployment_rate.idxmin()
low_unemploy_rate=avg_unemployment_rate.min()
print(f"state  with  high employment : {state_with_highest_unemployment}")
print(f"high employment rate : {high_unemployment_rate}")
print(f"state  with  low employment : {state_with_lowest_unemployment}")
print(f"low employment rate : {low_unemploy_rate}")
```

```
state  with  high employment : Tripura
high employment rate : 28.350357142857142
state  with  low employment : Meghalaya
low employment rate : 4.7988888888888885
```