In [6]:
```
#Importing libraries
```

In [7]:
```
import pandas as pd
import numpy as np
```

In [8]:
```
#Importing data
```

In [9]:
```
df = pd.read_excel('rdata.xlsx','Sheet2')
df = df.T
df
```

Out[9]:

	0	1	2	3	4	5	6	7	8	
1	1910-01-01	NaN	0.270968	0.229032	0	0.0741935	0.33871	NaN	0	0.
2	1910-02-01	NaN	1.45	1.1	1.23214	0.453571	1.01071	NaN	2.31429	0.
3	1910-03-01	NaN	5.86452	8.6871	5.8871	11.2355	4.96452	NaN	5.38387	5
4	1910-04-01	NaN	22.08	16.5333	18.1233	22.25	8.53	NaN	12.12	1
5	1910-05-01	NaN	12.1645	11.3548	9.38387	10.0258	8.2129	NaN	8.63871	7
...	...	...	...	...	...	...	...	...	...	
1088	2000-08-01	45.0161	29.2968	NaN	NaN	NaN	NaN	NaN	NaN	
1089	2000-09-01	22.5933	20.62	NaN	NaN	NaN	NaN	NaN	NaN	
1090	2000-10-01	8.41613	5.70323	NaN	NaN	NaN	NaN	NaN	NaN	
1091	2000-11-01	0.356667	0	NaN	NaN	NaN	NaN	NaN	NaN	
1092	2000-12-01	NaN	0	NaN	NaN	NaN	NaN	NaN	NaN	

1092 rows × 20 columns

Pre-processing

In [10]:
```
df.columns = ['Date',  'Amraghat','Badarpur','Barkhola','Bhanga','Bikrampur','Dewan','Dholai','Dullabcherra','Jafirbond','Koyah','Katlicherra','Lakhipur',
'Moneirkhal',
'Palonghat',
'Patharkandi',
'Salchapara',
'Silchar',
'Silchar Aero',
'Silguri'
]
```

In [11]:
```
df =  df.set_index("Date")
df.head()
```

Out[11]:

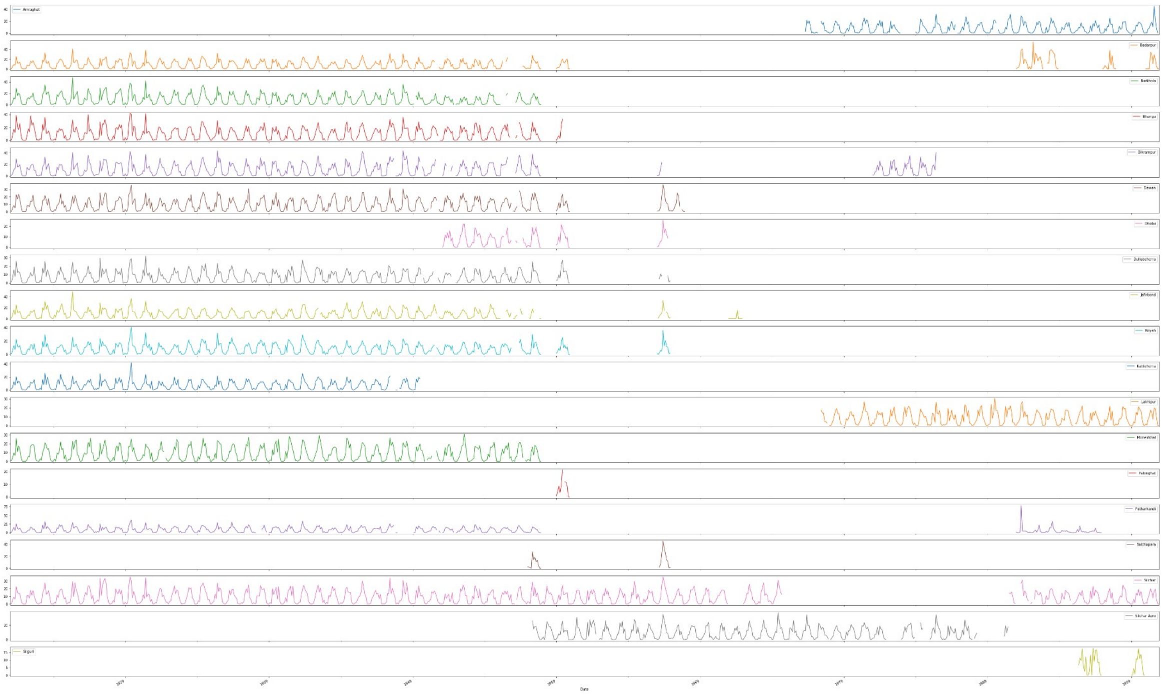
	Amraghat	Badarpur	Barkhola	Bhanga	Bikrampur	Dewan	Dholai	Dullabcherra
Date								
1910-01-01	NaN	0.270968	0.229032	0	0.0741935	0.33871	NaN	0
1910-02-01	NaN	1.45	1.1	1.23214	0.453571	1.01071	NaN	2.31429
1910-03-01	NaN	5.86452	8.6871	5.8871	11.2355	4.96452	NaN	5.38387
1910-04-01	NaN	22.08	16.5333	18.1233	22.25	8.53	NaN	12.12
1910-05-01	NaN	12.1645	11.3548	9.38387	10.0258	8.2129	NaN	8.63871

In [12]:
```
#Plotting the data
```

In [13]:

```
df.loc['1921-01-01':].plot(subplots =True,figsize = (60,40))
```

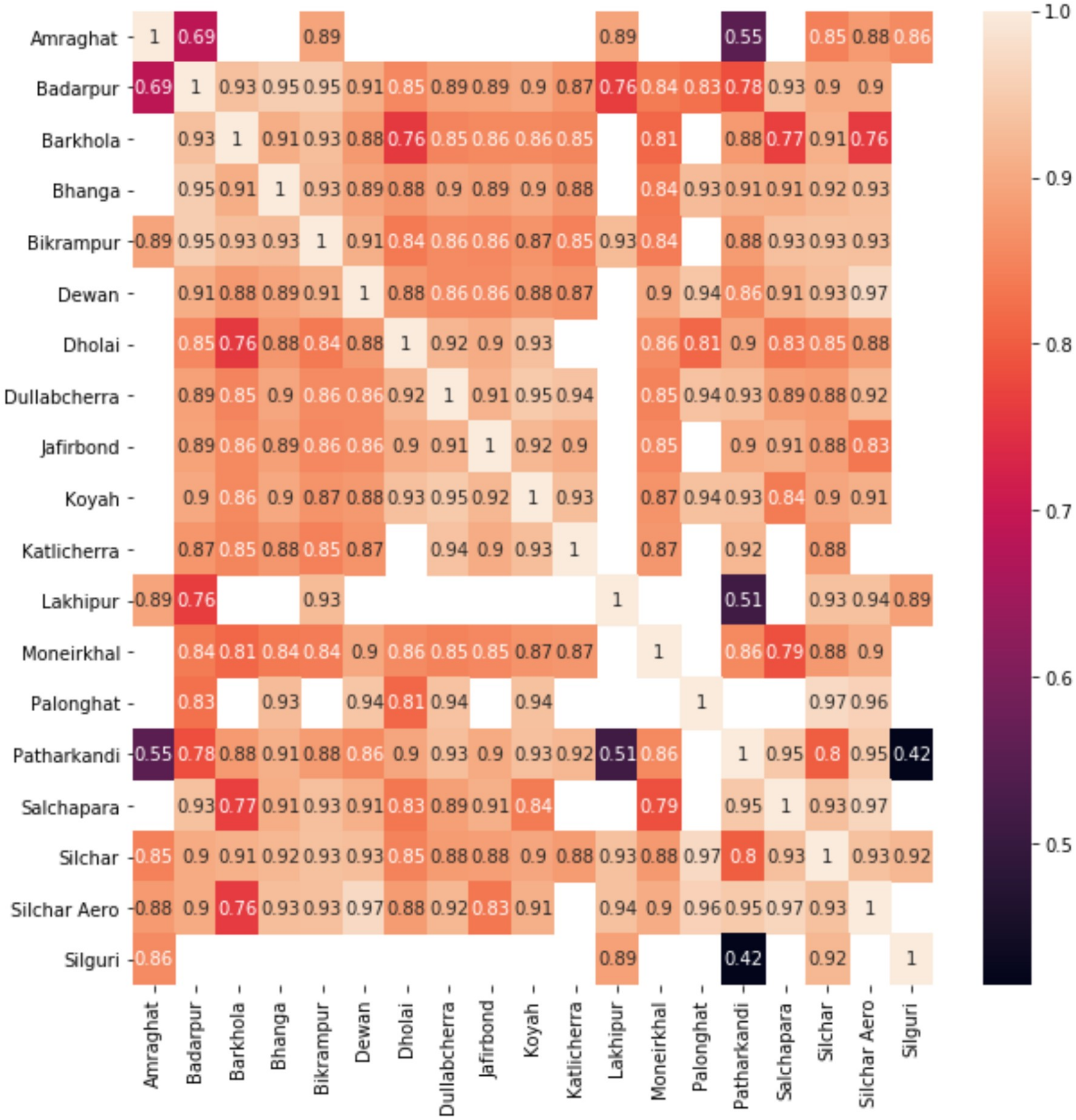
```
Out[13]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x000001134EE
         ABF88>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113534
         516C8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113534
         8A488>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113534
         BF488>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113534
         F5448>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113535
         2C388>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113535
         67388>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113535
         9F348>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113535
         A4E88>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113535
         DCF08>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113536
         47308>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113536
         7D208>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113536
         B51C8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113536
         EE1C8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113537
         24188>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113537
         5E148>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113537
         96108>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113537
         CF0C8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x00000113538
         06088>],
               dtype=object)
```



In [14]:
```
#Coorelation matrix
```

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
fig,ax = plt.subplots(figsize = (10,10))
df= df.astype(float)
corr  = df.corr()
sns.heatmap(corr,annot = True,fmt = '.2g',xticklabels = corr.columns.va
lues,yticklabels = corr.columns.values)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11355e24d08>
```



In [16]:
```
df = df.fillna(0)
```

In [17]:

```
keys = df.keys()
keys
```

Out[17]:

```
Index(['Amraghat', 'Badarpur', 'Barkhola', 'Bhanga', 'Bikrampur', 'De
wan',
       'Dholai', 'Dullabcherra', 'Jafirbond', 'Koyah', 'Katlicherra',
       'Lakhipur', 'Moneirkhal', 'Palonghat', 'Patharkandi', 'Salchap
ara',
       'Silchar', 'Silchar Aero', 'Silguri'],
      dtype='object')
```

In [18]:

```
#Spliting Train data
```

In [19]:

```
X_train = df.loc['1921-01-01':'1957-10-01',['Bhanga','Dholai','Dullabch
erra','Jafirbond','Koyah','Katlicherra','Salchapara','Silchar Aero']].a
stype(float).values
```

In [20]:

```
y_train = df.loc['1921-01-01':'1957-10-01','Patharkandi'].astype(floa
t).values
```

Model 1

In [21]:

```
#Feature Scaling
```

In [22]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
```

In [23]:

```
#Hidden layers and input layer initialization
```

In [24]:
```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add((Dense(12,activation='linear',input_shape = (8,))))
model.add((Dense(8,activation='linear')))
model.add((Dense(1,activation='linear')))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 12)                108       
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104       
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9         
=================================================================
Total params: 221
Trainable params: 221
Non-trainable params: 0
_________________________________________________________________
```

In [25]:
```
model.compile(loss='mean_squared_error' ,optimizer='adam' ,metrics=['ac
curacy'])
model.fit(X_train, y_train, epochs=200, verbose=0)
```

Out[25]:
```
<tensorflow.python.keras.callbacks.History at 0x1135e85ffc8>
```

In [26]:
```
y_pred=model.predict(X_train)
y_pred[:10]
```

Out[26]:
```
array([[ 0.78136927],
       [ 1.0848291 ],
       [ 6.6758637 ],
       [14.945499  ],
       [ 8.328669  ],
       [26.879194  ],
       [13.4290695 ],
       [13.083725  ],
       [14.766735  ],
       [ 5.4336443 ]], dtype=float32)
```

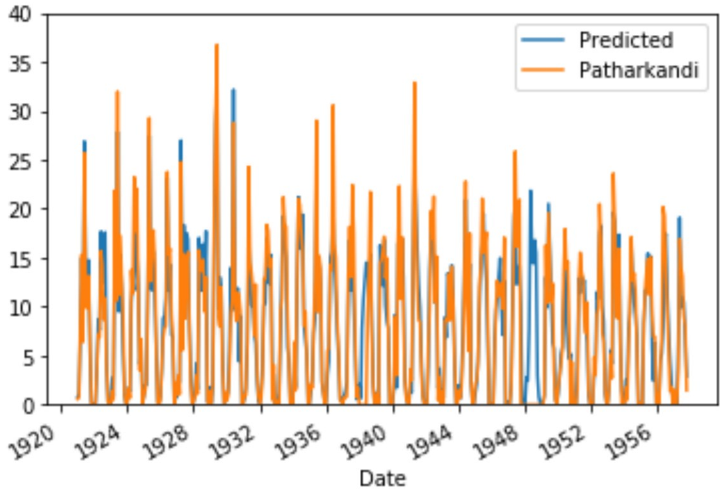
In [27]:
```
get_ipython().magic('pylab inline')
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [28]:
```
plot (df.loc['1921-01-01':'1957-10-01'].index,y_pred, label='Predicted
')
df['Patharkandi'].loc[ '1921-01-01':'1957-10-01'].plot()
figsize(200,10)
ylim(0, 40)
legend (loc='Best')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: M
atplotlibDeprecationWarning: Unrecognized location 'Best'. Falling ba
ck on 'best'; valid locations are
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.
  """
```

Out[28]:
```
<matplotlib.legend.Legend at 0x1135f893d88>
```



model2

In [29]:
```
#Creating independent variables following time series
```

In [30]:

```
m=[]
y=[]
for i in range(1910,2001):
  c=1
  while c<=12:
    m.append(c)
    c = c+1
    y.append(i)
```

In [31]:

```
patharkandi=pd.DataFrame(df['Patharkandi'])
patharkandi[ 'Month' ]=m
patharkandi[ 'Year' ]=y
patharkandi.head()
```

Out[31]:

	Patharkandi	Month	Year
Date			
1910-01-01	0.0	1	1910
1910-02-01	0.0	2	1910
1910-03-01	0.0	3	1910
1910-04-01	0.0	4	1910
1910-05-01	0.0	5	1910

In [32]:

```
X_train1=patharkandi.loc['Jan/1921' : 'Oct/1957', ['Month', 'Year']].va
lues
X_train1[0:5]

Y_train1=patharkandi.loc[ 'Jan/1921':'Oct/1957','Patharkandi'].astype('
float').values
Y_train1[0:5]
```

Out[32]:

```
array([ 0.58064516,  0.725     ,  8.14516129, 15.3       ,  6.4
])
```

In [33]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler().fit(X_train1)
X_train1=scaler.transform(X_train1)
X_train1[0:5]
```

Out[33]:

```
array([[-1.59076734, -1.68531391],
       [-1.30034224, -1.68531391],
       [-1.00991714, -1.68531391],
       [-0.71949205, -1.68531391],
       [-0.42906695, -1.68531391]])
```

In [34]:
```
from keras.models import Sequential
from keras.layers import Dense

model1=Sequential()
model1.add((Dense(12, activation='linear' ,input_shape=(2,))))
model1.add((Dense(8, activation='linear')))
model1.add((Dense(1, activation='linear')))

model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_3 (Dense)              (None, 12)                36        
_________________________________________________________________
dense_4 (Dense)              (None, 8)                 104       
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 9         
=================================================================
Total params: 149
Trainable params: 149
Non-trainable params: 0
_________________________________________________________________
```

In [35]:
```
model1.compile(loss='mean_squared_error' ,optimizer='adam' ,metrics=['accuracy'])
model1.fit(X_train1, Y_train1, epochs=200, verbose=0)
```

Out[35]:
```
<tensorflow.python.keras.callbacks.History at 0x1135fb324c8>
```

In [36]:
```
y_pred1=model1.predict(X_train1)
y_pred1[:10]
```

Out[36]:
```
array([[8.398125],
       [8.472678],
       [8.54723 ],
       [8.621783],
       [8.696336],
       [8.770888],
       [8.845441],
       [8.919993],
       [8.994546],
       [9.069098]], dtype=float32)
```

In [37]:

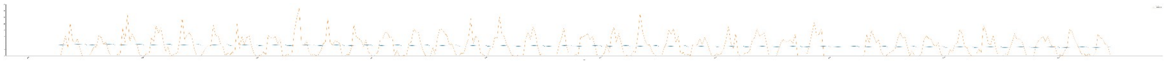
```
plot (patharkandi.loc[ 'Jan/1921':'Oct/1957'].index,y_pred1, label='Pre
dicted')
patharkandi['Patharkandi'].loc['Jan/1921': 'Oct/1957' ].plot()

figsize(200,15)
ylim(0,40)
legend(loc='Best')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: M
atplotlibDeprecationWarning: Unrecognized location 'Best'. Falling ba
ck on 'best'; valid locations are
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.
```

Out[37]:

```
<matplotlib.legend.Legend at 0x11360e3b908>
```



model3

In [38]:

```
X_train2=df.loc[ 'Jan/1921':'Oct/1957',['Amraghat', 'Badarpur', 'Barkho
la', 'Bikrampur', 'Dewan','Lakhipur', 'Moneirkhal', 'Palonghat',
       'Silchar', 'Silguri']].astype(float).values
X_train2[0:5]
```

Out[38]:

```
array([[ 0.        ,  0.77741935,  0.57419355,  0.67419355,  0.980645
16,
         0.        ,  0.86774194,  0.        ,  0.83548387,  0.
],
       [ 0.        ,  0.14642857,  0.13571429,  0.98928571,  0.217857
14,
         0.        ,  0.75357143,  0.        ,  0.16428571,  0.
],
       [ 0.        ,  6.07096774,  5.11612903,  4.11290323,  5.187096
77,
         0.        ,  4.38064516,  0.        ,  6.06129032,  0.
],
       [ 0.        ,  8.98666667, 12.18333333, 10.88333333,  8.123333
33,
         0.        ,  9.51666667,  0.        ,  8.80666667,  0.
],
       [ 0.        ,  8.08709677, 10.03548387, 12.91612903,  5.022580
64,
         0.        ,  6.73225806,  0.        ,  6.53225806,  0.
]])
```

In [39]:

```
Y_train2=df.loc['Jan/1921':'Oct/1957','Patharkandi' ].astype('float').v
alues
Y_train2[0:5]
```

Out[39]:

```
array([ 0.58064516,  0.725     ,  8.14516129, 15.3       ,  6.4
])
```

In [40]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler().fit(X_train2)
X_train2=scaler.transform(X_train2)
X_train2[0:5]
```

Out[40]:

```
array([[ 0.        , -0.93843187, -0.95472127, -0.9743227 , -0.969638
4 ,
         0.        , -0.94383676,  0.        , -0.99107734,  0.
],
       [ 0.        , -1.01508167, -1.00228522, -0.94444263, -1.063905
53,
         0.        , -0.95807339,  0.        , -1.06981712,  0.
],
       [ 0.        , -0.29539644, -0.46203571, -0.64823115, -0.449795
25,
         0.        , -0.50579119,  0.        , -0.37802609,  0.
],
       [ 0.        ,  0.05878898,  0.3045779 , -0.00619379, -0.086928
26,
         0.        ,  0.13465102,  0.        , -0.05595974,  0.
],
       [ 0.        , -0.05048655,  0.07159035,  0.18657547, -0.470126
54,
         0.        , -0.21255407,  0.        , -0.32277579,  0.
]])
```

In [41]:

```
from keras.models import Sequential

from keras.layers import Dense

model3=Sequential()

model3.add((Dense(12, activation='linear' , input_shape=(10, ))))
model3.add((Dense(8, activation='linear')))
model3.add((Dense(1, activation='linear')))

model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 12)                132
_________________________________________________________________
dense_7 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 9
=================================================================
Total params: 245
Trainable params: 245
Non-trainable params: 0
_________________________________________________________________
```

In [42]:
```
model3.compile(loss='mean_squared_error' ,optimizer='adam' ,metrics=['accuracy'])
model3.fit(X_train2, Y_train2, epochs=200, verbose=0)
```

Out[42]:
```
<tensorflow.python.keras.callbacks.History at 0x11364de8648>
```

In [43]:
```
y_pred2=model3.predict(X_train2)
y_pred2[:10]
```

Out[43]:
```
array([[ 0.99550235],
       [ 0.6649972 ],
       [ 4.918351  ],
       [ 8.419894  ],
       [ 7.119585  ],
       [22.439768  ],
       [10.824287  ],
       [12.1160755 ],
       [12.360853  ],
       [ 3.9329264 ]], dtype=float32)
```

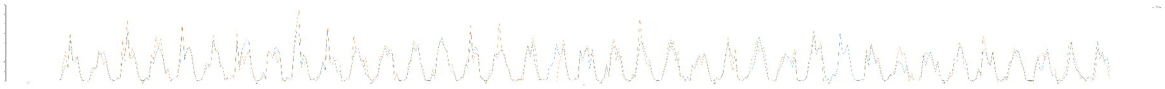
In [44]:
```
plot (df.loc[ 'Jan/1921' : 'Oct/1957'].index,y_pred2, label='Predicted')
df['Patharkandi'].loc['Jan/1921' : 'Oct/1957'].plot()
figsize(200,15)

ylim(0,40)

legend(loc='Best')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: MatplotlibDeprecationWarning: Unrecognized location 'Best'. Falling back on 'best'; valid locations are
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.
  import sys
```

Out[44]:
```
<matplotlib.legend.Legend at 0x11364ddae48>
```



Accuracy

In [45]:

```
#RMSE
from sklearn import metrics
rmse1 = np.sqrt(metrics.mean_squared_error(y_train, y_pred))
rmse2 = np.sqrt(metrics.mean_squared_error(y_train, y_pred1))
rmse3 = np.sqrt(metrics.mean_squared_error(y_train, y_pred2))
print("Root Mean Square Error",rmse1)
print("Root Mean Square Error",rmse2)
print("Root Mean Square Error",rmse3)
```

```
Root Mean Square Error 3.292577702814549
Root Mean Square Error 7.5246423384285555
Root Mean Square Error 3.7517170293485025
```

In [46]:

```
#MSE and MAE
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error as mse
mae1 = mae(y_pred,y_train)
print('Training Mean Absolute Error', mae1 )
mse1 = mse(y_pred, y_train)
print('Training Mean Squared Error', mse1 )
mae2 = mae(y_pred1,y_train)
print('Training Mean Absolute Error', mae2 )
mse2 = mse(y_pred2, y_train)
print('Training Mean Squared Error', mse2 )
mae3 = mae(y_pred2,y_train)
print('Training Mean Absolute Error', mae3 )
mse3 = mse(y_pred2, y_train)
print('Training Mean Squared Error', mse3 )
```

```
Training Mean Absolute Error 1.8976202957956696
Training Mean Squared Error 10.841067929071531
Training Mean Absolute Error 6.352984061331217
Training Mean Squared Error 14.075380668303552
Training Mean Absolute Error 2.243752769602636
Training Mean Squared Error 14.075380668303552
```

In [47]:

```
#R_2
from sklearn.metrics import r2_score
r2_1 = r2_score(y_train,y_pred)
print('R2 score',r2_1)
r2_2 = r2_score(y_train,y_pred1)
print('R2 score',r2_2)
r2_3 = r2_score(y_train,y_pred2)
print('R2 score',r2_3)
```

```
R2 score 0.8101387702123997
R2 score 0.00840130249887483
R2 score 0.75349577173606
```

In [ ]:

In [ ]: