In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm,ttest_1samp,ttest_ind,ttest_rel
from statsmodels.stats.weightstats import ztest
from bioinfokit.analys import stat
from scipy import stats
from scipy.stats import t
from scipy.stats import chisquare,f_oneway,kruskal
from scipy.stats import ttest_ind_from_stats # Takes sample means, std, n and returns stat and p
from scipy.stats import chi2
import statistics
```

In [2]:

```
df = pd.read_csv("delhivery_data.csv")
df.head()
```

Out[2]:

tual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	segment_actual_time	segment_osrm_time	segment_osrm_distance
10.435660	14.0	11.0	11.9653	1.272727	14.0	11.0	11.9653
18.936842	24.0	20.0	21.7243	1.200000	10.0	9.0	9.7590
27.637279	40.0	28.0	32.5395	1.428571	16.0	7.0	10.8152
36.118028	62.0	40.0	45.5620	1.550000	21.0	12.0	13.0224
39.386040	68.0	44.0	54.2181	1.545455	6.0	5.0	3.9153

In [4]:

```
df.isna().sum() #since we have source center and destination center we have ignore the null values in source and destination names
```

Out[4]:

```
data                            0
trip_creation_time              0
route_schedule_uuid             0
route_type                      0
trip_uuid                       0
source_center                   0
source_name                   293
destination_center              0
destination_name              261
od_start_time                   0
od_end_time                     0
start_scan_to_end_scan          0
is_cutoff                       0
cutoff_factor                   0
cutoff_timestamp                0
actual_distance_to_destination  0
actual_time                     0
osrm_time                       0
osrm_distance                   0
factor                          0
segment_actual_time             0
segment_osrm_time               0
segment_osrm_distance           0
segment_factor                  0
dtype: int64
```

In [5]:

```
df.shape
```

Out[5]:

```
(144867, 24)
```

In [6]:

```
df.size
```

Out[6]:

```
3476808
```

In [7]:

```
df.dtypes
```

Out[7]:

```
data                                object
trip_creation_time                  object
route_schedule_uuid                 object
route_type                          object
trip_uuid                           object
source_center                       object
source_name                         object
destination_center                  object
destination_name                    object
od_start_time                       object
od_end_time                         object
start_scan_to_end_scan             float64
is_cutoff                             bool
cutoff_factor                        int64
cutoff_timestamp                    object
actual_distance_to_destination     float64
actual_time                        float64
osrm_time                          float64
osrm_distance                      float64
factor                             float64
segment_actual_time                float64
segment_osrm_time                  float64
segment_osrm_distance              float64
segment_factor                     float64
dtype: object
```

In [8]:

```
df.describe()
```

Out[8]:

	start_scan_to_end_scan	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	segi
count	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	144867.000000	
mean	961.262986	232.926567	234.073372	416.927527	213.868272	284.771297	2.120107	
std	1037.012769	344.755577	344.990009	598.103621	308.011085	421.119294	1.715421	
min	20.000000	9.000000	9.000045	9.000000	6.000000	9.008200	0.144000	
25%	161.000000	22.000000	23.355874	51.000000	27.000000	29.914700	1.604264	
50%	449.000000	66.000000	66.126571	132.000000	64.000000	78.525800	1.857143	
75%	1634.000000	286.000000	286.708875	513.000000	257.000000	343.193250	2.213483	
max	7898.000000	1927.000000	1927.447705	4532.000000	1686.000000	2326.199100	77.387097	

In [9]:

```
df.describe(include = ['object'])
```

Out[9]:

	data	trip_creation_time	route_schedule_uuid	route_type	trip_uuid	source_center	source_name	destination_ce
count	144867	144867	144867	144867	144867	144867	144574	144
unique	2	14817	1504	2	14817	1508	1498	1
top	training	2018-09-28 05:23:15.359220	thanos::sroute:4029a8a2-6c74-4b7e-a6d8-f9e069f...	FTL	trip-153811219535896559	IND000000ACB	Gurgaon_Bilaspur_HB (Haryana)	IND000000A
freq	104858	101	1812	99660	101	23347	23347	15

In [12]:

```
df.groupby(by = ['trip_uuid','source_center','destination_center']).sum()
```

Out[12]:

ual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	segment_actual_time	segment_osrm_time	segment_osrm_distance
3778.765471	6484.0	3464.0	4540.1261	31.355359	728.0	534.0	670.6205
5082.046634	9198.0	4323.0	6037.6386	45.164741	820.0	474.0	649.8528
53.310332	96.0	55.0	60.3157	5.239271	46.0	26.0	28.1995
186.897974	303.0	155.0	209.1151	11.255861	95.0	39.0	55.9899
1725.590250	2601.0	1427.0	1975.7409	20.854778	608.0	231.0	317.7408
...	...	...	...	...	...	...	...
88.326510	119.0	106.0	106.7084	4.234812	49.0	42.0	42.1431
90.049767	173.0	108.0	111.8555	5.936254	89.0	77.0	78.5869
21.672374	51.0	22.0	25.5371	4.767857	29.0	14.0	16.0184
62.547507	278.0	59.0	76.5169	8.194678	233.0	42.0	52.5303
47.691610	72.0	47.0	51.2851	3.043956	41.0	25.0	28.0484

In [13]:

```
df.groupby(by = ['trip_uuid','source_center','destination_center']).cumsum()
```

Out[13]:

	start_scan_to_end_scan	is_cutoff	cutoff_factor	actual_distance_to_destination	actual_time	osrm_time	osrm_distance	factor	segm
0	86.0	1	9	10.435660	14.0	11.0	11.9653	1.272727	
1	172.0	2	27	29.372503	38.0	31.0	33.6896	2.472727	
2	258.0	3	54	57.009782	78.0	59.0	66.2291	3.901299	
3	344.0	4	90	93.127810	140.0	99.0	111.7911	5.451299	
4	430.0	4	129	132.513850	208.0	143.0	166.0092	6.996753	
...	...	...	...	...	...	...	...	...	
144862	2135.0	5	135	140.930220	345.0	230.0	228.5453	7.786162	
144863	2562.0	6	189	195.022752	465.0	306.0	314.2282	9.365109	
144864	2989.0	7	252	261.186343	605.0	394.0	411.3215	10.956018	
144865	3416.0	8	324	334.867011	763.0	492.0	522.5924	12.568263	
144866	3843.0	8	394	404.906021	1189.0	587.0	611.3243	17.052474	

144867 rows × 12 columns

In [22]:

```
df = df.groupby(by = ['trip_uuid']).first()
```

In [23]:

```
df[['Dest_Address','Dest_State']] = df['destination_name'].str.split(' ',expand=True).drop([2,3,4],axis=1)
df[['dest_city','dest_place','dest_code']]=df['Dest_Address'].str.split('_',expand=True).drop(3,axis=1)
df = df.drop('Dest_Address',axis=1)
```

In [24]:

```
df[['src_Address','src_State']] = df['source_name'].str.split(' ',expand=True).drop([2,3,4],axis=1)
df[['src_city','src_place','src_code']]=df['src_Address'].str.split('_',expand=True).drop(3,axis=1)
df = df.drop('src_Address',axis=1)
```

In [25]:

```
df['od_start_time'] = pd.to_datetime(df['od_start_time'], format='%Y-%m-%d %H:%M:%S.%f')
df['od_end_time'] = pd.to_datetime(df['od_end_time'], format='%Y-%m-%d %H:%M:%S.%f')
```

In [26]:

```
df['time_diff_hrs'] = (df['od_end_time']-df['od_start_time']).astype('timedelta64[h]')
```

In [27]:

```
## H0: mean1 = mean2
## Ha: mean1 != mean2
sample1 = df['osrm_time']
sample2 = df['actual_time']
stats.ttest_ind(sample1, sample2)#Reject
```

Out[27]:

```
Ttest_indResult(statistic=-33.58499436336649, pvalue=9.967075445227124e-243)
```

In [28]:

```
## H0: Same distribution
## Ha: Different Distribution
stats.ks_2samp(sample1, sample2)#Reject
```

Out[28]:

```
KstestResult(statistic=0.4579199568063711, pvalue=0.0)
```

In [29]:

```
sample1.describe()
```

Out[29]:

```
count    14817.000000
mean        18.656881
std         23.714105
min          6.000000
25%         11.000000
50%         16.000000
75%         22.000000
max       1611.000000
Name: osrm_time, dtype: float64
```

In [30]:

```
sample2.describe()
```

Out[30]:

```
count    14817.000000
mean        39.997975
std         73.623517
min          9.000000
25%         20.000000
50%         30.000000
75%         42.000000
max       3051.000000
Name: actual_time, dtype: float64
```

In [31]:

```
## H0: mean1 = mean2
## Ha: mean1 != mean2
sample1 = df['actual_time']
sample2 = df['segment_actual_time']
stats.ttest_ind(sample1, sample2)#fail to reject
```

Out[31]:

```
Ttest_indResult(statistic=0.0, pvalue=1.0)
```

In [32]:

```
## H0: Same distribution
## Ha: Different Distribution
stats.ks_2samp(sample1, sample2)#fail to reject
```

Out[32]:

```
KstestResult(statistic=0.0, pvalue=1.0)
```

In [33]:

```
## H0: mean1 = mean2
## Ha: mean1 != mean2
sample1 = df['segment_osrm_distance']
sample2 = df['osrm_distance']
stats.ttest_ind(sample1, sample2)#fail to reject
```

Out[33]:

```
Ttest_indResult(statistic=0.0, pvalue=1.0)
```

In [34]:

```
## H0: Same distribution
## Ha: Different Distribution
stats.ks_2samp(sample1, sample2)#fail to reject
```

Out[34]:

```
KstestResult(statistic=0.0, pvalue=1.0)
```

In [35]:

```
## H0: mean1 = mean2
## Ha: mean1 != mean2
sample1 = df['osrm_time']
sample2 = df['segment_osrm_time']
stats.ttest_ind(sample1, sample2)#fail to reject
```

Out[35]:

```
Ttest_indResult(statistic=0.0, pvalue=1.0)
```

In [36]:

```
## H0: Same distribution
## Ha: Different Distribution
stats.ks_2samp(sample1, sample2)#fail to reject
```

Out[36]:

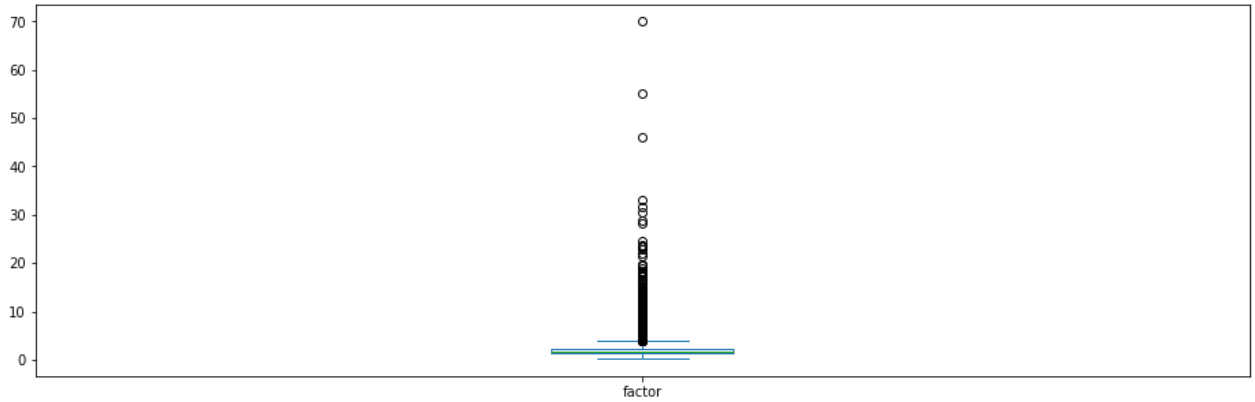
```
KstestResult(statistic=0.0, pvalue=1.0)
```

In [37]:

```
plt.figure()
df["factor"].plot.box(figsize=(16,5))#too many outliers
```

Out[37]:

<AxesSubplot:>



In [38]:

```
percentile25 = df['factor'].quantile(0.25)
percentile75 = df['factor'].quantile(0.98)
iqr = percentile75 - percentile25
```

In [39]:

```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
df.shape
```

Out[39]:

(14817, 32)

In [40]:

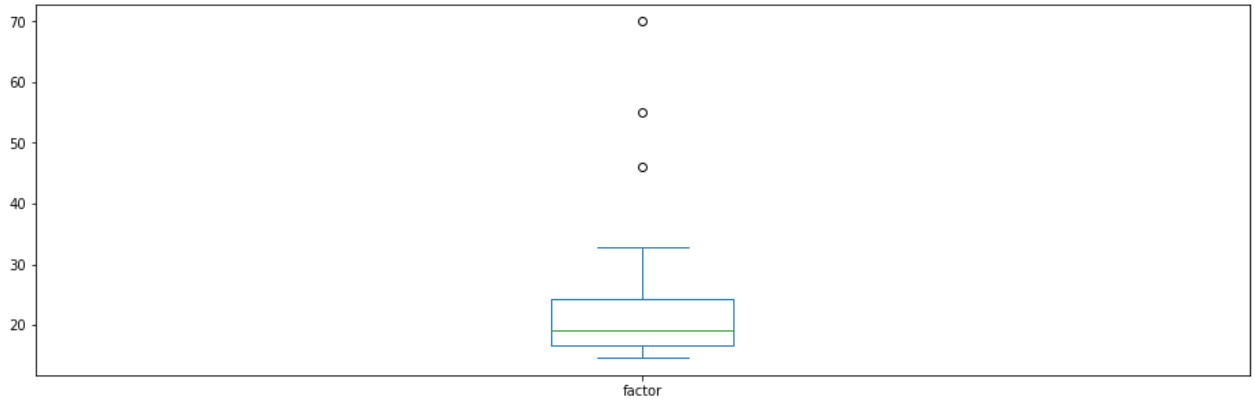
```
df = df[df['factor']>upper_limit]
```

In [41]:

```
plt.figure()
df['factor'].plot.box(figsize=(16,5))
```

Out[41]:

<AxesSubplot:>



In [ ]:

```

```