In [1]:

```
import pandas as pd
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
df = pd.read_csv("1rec-crime-pfa.csv", parse_dates=["12 months ending"])
df
```

Out[1]:

	12 months ending	PFA	Region	Offence	Rolling year total number of offences
0	2003-03-31	Avon and Somerset	South West	All other theft offences	25959
1	2003-03-31	Avon and Somerset	South West	Bicycle theft	3090
2	2003-03-31	Avon and Somerset	South West	Criminal damage and arson	26202
3	2003-03-31	Avon and Somerset	South West	Death or serious injury caused by illegal driving	2
4	2003-03-31	Avon and Somerset	South West	Domestic burglary	14561
...	...	...	...	...	...
46464	2018-12-31	Wiltshire	South West	Stalking and harassment	2380
46465	2018-12-31	Wiltshire	South West	Theft from the person	347
46466	2018-12-31	Wiltshire	South West	Vehicle offences	2895
46467	2018-12-31	Wiltshire	South West	Violence with injury	5701
46468	2018-12-31	Wiltshire	South West	Violence without injury	5840

46469 rows × 5 columns

In [2]:

```
dfch = pd.read_excel("population.xlsx", parse_dates=["12 months ending"])
df_new = pd.merge(df, dfch, on=("Region", "12 months ending"))
df_new["number of offences per 1000 people"] = df_new["Rolling year total number of offences"]/df_new["Population"] * 1000
df_new
```

Out[2]:

	12 months ending	PFA	Region	Offence	Rolling year total number of offences	Population	number of offences per 1000 people
0	2003-03-31	Avon and Somerset	South West	All other theft offences	25959	4991000.0	5.201162
1	2003-03-31	Avon and Somerset	South West	Bicycle theft	3090	4991000.0	0.619114
2	2003-03-31	Avon and Somerset	South West	Criminal damage and arson	26202	4991000.0	5.249850
3	2003-03-31	Avon and Somerset	South West	Death or serious injury caused by illegal driving	2	4991000.0	0.000401
4	2003-03-31	Avon and Somerset	South West	Domestic burglary	14561	4991000.0	2.917451
...	...	...	...	...	...	...	...
44414	2018-12-31	West Midlands	West Midlands	Stalking and harassment	15002	5873003.0	2.554400
44415	2018-12-31	West Midlands	West Midlands	Theft from the person	3230	5873003.0	0.549974
44416	2018-12-31	West Midlands	West Midlands	Vehicle offences	37250	5873003.0	6.342581
44417	2018-12-31	West Midlands	West Midlands	Violence with injury	30561	5873003.0	5.203641
44418	2018-12-31	West Midlands	West Midlands	Violence without injury	24861	5873003.0	4.233098

44419 rows × 7 columns

1 Гипотеза: Удаление данных организаций не сильно отразится на общую картину распределения преступлений. Сначала покажем общее распределение преступлений с течением времени до обработки.

In [3]:

```
time_offence = df.groupby("12 months ending").sum()["Rolling year total number of offences"]
fig_time_offence = px.line(time_offence, y="Rolling year total number of offences")
fig_time_offence.show()
```

После удаления неудобных данных распределение приняло следующий вид:

In [4]:

```
time_offence_new = df_new.groupby("12 months ending").sum()["Rolling year total number of offences"]
fig_time_offence_new = px.line(time_offence_new, y="Rolling year total number of offences")
fig_time_offence_new.show()
```

По всей видимости, мы нашли объяснение двойному минимуму распределения преступлений. Он появлялся только из-за внезапного появления в середине 2011 года новых строк с данными от частных организаций. Однако возрастание преступности, начиная с 2014 года никуда не делся. В целом можно заключить, что гипотеза почти подтвердилась.

2 Гипотеза: Нормировка преступлений гораздо точнее покажет опасность отдельных районов. Сначала покажем общее распределение преступлений по регионам до обработки.

In [5]:

```
Region_offence = df_new.groupby("Region").sum()["Rolling year total number of offences"].sort_values()
fig_Region_offence = px.bar(Region_offence, y='Rolling year total number of offences')
fig_Region_offence.show()
```

Теперь покажем распределение преступлений с учётом количества жителей в регионах.

In [6]:

```
Region_offence_new = df_new.groupby("Region").sum()["number of offences per 1000 people"].sort_values()
fig_Region_offence_new = px.bar(Region_offence_new, y='number of offences per 1000 people')
fig_Region_offence_new.show()
```

Хоть Лондон и оставил за собой криминальное лидерство, можно заметить существенные изменения в распределении преступлений. Так Северо-Восточный округ с последнего 10 места переместился аж на 4, а Юго-Восточный со 2 на 8 место. Но, что самое главное, теперь распределение приблизилось к равномерному, то есть на самом деле в стране нет такого сильного криминогенного перекоса. Гипотеза полностью подтвердилась.

3 Гипотеза: В зоне ответственности столичной полиции криминальная обстановка не настолько сильно отличается относительно других районов, если сделать поправку на количество жителей, которое они обхватывают. Покажем, какое распределение мы видели во 2 задании.

In [7]:

```
PFA_offence = df_new.groupby("PFA").sum()["Rolling year total number of offences"].sort_values()
fig_PFA_offence = px.bar(PFA_offence, y='Rolling year total number of offences')
fig_PFA_offence.show()
```

С поправкой на население распределение принимает следующий вид:

In [8]:

```
PFA_offence_new = df_new.groupby("PFA").sum()["number of offences per 1000 people"].sort_values()
fig_PFA_offence_new = px.bar(PFA_offence_new, y='number of offences per 1000 people')
fig_PFA_offence_new.show()
```

Видим, что некоторые районы сместились, например Northumbria с 15 места поднялась на 3. Но общая картина распределения практически не изменилась и столичная полиция лидирует с большим отрывом. Следовательно гипотеза не подтвердилась.

Гипотеза 4. У разных полицейских участков сильно отличается количество подкотрольных им регионов. Так как полицейских участков намного больше, чем регионов, корректнее было бы сформулировать гипотезу наоборот: в разных регионах количество полицейских отделов сильно разнится.

In [9]:

```
for i in df_new.Region.unique():
  print(i,
      df_new["PFA"].loc[df_new["Region"] == i].unique())
```

```
South West ['Avon and Somerset' 'Devon and Cornwall' 'Dorset' 'Gloucestershire'
 'Wiltshire']
East ['Bedfordshire' 'Cambridgeshire' 'Essex' 'Hertfordshire' 'Norfolk'
 'Suffolk']
North West ['Cheshire' 'Cumbria' 'Greater Manchester' 'Lancashire' 'Merseyside']
London ['City of London' 'Metropolitan Police']
North East ['Cleveland' 'Durham' 'Northumbria']
East Midlands ['Derbyshire' 'Leicestershire' 'Lincolnshire' 'Northamptonshire'
 'Nottinghamshire']
Wales ['Dyfed-Powys' 'Gwent' 'North Wales' 'South Wales']
South East ['Hampshire' 'Kent' 'Surrey' 'Sussex' 'Thames Valley']
Yorkshire and The Humber ['Humberside' 'North Yorkshire' 'South Yorkshire' 'West Yorkshire']
West Midlands ['Staffordshire' 'Warwickshire' 'West Mercia' 'West Midlands']
```

Можно, конечно, отобразить, как было изначально сформулировано, но так менее наглядно. Зато отчётливо видно, что ни один полицейский участок не дежурит сразу в нескольких регионах страны.

In [10]:

```
for i in df_new.PFA.unique():
  print(i,
      df_new["Region"].loc[df_new["PFA"] == i].unique())
```

```
Avon and Somerset ['South West']
Devon and Cornwall ['South West']
Dorset ['South West']
Gloucestershire ['South West']
Wiltshire ['South West']
Bedfordshire ['East']
Cambridgeshire ['East']
Essex ['East']
Hertfordshire ['East']
Norfolk ['East']
Suffolk ['East']
Cheshire ['North West']
Cumbria ['North West']
Greater Manchester ['North West']
Lancashire ['North West']
Merseyside ['North West']
City of London ['London']
Metropolitan Police ['London']
Cleveland ['North East']
Durham ['North East']
Northumbria ['North East']
Derbyshire ['East Midlands']
Leicestershire ['East Midlands']
Lincolnshire ['East Midlands']
Northamptonshire ['East Midlands']
Nottinghamshire ['East Midlands']
Dyfed-Powys ['Wales']
Gwent ['Wales']
North Wales ['Wales']
South Wales ['Wales']
Hampshire ['South East']
Kent ['South East']
Surrey ['South East']
Sussex ['South East']
Thames Valley ['South East']
Humberside ['Yorkshire and The Humber']
North Yorkshire ['Yorkshire and The Humber']
South Yorkshire ['Yorkshire and The Humber']
West Yorkshire ['Yorkshire and The Humber']
Staffordshire ['West Midlands']
Warwickshire ['West Midlands']
West Mercia ['West Midlands']
West Midlands ['West Midlands']
```

Видим, что количество полицейских отделов разнится от 2 в Лондоне до 6 в Восточном округе. Из этой картины мы понимаем, почему на графике преступлений по районам столичная полиция имеет такой выброс. Там всего 2 участка, 1 из которых вообще почти не имеет (или не выкладывает) преступлений. Получается всего один участок на самый крупный регион страны. Можно заключить, что гипотеза скорее подтвердилась.

5 Гипотеза: возможно, уровень преступности как-то коррелирует с широтами, в которых находится регион, например, чем южнее, тем больше в среднем совершается в год преступлений.

In [11]:

```
from urllib.request import urlopen
import json
with urlopen('https://martinjc.github.io/UK-GeoJSON/json/eng/topo_eer.json') as response:
    UK = json.load(response)
UK['objects']['eer']['geometries'][0]['properties']
```

Out[11]:

```
{'EER13CD': 'E15000001', 'EER13CDO': '01', 'EER13NM': 'North East'}
```

In [25]:

```
fig = px.choropleth(df_new, geojson=UK, locations='Region',
                    featureidkey="properties.EER13NM",
                    color='number of offences per 1000 people',
                           color_continuous_scale="tealrose",
                           scope = "europe",
                           range_color=(0, 10),
                           labels={'number':'number of offences per 1000 people'},
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

In [23]:

```
Region_offence_new
```

Out[23]:

```
Region
South West                  3321.075610
East                        3321.381941
South East                  3501.424042
Wales                       3565.196442
West Midlands               3678.548996
East Midlands               3743.515374
North East                  3842.195421
North West                  4190.284434
Yorkshire and The Humber    4405.745890
London                      5122.507825
Name: number of offences per 1000 people, dtype: float64
```

In [26]:

```
fig = px.choropleth(Region_offence_new , geojson=UK, locations='Region',
                    featureidkey="properties.EER13NM",
                    color='number of offences per 1000 people',
                           color_continuous_scale="tealrose",
                           scope = "europe",
                           range_color=(3000, 5000),
                           labels={'number':'number of offences per 1000 people'},
                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-26-b7e5d741d8d2> in <module>
----> 1 fig = px.choropleth(Region_offence_new , geojson=UK, locations='Region',
      2                     featureidkey="properties.EER13NM",
      3                     color='number of offences per 1000 people',
      4                           color_continuous_scale="tealrose",
      5                           scope = "europe",

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_chart_types.py in choropleth(data_frame, lat, lon, locations, locationmode, geojson, featureidkey, color, facet_row, facet_col, facet_col_wrap, facet_row_spacing, facet_col_spacing, hover_name, hover_data, custom_data, animation_frame, animation_group, category_orders, labels, color_discrete_sequence, color_discrete_map, color_continuous_scale, range_color, color_continuous_midpoint, projection, scope, center, fitbounds, basemap_visible, title, template, width, height)
    972     colored region mark on a map.
    973     """
--> 974     return make_figure(
    975         args=locals(),
    976         constructor=go.Choropleth,

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in make_figure(args, constructor, trace_patch, layout_patch)
   1859     apply_default_cascade(args)
   1860 
-> 1861     args = build_dataframe(args, constructor)
   1862     if constructor in [go.Treemap, go.Sunburst] and args["path"] is not None:
   1863         args = process_dataframe_hierarchy(args)

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in build_dataframe(args, constructor)
   1375     # now that things have been prepped, we do the systematic rewriting of `args`
   1376 
-> 1377     df_output, wide_id_vars = process_args_into_dataframe(
   1378         args, wide_mode, var_name, value_name
   1379     )

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in process_args_into_dataframe(args, wide_mode, var_name, value_name)
   1181                     if argument == "index":
   1182                         err_msg += "\n To use the index, pass it in directly as `df.index`."
-> 1183                     raise ValueError(err_msg)
   1184             elif length and len(df_input[argument]) != length:
   1185                 raise ValueError(

ValueError: Value of 'locations' is not the name of a column in 'data_frame'. Expected one of ['number of offences per 1000 people'] but received: Region
```

In [27]:

```
fig = px.choropleth(Region_offence_new , geojson=UK,
                    featureidkey="properties.EER13NM",
                    color='number of offences per 1000 people',
                          color_continuous_scale="tealrose",
                          scope = "europe",
                          range_color=(3000, 5000),
                          labels={'number':'number of offences per 1000 people'},
                         )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

In [39]:

```
df_Region_offence_new  = pd.DataFrame(Region_offence_new)
df_Region_offence_new
```

Out[39]:

	number of offences per 1000 people
Region	
South West	3321.075610
East	3321.381941
South East	3501.424042
Wales	3565.196442
West Midlands	3678.548996
East Midlands	3743.515374
North East	3842.195421
North West	4190.284434
Yorkshire and The Humber	4405.745890
London	5122.507825

In [31]:

```
fig = px.choropleth(Region_offence_new , geojson=UK, locations='Region',
                    featureidkey="properties.EER13NM",
                    color='number of offences per 1000 people',
                          color_continuous_scale="tealrose",
                          scope = "europe",
                          range_color=(3000, 5000),
                          labels={'number':'number of offences per 1000 people'},
                         )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-b7e5d741d8d2> in <module>
----> 1 fig = px.choropleth(Region_offence_new , geojson=UK, locations='Region',
      2                     featureidkey="properties.EER13NM",
      3                     color='number of offences per 1000 people',
      4                           color_continuous_scale="tealrose",
      5                           scope = "europe",

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_chart_types.py in choropleth(data_frame, lat, lon, loca
tions, locationmode, geojson, featureidkey, color, facet_row, facet_col, facet_col_wrap, facet_row_spacing, facet_col_spacing, hover_name, hover
_data, custom_data, animation_frame, animation_group, category_orders, labels, color_discrete_sequence, color_discrete_map, color_continuou
s_scale, range_color, color_continuous_midpoint, projection, scope, center, fitbounds, basemap_visible, title, template, width, height)
    972     colored region mark on a map.
    973     """
--> 974     return make_figure(
    975         args=locals(),
    976         constructor=go.Choropleth,

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in make_figure(args, constructor, trace_patch
, layout_patch)
   1859     apply_default_cascade(args)
   1860 
-> 1861     args = build_dataframe(args, constructor)
   1862     if constructor in [go.Treemap, go.Sunburst] and args["path"] is not None:
   1863         args = process_dataframe_hierarchy(args)

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in build_dataframe(args, constructor)
   1375     # now that things have been prepped, we do the systematic rewriting of `args`
   1376 
-> 1377     df_output, wide_id_vars = process_args_into_dataframe(
   1378         args, wide_mode, var_name, value_name
   1379     )

c:\users\vanya\appdata\local\programs\python\python38-32\lib\site-packages\plotly\express\_core.py in process_args_into_dataframe(args, wide_m
ode, var_name, value_name)
   1181                         if argument == "index":
   1182                             err_msg += "\n To use the index, pass it in directly as `df.index`."
-> 1183                         raise ValueError(err_msg)
   1184                 elif length and len(df_input[argument]) != length:
   1185                     raise ValueError(

ValueError: Value of 'locations' is not the name of a column in 'data_frame'. Expected one of ['number of offences per 1000 people'] but received: Region
```

In [ ]: