Университет ИТМО

Практическая работа №4

по дисциплине «Визуализация и моделирование»

Автор: Власов Матвей Иванович
Поток: ВИМ 1.1
Группа: K3240
Факультет: ИКТ
Преподаватель: Чернышева А. В.

Санкт-Петербург, 2021 г.

Датасет

Для дальнейшей работы выбран датасет: "Russian Presidental Elections 2018 Voting Data": https://www.kaggle.com/valenzione/russian-presidental-elections-2018-voting-data

Описание датасета

В нашем датасете содержится информация об итогах выборов 2018 года, полученная с официального сайта ЦИК РФ.

Названия большинства стобцов исходного датасета представим в графе "Описание а сами названия сократим для удобства в дальнейшем:

Столбец	Описание	Тип	Шкала	Предобработка
PS_ID	Идентификатор избирательного участка	INT	Номинальная	Не требуется
REGION	Название региона	STRING	Номинальная	Убрать цифры в начале строки (если есть)
SUBREGION	Название округа	STRING	Номинальная	Убрать цифры в начале строки (если есть)
N_ALL	Число избирателей, включенных в список избирателей	INT	Относительная	Не требуется
N_GIVEN	Число избирательных бюллетеней, полученных участковой избирательной комиссией	INT	Относительная	Удалить после подсчёта N_VOTED
N_EARLY	Число избирательных бюллетеней, выданных избирателям, проголосовавшим досрочно	INT	Относительная	Не требуется
N_IN	Число избирательных бюллетеней, выданных в помещении для голосования в день голосования	INT	Относительная	Удалить (избыточные данные)
N_OUT	Число избирательных бюллетеней, выданных вне помещения для голосования в день голосования	INT	Относительная	Не требуется
N_LEFT	Число погашенных избирательных бюллетеней	INT	Относительная	Удалить после подсчёта N_VOTED
N_PORTABLE	Число избирательных бюллетеней в переносных ящиках для голосования	INT	Относительная	Удалить (избыточные данные)
N_STATIC	Число бюллетеней в стационарных ящиках для голосования	INT	Относительная	Удалить (избыточные данные)
N_INVALID	Число недействительных избирательных бюллетеней	INT	Относительная	Не требуется
N_VALID	Число действительных избирательных бюллетеней	INT	Относительная	Удалить (избыточные данные)
N_LOST	Число утраченных избирательных бюллетеней	INT	Относительная	Удалить (незначительные данные)
N_UNUSED	Число избирательных бюллетеней, не учтенных при получении	INT	Относительная	Удалить (незначительные данные)

Столбец	Описание	Тип	Шкала	Предобработка
BABURIN	Бабурин Сергей Николаевич	INT	Относительная	Не требуется
GRUDININ	Грудинин Павел Николаевич	INT	Относительная	Не требуется
ZHIRINOVSKY	Жириновский Владимир Вольфович	INT	Относительная	Не требуется
PUTIN	Путин Владимир Владимирович	INT	Относительная	Не требуется
SOBCHAK	Собчак Ксения Анатольевна	INT	Относительная	Не требуется
SURAYKIN	Сурайкин Максим Александрович	INT	Относительная	Не требуется
TITOV	Титов Борис Юрьевич	INT	Относительная	Не требуется
YAVLINSKY	Явлинский Григорий Алексеевич	INT	Относительная	Не требуется

Задачи, решаемые при помощи датасета

1. Визуализация результатов выборов.
2. Анализ данных на предмет возможных фальсификаций.
3. Выявление особенностей голосования в различных регионах.

Гипотезы

1. В Москве и Санкт-Петербурге ниже, чем в среднем, и процент за Путина, и явка (в больших городах более образованное население, а также большое количество наблюдателей, что затрудняет фальсификации).
2. В Крыму высокая явка и поддержка президента (из-за присоединения территории).
3. В регионах с большим количеством избирателей, проголосовавших досрочно, процент за Путина выше, чем в среднем (голоса, поданные досрочно, легче сфальсифицировать).
4. Есть регионы, где победил не Путин (у Грудинина в среднем больше 11 процентов - вполне возможно, что где-то он набрал больше Путина).

Работа с датасетом

Обратите внимание, что часть кода и таблиц отображается неполностью.
Для просмотра недостающей информации откройте файл с исходным кодом.

Отметим, что данные нашего датасета немного не совпадают с данными на сайте ЦИК.\ Расхождения замечены в нескольких регионах. Как мы это учтём:\

1. При анализе выборов по регионам заменим часть датасета на данные с сайта ЦИК там, где замечены несовпадения.\
2. При дальнейшем анализе данных регионов по участкам будем иметь в виду, что наша информация немного неполная/неактуальная.

In [1]:
```
import folium
import json

import io
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns


from PIL import Image
from wordcloud import WordCloud
```

In [2]:
```
df = pd.read_csv('../voting_data.csv')
df
```

Out[2]:

	PS_ID	REGION	SUBREGION	N_ALL	N_GIVEN	N_EARLY	N_IN	N_OUT	N_LEI
0	8140	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2132	2000	0	1447	11	5
1	8141	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2207	2000	0	1470	14	5
2	8142	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2249	2000	0	1490	7	5
3	8143	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	1769	1500	0	1065	48	3
4	8144	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	1880	1500	0	1171	13	3
...	...	...	...	...	...	...	...	...	
97689	1310	Ямало-Ненецкий автономный округ	13 Ямальская	2892	2900	1000	1891	0	
97690	1311	Ямало-Ненецкий автономный округ	13 Ямальская	2867	2900	921	1941	0	
97691	1312	Ямало-Ненецкий	13 Ямальская	2895	2900	927	1964	0	

		автономный округ							
97692	1313	Ямало-Ненецкий автономный округ	13 Ямальская	3397	3450	1103	2281	0	
97693	1314	Ямало-Ненецкий автономный округ	13 Ямальская	2666	3000	1022	1632	0	3

97694 rows × 23 columns

Добавим в таблицу информацию об общем количестве голосов на участке

In [3]:
```
voted_list = []
turnout_list = []

for _, row in df.iterrows():
    voted = row['N_EARLY'] + row['N_IN'] + row['N_OUT']
    voted_list.append(voted) # or N_GIVEN - N_LEFT
    turnout_list.append(voted/row['N_ALL'])

df['N_VOTED'] = voted_list
df['TURNOUT'] = turnout_list
df
```

Out[3]:

	PS_ID	REGION	SUBREGION	N_ALL	N_GIVEN	N_EARLY	N_IN	N_OUT	N_LEI
0	8140	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2132	2000	0	1447	11	5
1	8141	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2207	2000	0	1470	14	5
2	8142	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	2249	2000	0	1490	7	5
3	8143	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	1769	1500	0	1065	48	3
4	8144	98 Город Байконур (Республика Казахстан)	98 Город Байконур (Республика Казахстан)	1880	1500	0	1171	13	3
...	...	...	...	...	...	...	...	...	
97689	1310	Ямало-Ненецкий автономный округ	13 Ямальская	2892	2900	1000	1891	0	
97690	1311	Ямало-Ненецкий автономный округ	13 Ямальская	2867	2900	921	1941	0	
97691	1312	Ямало-Ненецкий	13 Ямальская	2895	2900	927	1964	0	

		автономный округ							
97692	1313	Ямало-Ненецкий автономный округ	13 Ямальская	3397	3450	1103	2281	0	
97693	1314	Ямало-Ненецкий автономный округ	13 Ямальская	2666	3000	1022	1632	0	3

97694 rows × 25 columns

Посмотрим, какие столбцы у нас есть

In [4]:
```
list(df.columns)
```

Out[4]:
```
['PS_ID',
 'REGION',
 'SUBREGION',
 'N_ALL',
 'N_GIVEN',
 'N_EARLY',
 'N_IN',
 'N_OUT',
 'N_LEFT',
 'N_PORTABLE',
 'N_STATIC',
 'N_INVALID',
 'N_VALID',
 'N_LOST',
 'N_UNUSED',
 'BABURIN',
 'GRUDININ',
 'ZHIRINOVSKY',
 'PUTIN',
 'SOBCHAK',
 'SURAYKIN',
 'TITOV',
 'YAVLINSKY',
 'N_VOTED',
 'TURNOUT']
```

Заметим, что в таблице много столбцов с избыточными/ненужными данными, а именно:\ N_GIVEN, N_LEFT - могут использоваться только для подсчёта N_VOTED, что мы уже сделали\ N_IN - вряд ли будем использовать, но в случае необходимости посчитаем как N_VOTED - N_OUT - N_EARLY\ N_PORTABLE, N_STATIC - почти полностью совпадают с N_OUT и N_IN, нет необходимости их хранения\ N_VALID - можно посчитать как N_VOTED - N_INVALID\ N_UNUSED, N_LOST - почти всегда равны 0 (см. ниже) и не влияют на общую картину

In [5]:
```
df['N_UNUSED'].sum()
```

Out[5]:
```
104
```

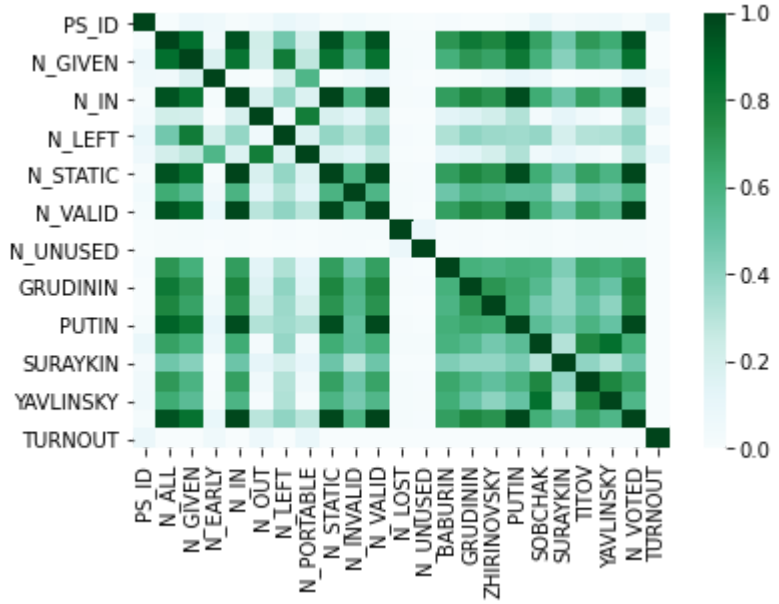
In [6]:
```
df['N_LOST'].sum()
```

Out[6]:
```
1047
```

Покажем корреляцию столбцов

In [7]:
```
corr = df.corr()
sns.heatmap(corr, cmap="BuGn", vmin=0)
```

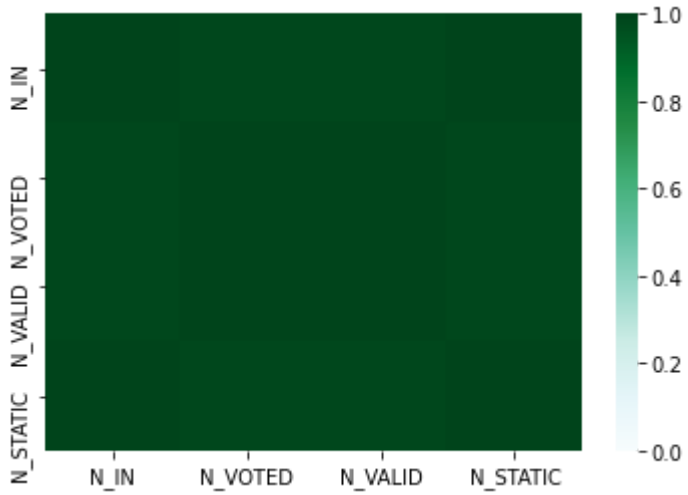
Out[7]: <AxesSubplot:>



Рассмотрим ближе интересующие нас стобцы

In [8]:
```
corr = df[['N_IN', 'N_VOTED', 'N_VALID', 'N_STATIC']].corr()
sns.heatmap(corr, cmap="BuGn", vmin=0)
```

Out[8]: <AxesSubplot:>



In [9]:
```
corr = df[['N_OUT', 'N_PORTABLE']].corr()
sns.heatmap(corr, cmap="BuGn", vmin=0)
```

Out[9]: <AxesSubplot:>

Удалим ненужные столбцы после создания таблицы с результатами выборов по регионам

У нас есть два региона, названия которых в датасете начинаются с цифры. Для удобства эти цифры удалим\

In [10]:
```
def remove_digits(text):
    if text[0].isdigit() and '№' not in text:
        temp = list(map(lambda x: '' if x.isdigit() else x, text))
        text = ''.join(temp).strip()
    return text
```

In [11]:
```
df['REGION'] = df['REGION'].apply(remove_digits)
df['SUBREGION'] = df['SUBREGION'].apply(remove_digits)
df['REGION'].unique()[:2]
```

Out[11]:
```
array(['Город Байконур (Республика Казахстан)',
       'Территория за пределами РФ'], dtype=object)
```

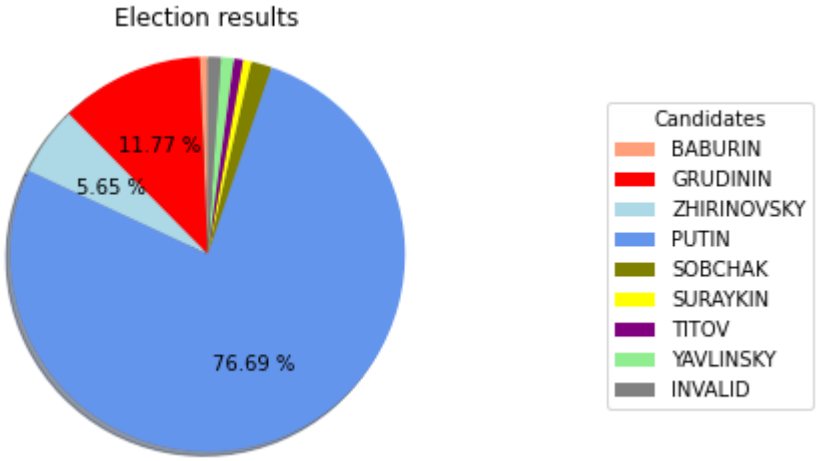
Объявим некоторые константы, которые пригодятся при анализе данных

In [12]:
```
start = list(df.columns).index('BABURIN')
end = list(df.columns).index('YAVLINSKY') + 1
CANDS = [i for i in df.columns[start:end]]
COLORS = ['lightsalmon', 'red', 'lightblue', 'CornflowerBlue', 'olive',
DF_LIST = [i for i in range(df.shape[0])]
INCOMPLETE_REGIONS = ['Ханты-Мансийский автономный округ - Югра', 'Респу
                      'Чувашская Республика - Чувашия', 'Брянская область', 'Во
                      'Калининградская область', 'Кемеровская область', 'Магада
                      'Нижегородская область', 'город Москва', 'Территория за п
```

Представим результаты выборов на круговой диаграмме

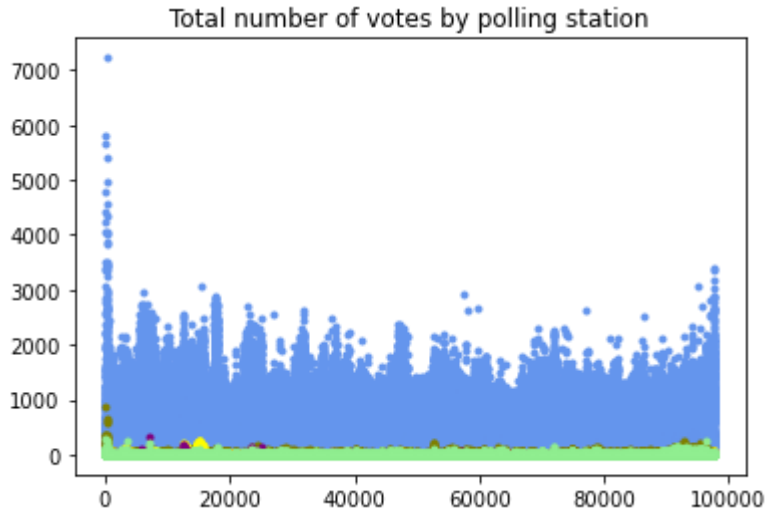
In [13]:
```
labels = CANDS.copy() + ['INVALID']
start = list(df.columns).index('BABURIN')
end = list(df.columns).index('YAVLINSKY') + 1
sizes = [df[i].sum() for i in df.columns[start:end]]
sizes.append(df['N_INVALID'].sum())

_, ax = plt.subplots()
ax.pie(sizes, labels=labels,
       autopct=(lambda x: f'{x:.2f} %' if x > 5.0 else ''),
       shadow=True, startangle=90, labeldistance=None,
       colors=COLORS + ['grey'])
ax.axis('equal')
ax.legend(labels, title="Candidates", loc="center", bbox_to_anchor=(1, 0
```

```
plt.title('Election results')
plt.show()
```



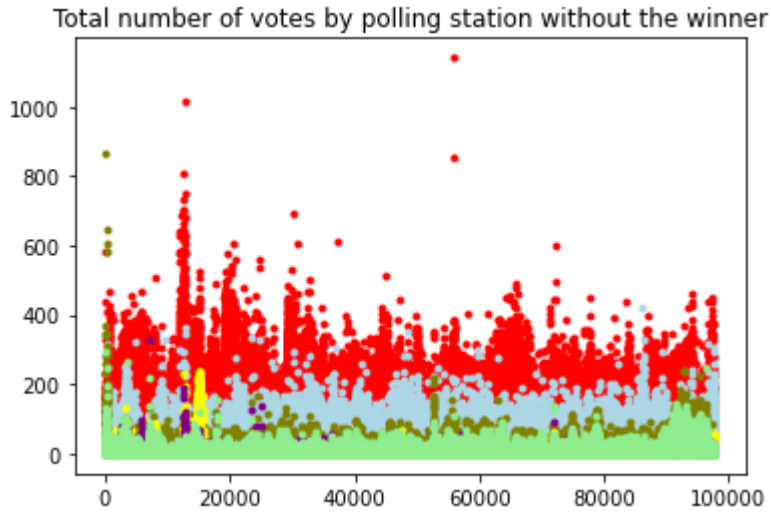
На диаграмме мы видим распределение голосов на выборах. Отметим, что показанные на диаграмме проценты соответствуют официальным данным (наши недостающие данных составляют слишком малую часть, поэтому на общие итоги не влияют)

In [14]:
```
for k in range(len(CANDS)):
    plt.plot(DF_LIST, [df[CANDS[k]][i] for i in range(df.shape[0])], '.'

plt.title('Total number of votes by polling station')
plt.show()
```



На диаграмме выше представлено количество голосов за каждого кандидата по УИК. Как мы видим, за Путина в среднем голосовало по несколько тысяч человек на участке, в то время как за других кандидатов - явно меньше тысячи. Чтобы подробнее увидеть информацию о других кандидатах, уберём голоса за победителя

In [15]:
```
for k in range(len(CANDS)):
    if k == CANDS.index('PUTIN'): continue
    plt.plot(DF_LIST, [df[CANDS[k]][i] for i in range(df.shape[0])], '.'

plt.title('Total number of votes by polling station without the winner')
plt.show()
```



По данной диаграмме видно, что на втором месте с достаточным отрывов находится Грудинин. В районе 17000-х участков можно заметить резкий рост голосов за Сурайкина. Возможно, в дальнейшем остановимся на этом подробнее\

Заметим, что по анализу отдельных УИК тяжело сделать какие-либо выводы, кроме самых очевидных. Будем проводить анализ по регионам

Создадим таблицу с данным по регионам. Для этого просуммируем данные из всех участков данного региона и запишем их в одну строку. Для регионов, где в первой таблице содержатся неполные данные, возьмём данные с сайта ЦИК

In [16]:
```
try:
    df_regions = pd.read_csv('regions_data.csv', index_col=0)
except FileNotFoundError:
    df_regions = pd.DataFrame(columns=df.columns)
    df_regions = df_regions.drop(columns=['PS_ID', 'SUBREGION', 'N_VOTED

    for region in df['REGION'].unique():
        info = {}
        new_region = remove_digits(region)

        info['REGION'] = new_region
        for col in df.columns[3:-2]:
            info[col] = df[df['REGION'] == region][col].sum()
        df_regions = df_regions.append(info, ignore_index=True)

    for region in INCOMPLETE_REGIONS:
        new_info = []
        data = pd.read_csv('../' + region + '.csv')
        for row in data['Unnamed: 2']:
            try:
                new_info.append(int(row))
            except ValueError:
                continue

        index = int(df_regions[df_regions['REGION'] == region].index.val
        for i, col in enumerate(df_regions.columns[1:]):
            df_regions.at[index, col] = new_info[i+2]

df_regions
```

Out[16]:

	REGION	N_ALL	N_GIVEN	N_EARLY	N_IN	N_OUT	N_LEFT	N_PORTABLE
0	Город Байконур	14575	12800	0	9491	168	3141	168

	(Республика Казахстан)							
1	Территория за пределами РФ	483957	1408383	53482	403108	18026	933738	71484
2	Республика Адыгея (Адыгея)	336720	328666	0	235040	15183	78443	15182
3	Республика Алтай	158891	156580	1186	97103	4630	53661	5816
4	Республика Башкортостан	3045698	2866878	0	2186576	111258	569036	111248
...	...	...	...	...	...	...	...	...
82	Еврейская автономная область	128844	120568	106	73398	4123	42941	4229
83	Ненецкий автономный округ	39470	33688	6116	18414	579	8579	6695
84	Ханты-Мансийский автономный округ - Югра	1130343	1095884	26633	745638	15804	307809	42426
85	Чукотский автономный округ	33541	32298	2531	24193	874	4700	3405
86	Ямало-Ненецкий автономный округ	370823	372171	41135	295648	4008	31380	45143

87 rows × 23 columns

Добавим в таблицу данные об общем количестве проголосовавших и явке

In [17]:
```
turnout_list = []
voted_list = []

for _, row in df_regions.iterrows():
    voted = row['N_EARLY'] + row['N_IN'] + row['N_OUT']
    voted_list.append(voted)
    turnout_list.append(voted/row['N_ALL'])

df_regions['N_VOTED'] = voted_list
df_regions['TURNOUT'] = turnout_list
df_regions.to_csv('regions_data.csv')
df_regions
```

Out[17]:

	REGION	N_ALL	N_GIVEN	N_EARLY	N_IN	N_OUT	N_LEFT	N_PORTABLE
0	Город Байконур (Республика Казахстан)	14575	12800	0	9491	168	3141	168
1	Территория за пределами РФ	483957	1408383	53482	403108	18026	933738	71484
2	Республика Адыгея	336720	328666	0	235040	15183	78443	15182

	(Адыгея)							
3	Республика Алтай	158891	156580	1186	97103	4630	53661	5816
4	Республика Башкортостан	3045698	2866878	0	2186576	111258	569036	111248
...	...	...	...	...	...	...	...	...
82	Еврейская автономная область	128844	120568	106	73398	4123	42941	4229
83	Ненецкий автономный округ	39470	33688	6116	18414	579	8579	6695
84	Ханты-Мансийский автономный округ - Югра	1130343	1095884	26633	745638	15804	307809	42426
85	Чукотский автономный округ	33541	32298	2531	24193	874	4700	3405
86	Ямало-Ненецкий автономный округ	370823	372171	41135	295648	4008	31380	45143

87 rows × 23 columns

Как и в исходном датасете, здесь нам не нужны некоторые столбцы. Удалим их в обеих таблицах

In [18]:
```
df = df.drop(columns=['N_GIVEN', 'N_LEFT', 'N_IN', 'N_PORTABLE', 'N_STAT
df_regions = df_regions.drop(columns=['N_GIVEN', 'N_LEFT', 'N_IN', 'N_PO
DF_REGIONS_LIST = [i for i in range(df_regions.shape[0])]
df_regions
```

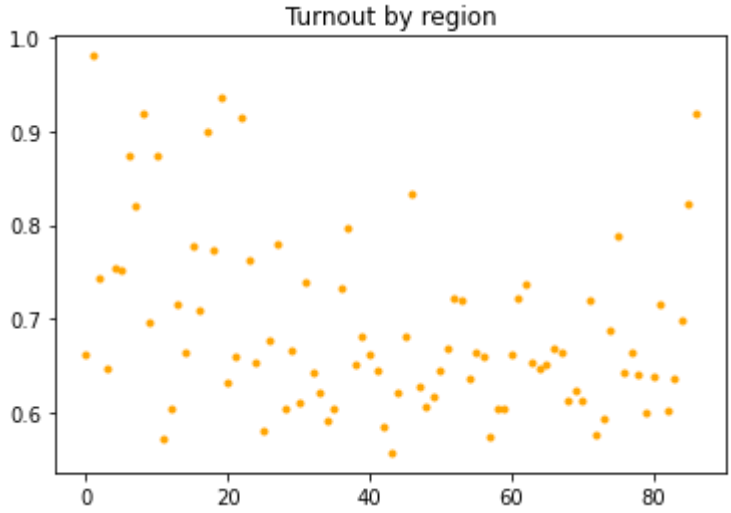
Out[18]:

	REGION	N_ALL	N_EARLY	N_OUT	N_INVALID	BABURIN	GRUDININ	ZHIRINOV
0	Город Байконур (Республика Казахстан)	14575	0	168	104	32	1026	
1	Территория за пределами РФ	483957	53482	18026	5801	2010	23871	
2	Республика Адыгея (Адыгея)	336720	0	15183	2647	1165	28711	
3	Республика Алтай	158891	1186	4630	974	446	21259	
4	Республика Башкортостан	3045698	0	111258	18506	15845	277797	11
...	...	...	...	...	...	...	...	
82	Еврейская автономная область	128844	106	4123	1331	501	14066	
83	Ненецкий автономный округ	39470	6116	579	267	185	3397	

84	Ханты-Мансийский автономный округ - Югра	1130343	26633	15804	10824	3916	94785	5
85	Чукотский автономный округ	33541	2531	874	293	115	1616	
86	Ямало-Ненецкий автономный округ	370823	41135	4008	2616	1052	19488	1

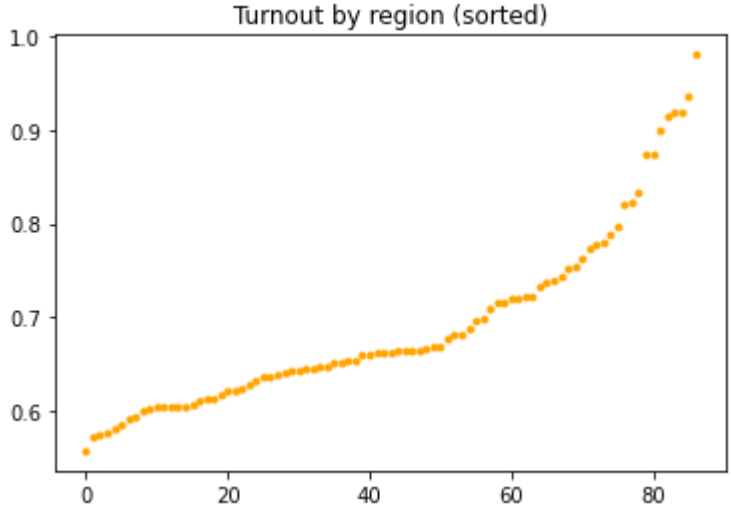
87 rows × 15 columns

In [19]:
```
plt.plot(DF_REGIONS_LIST, turnout_list, '.', color='orange')
plt.title('Turnout by region')
plt.show()
```



На графике показано распределение явки по регионам. Видно, что во всех регионах явка больше половину, в некоторых - почти 100 % (к этим регионам ещё вернёмся)

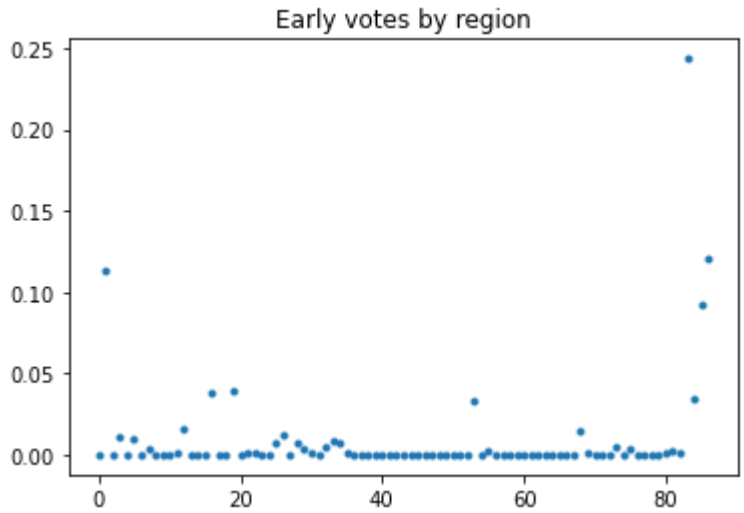
In [20]:
```
turnout_list = [df_regions['N_VOTED'][i]/df_regions['N_ALL'][i] for i in
plt.plot(DF_REGIONS_LIST, sorted(turnout_list), '.', color='orange')
plt.title('Turnout by region (sorted)')
plt.show()
```



Для удобства отсортировали график явки по регионам. Видим, что в большинстве

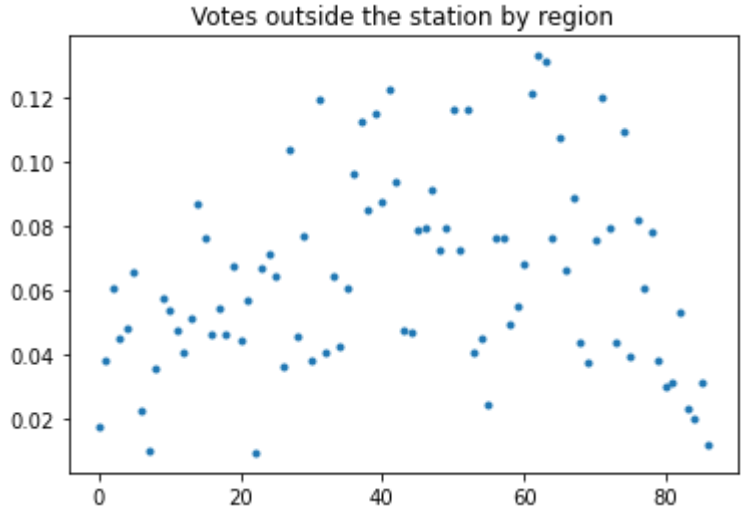
регионов явка составила около 65 %

In [21]:
```
plt.plot(DF_REGIONS_LIST, [df_regions['N_EARLY'][i]/df_regions['N_VOTED'
plt.title('Early votes by region')
plt.show()
```



На графике выше - процент голосов до дня голосования. В среднем он почти нулевой, однако есть регионы, где он составляет 5, 10 и даже 25 процентов

In [22]:
```
plt.plot(DF_REGIONS_LIST, [df_regions['N_OUT'][i]/df_regions['N_VOTED'][
plt.title('Votes outside the station by region')
plt.show()
```



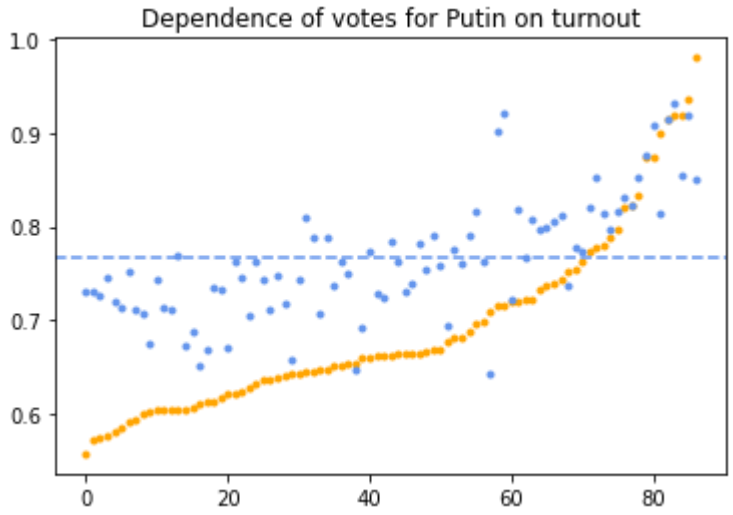
На данном графике - процент голосов за пределами УИК. Видим, что точки на графике распредены почти равномерно

Добавим функцию для изображения линии, равной среднему проценту победителя в выборах. Будем её использовать во многих графиках

In [23]:
```
def add_winner_average_line(obj):
    putin_list = [df_regions['PUTIN'][i]/df_regions['N_VOTED'][i] for i 
    obj.axhline(y=sum(putin_list)/len(putin_list), ls='--', color=COLORS
```

In [24]:
```
putin_list = [df_regions['PUTIN'][i]/df_regions['N_VOTED'][i] for i in r

turnout_sorted, putin_sorted = zip(*sorted(zip(turnout_list, putin_list)
plt.plot(DF_REGIONS_LIST, turnout_sorted, '.', color='orange')
```

```
plt.plot(DF_REGIONS_LIST, putin_sorted, '.', color=COLORS[CANDS.index('P

add_winner_average_line(plt)
plt.title('Dependence of votes for Putin on turnout')
plt.show()
```



На графике представлена зависимость голосов за Путина от явки. Как мы видим, наибольший процент голосов за Путина - в регионах с максимальной явкой. Более того, среди регионов с явкой меньше 75 % нет почти ни одного, где процент голосов за Путина превышал бы средний. Подозрительно. Далее изучим подробнее регионы с самой большой явкой (и низкой тоже). Для этого отсортируем нашу таблицу по явке

In [25]:
```
df_regions_sorted = df_regions.sort_values(by='TURNOUT', ascending=False
df_regions_sorted
```

Out[25]:

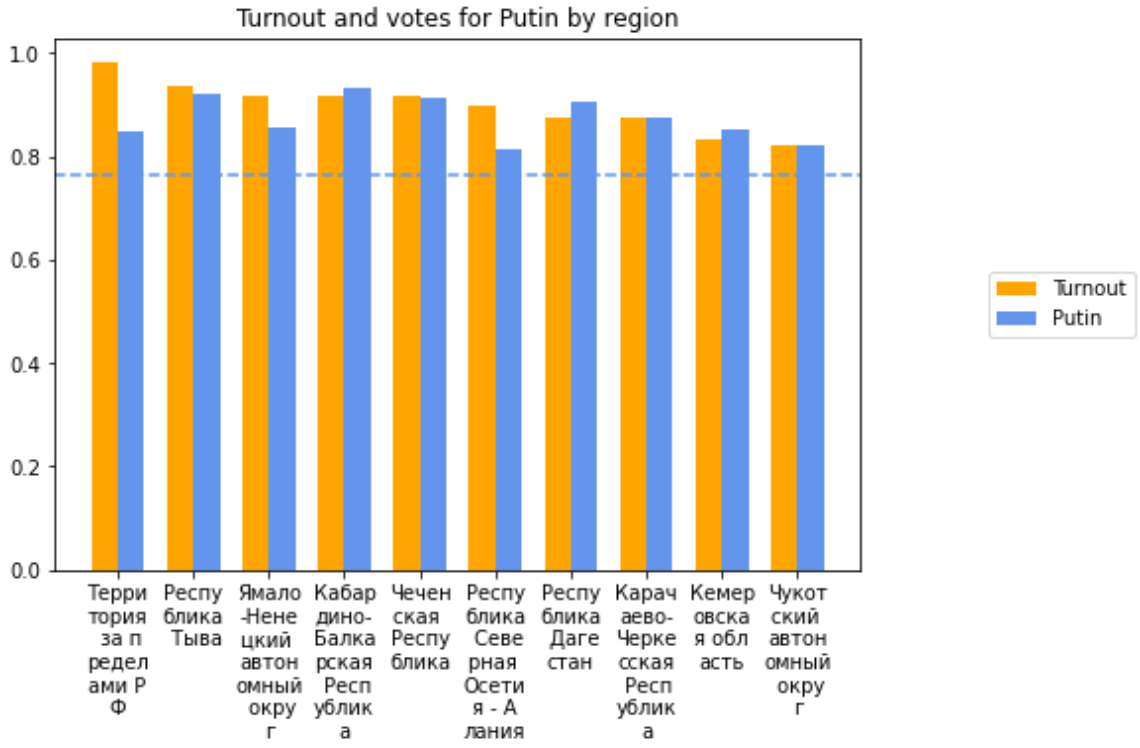
	REGION	N_ALL	N_EARLY	N_OUT	N_INVALID	BABURIN	GRUDININ	ZHIRINOV
0	Территория за пределами РФ	483957	53482	18026	5801	2010	23871	
1	Республика Тыва	175102	6476	11046	1160	406	5762	
2	Ямало-Ненецкий автономный округ	370823	41135	4008	2616	1052	19488	1
3	Кабардино-Балкарская Республика	528431	0	17453	628	1222	20133	
4	Чеченская Республика	709635	500	6250	2497	7679	30012	
...	...	...	...	...	...	...	...	
82	Забайкальский край	790054	3397	29543	5767	2450	62375	4
83	Тверская область	1070221	0	48898	6027	4718	77050	4
84	Новгородская область	502905	0	21925	3448	2345	36661	2
85	Республика Карелия	519667	257	14071	3656	2343	33693	2
86	Иркутская область	1877547	178	50075	11545	5480	166540	6

87 rows × 15 columns

Создадим функцию отображения явки и голосов за Путина в виде столбчатой диаграммы, чтобы подробнее проанализировать интересующие нас регионы

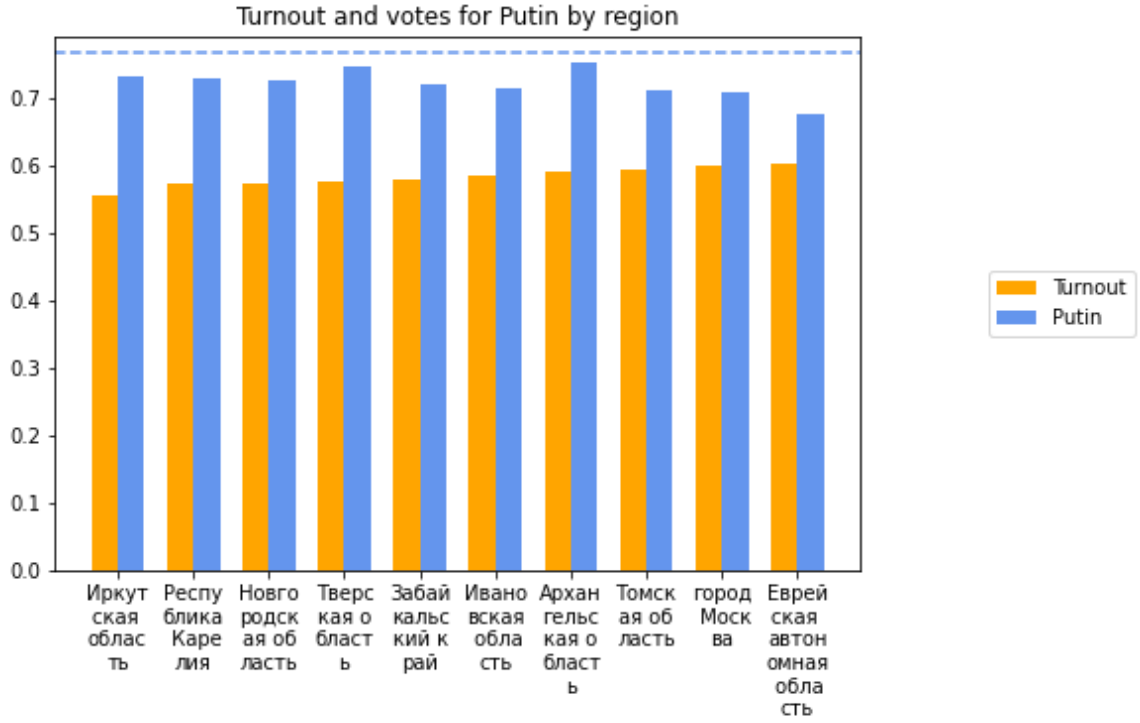
In [26]:
```
def draw_turnout_bar(num, reverse=False):
    if num > 10:
        print('The maxmium number of regions is 10')
        return

    start = 0 if not reverse else df_regions_sorted.shape[0] - 1
    stop = num if not reverse else start - num
    step = 1 if not reverse else -1
    turnout_num_list = [df_regions_sorted['TURNOUT'][i] for i in range(s
    putin_num_list = [df_regions_sorted['PUTIN'][i]/df_regions_sorted['N
    region_num_list = []
    for i in range(start, stop, step):
        new_region = ''
        region = df_regions_sorted['REGION'][i]
        for j in range(0, len(region), 15 - num):
            new_region += region[j:j + 15 - num] + '\n'
        region_num_list.append(new_region)

    ind = np.arange(len(turnout_num_list))
    width = 0.35

    fig, ax = plt.subplots()
    ax.bar(ind - width/2, turnout_num_list, width,
                    label='Turnout', color='orange')
    ax.bar(ind + width/2, putin_num_list, width,
                    label='Putin', color=COLORS[CANDS.index('PUTIN')])

    fig.tight_layout(pad=0.1)
    ax.set_title('Turnout and votes for Putin by region')
    ax.set_xticks(ind)
    ax.set_xticklabels(region_num_list, fontdict={'fontsize': 10})
    ax.legend(['Turnout', 'Putin'], loc="center", bbox_to_anchor=(1, 0,
    add_winner_average_line(ax)
```

In [27]:
```
draw_turnout_bar(10)
```



На диаграмме видно, что среди первых десяти регионах по явке нет ни одного, где процент за Путина был бы ниже среднего

In [28]:
```
draw_turnout_bar(10, reverse=True)
```



Здесь же, наоборот, нет ни одного региона, где количество голосов за Путина превышало бы среднее. Что ж, рассмотрим данные регионы ещё подробнее

Создадим функцию для визуализации по региону, которая вызывает ещё три - каждая рисует один график

In [29]:
```
def visualize_region_results(region):
```

```
    df_region = df.loc[df['REGION'] == region]
    if not len(df_region):
        print('You entered incorrect region')
        return
    if region in INCOMPLETE_REGIONS:
        print('Note that the data about this region is incomplete')
    df_region = df_region.reset_index()
    visualize_turnout(df_region)
    visualize_winner(df_region)
    visualize_dependence(df_region)
```

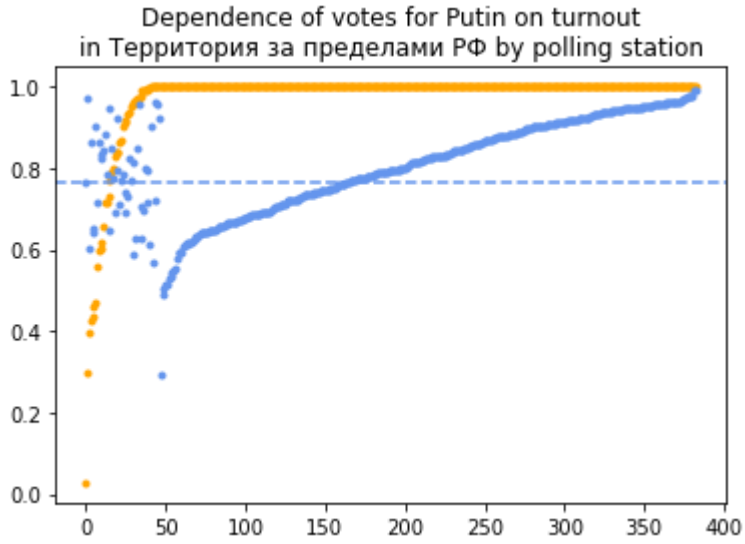
In [30]:
```
def visualize_turnout(df_region):
    plt.plot([i for i in range(df_region.shape[0])], sorted([df_region['
    plt.title(f"Turnout in {df_region['REGION'][0]} by polling station")
    plt.show()
```

In [31]:
```
def visualize_winner(df_region):
    plt.plot([i for i in range(df_region.shape[0])], sorted([df_region['
    plt.title(f"Votes for Putin in {df_region['REGION'][0]} by polling s
    plt.show()
```

In [32]:
```
def visualize_dependence(df_region):
    turnout_list = [df_region['N_VOTED'][i]/df_region['N_ALL'][i] for i 
    putin_list = [df_region['PUTIN'][i]/df_region['N_VOTED'][i] for i in
    turnout_sorted, putin_sorted = zip(*sorted(zip(turnout_list, putin_l
    plt.plot([i for i in range(df_region.shape[0])], turnout_sorted, '.'
    plt.plot([i for i in range(df_region.shape[0])], putin_sorted, '.', 
    add_winner_average_line(plt)

    plt.title(f"Dependence of votes for Putin on turnout\nin {df_region[
    plt.show()
```

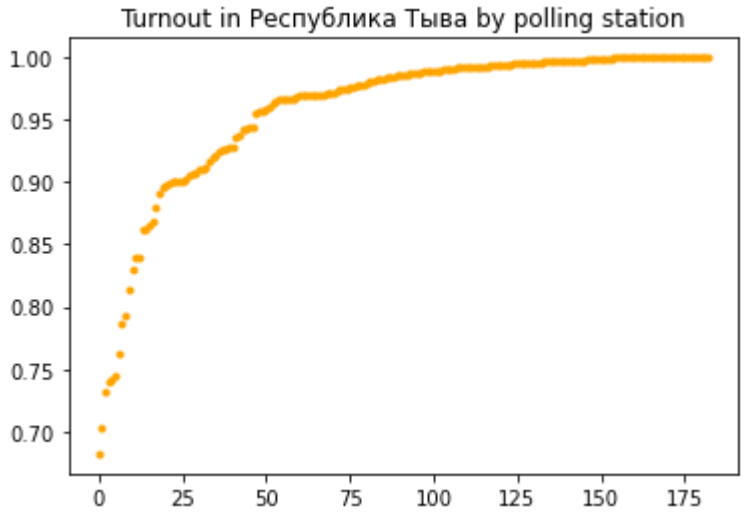
In [33]:
```
visualize_region_results(df_regions_sorted['REGION'][0])
```

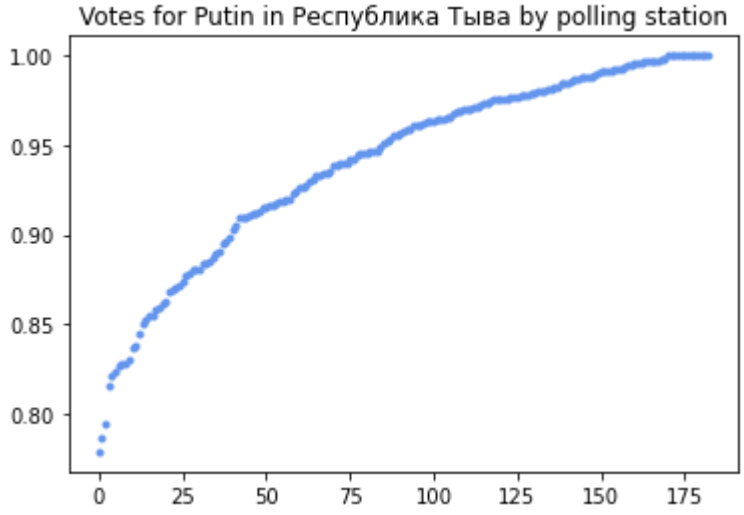
```
Note that the data about this region is incomplete
```

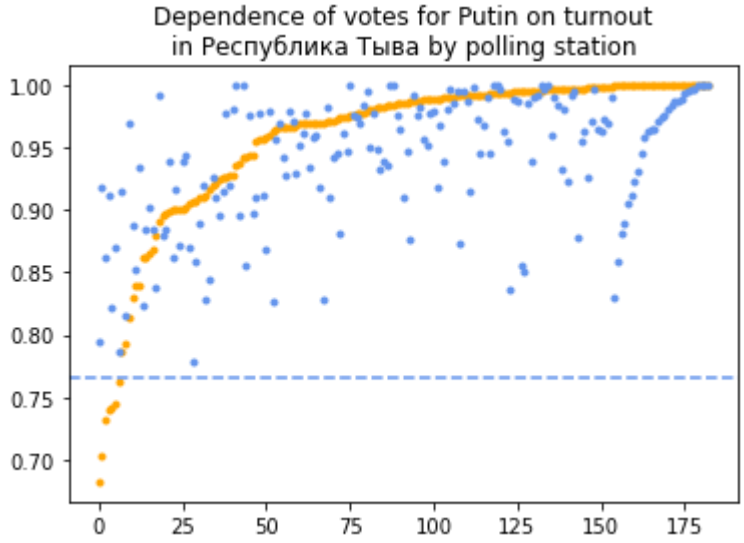


Как мы видим, при голосовании за пределами РФ явка составляет почти 100 % на большинстве участков. Это, скорее всего, связано с тем, что на участках за пределами РФ регистрируются заранее. Есть человек подал заявление на включение в список избирателей, он наверняка сходит и на выборы

In [34]:
```
visualize_region_results(df_regions_sorted['REGION'][1])
```

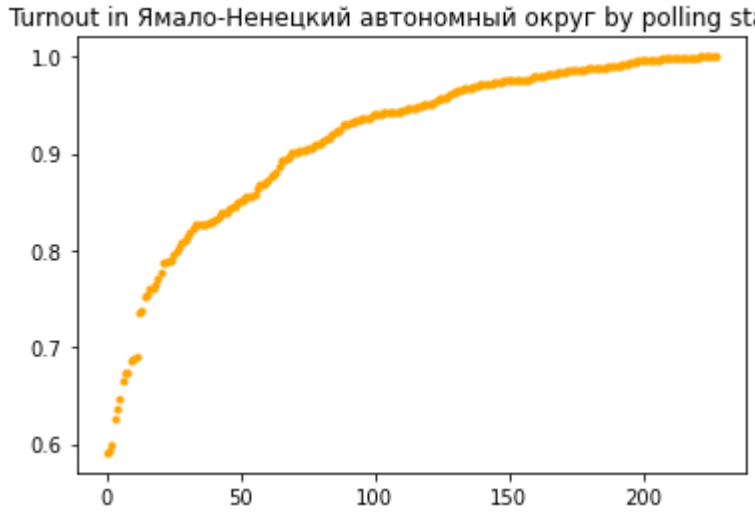


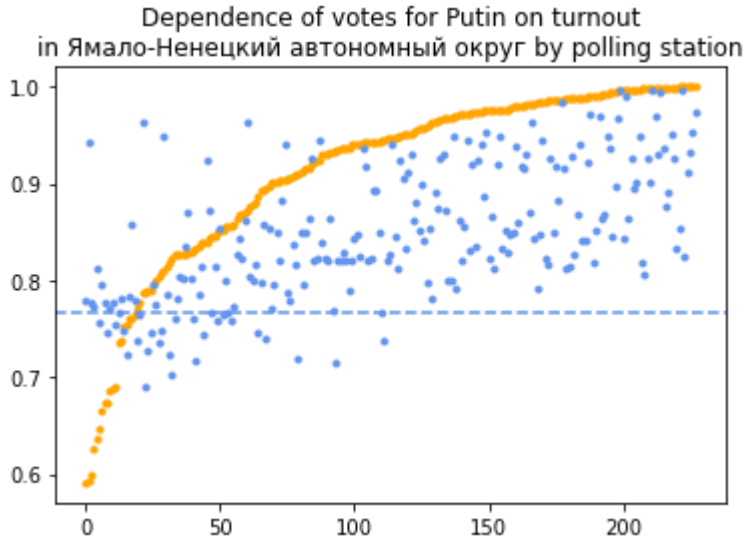




Видим, что даже в пределах региона нет ни одного участка, где процент у Путина был меньше среднего (как, например, на территории за пределами РФ)

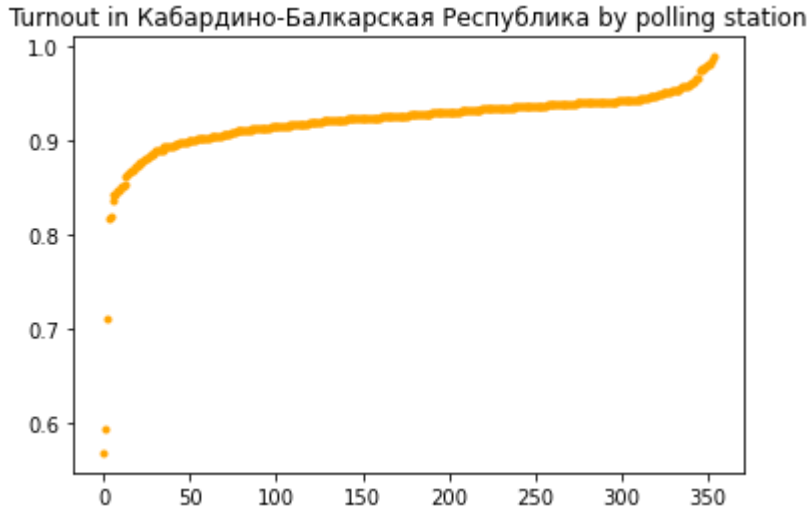
In [35]:
```
visualize_region_results(df_regions_sorted['REGION'][2])
```

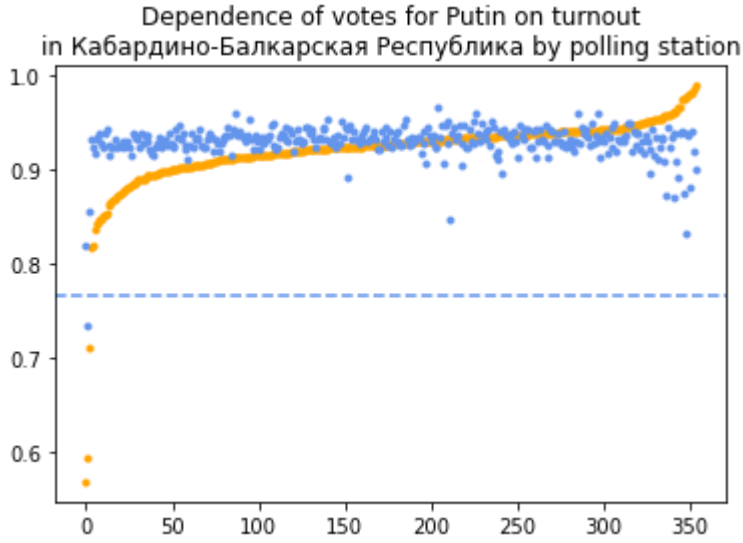




Заметим, что скопление точек у линии среднего значения именно там, где низкая явка

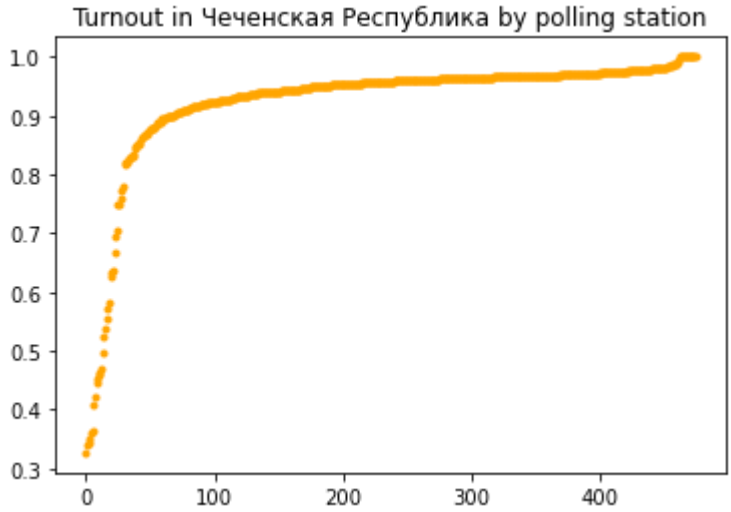
In [36]:
```
visualize_region_results(df_regions_sorted['REGION'][3])
```

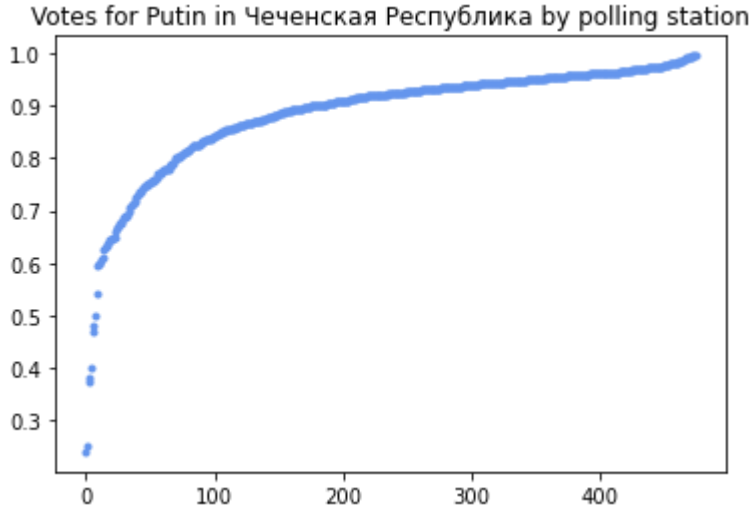


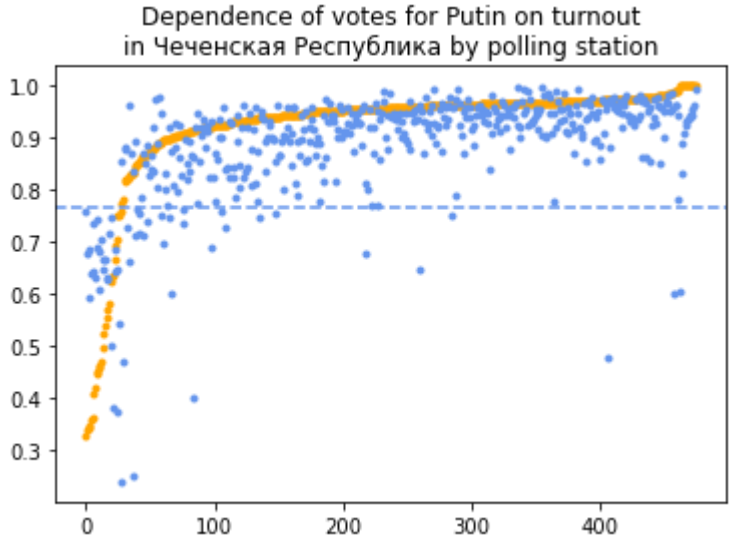


Невероятно высокий процент голосов за Путина вне зависимости от явки (которая, к слову, тоже сосредоточена в районе 90 %)

In [37]:
```
visualize_region_results(df_regions_sorted['REGION'][4])
```



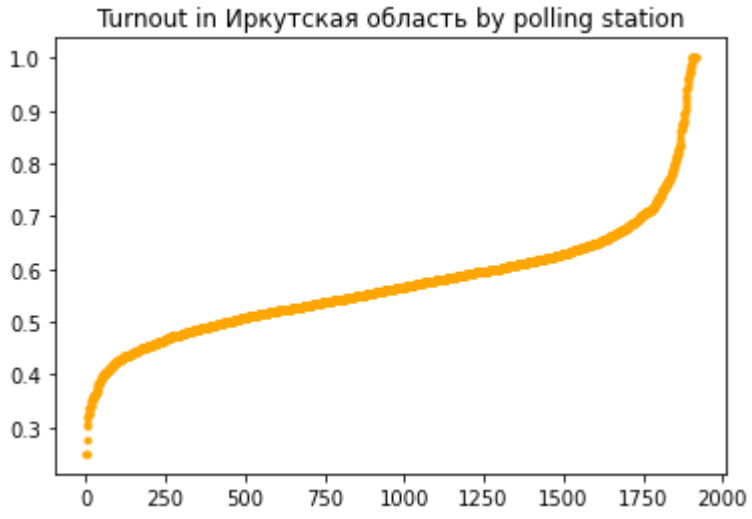


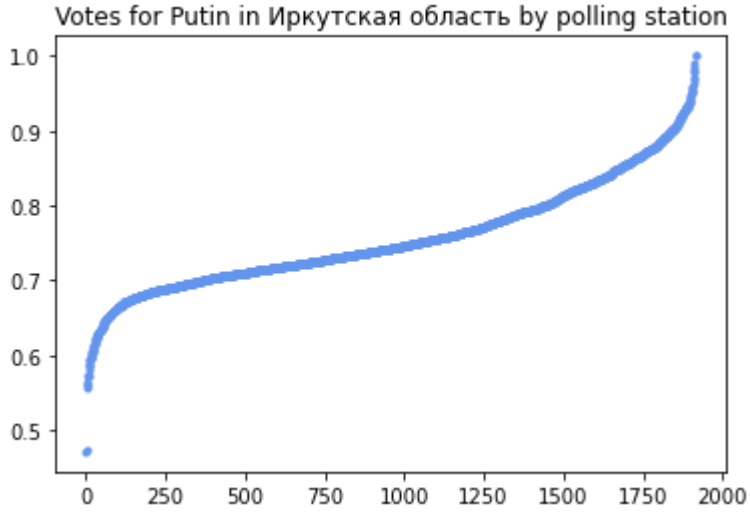


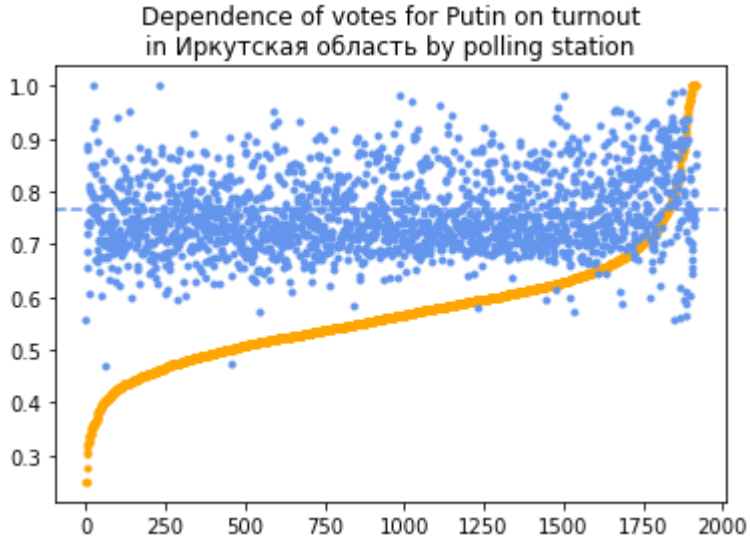
Знакомая ситуация: низкий процент у Путина там же, где низкая явка. И это при том, что на прошлых выборах в данном регионе Путин получил почти 100 % при почти стопроцентной явке. Кажется: в этот раз на нескольких участках были наблюдатели (и испортили всю картину!)

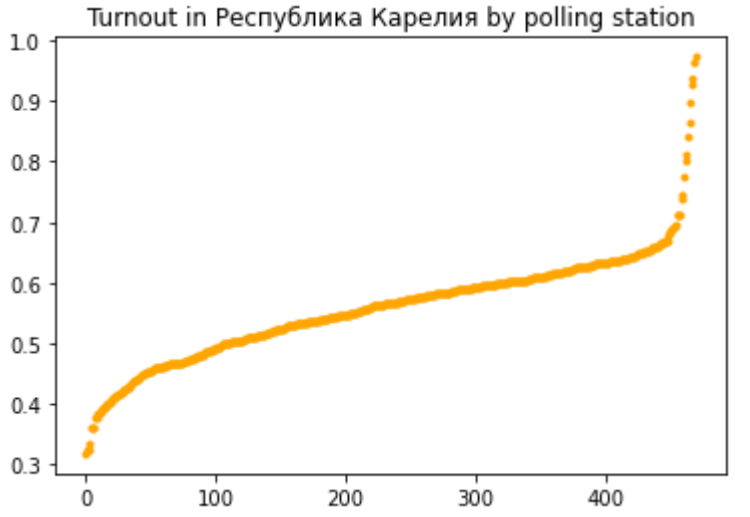
Теперь посмотрим на регионы с низкой явкой

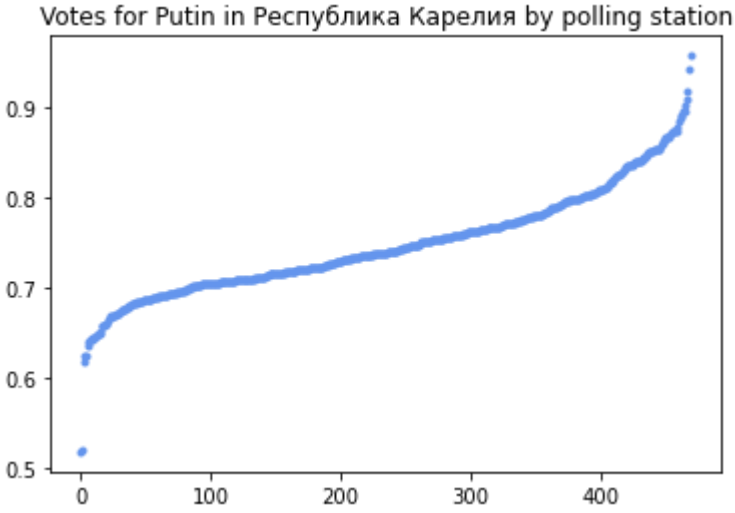
In [38]:
```
for i in range(1, 6):
    visualize_region_results(df_regions_sorted['REGION'][df_regions.shape
```

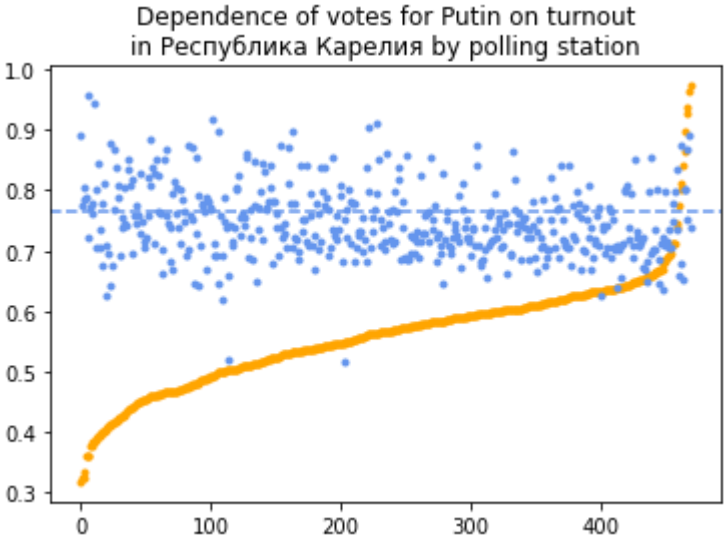


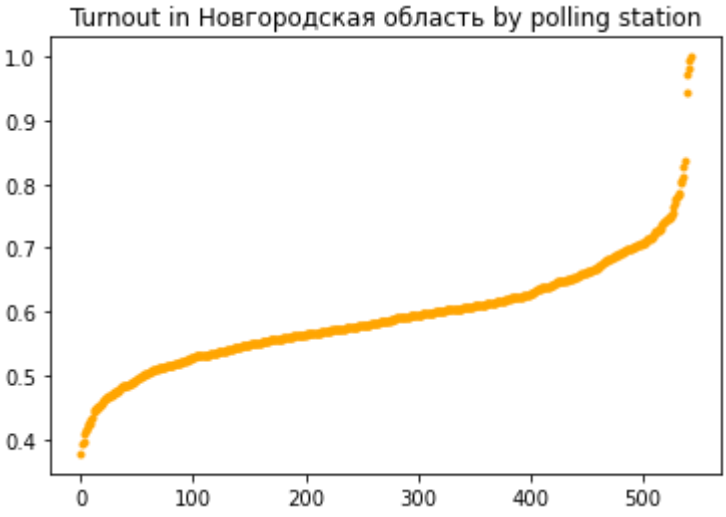

Votes for Putin in Иркутская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0
250
500
750
1000
1250
1500
1750
2000


Dependence of votes for Putin on turnout
in Иркутская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
250
500
750
1000
1250
1500
1750
2000

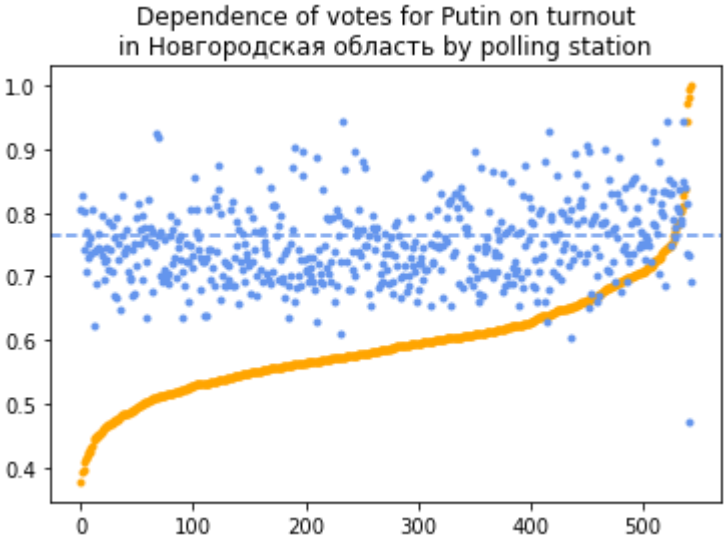

Turnout in Республика Карелия by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
100
200
300
400

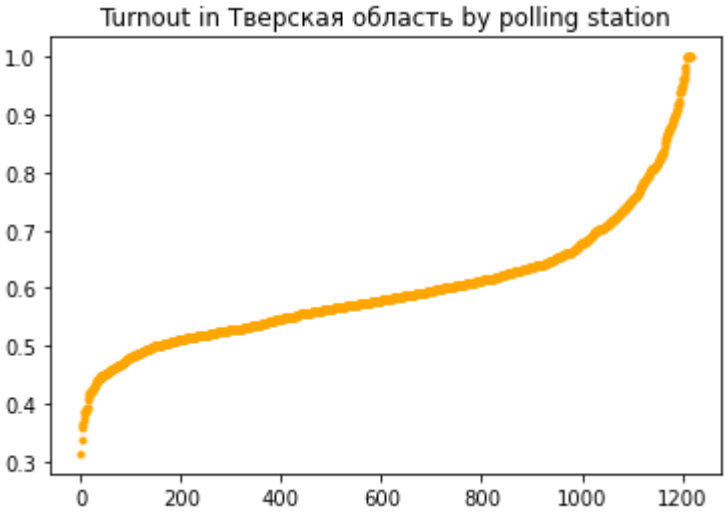

Votes for Putin in Республика Карелия by polling station
0.9
0.8
0.7
0.6
0.5
0
100
200
300
400

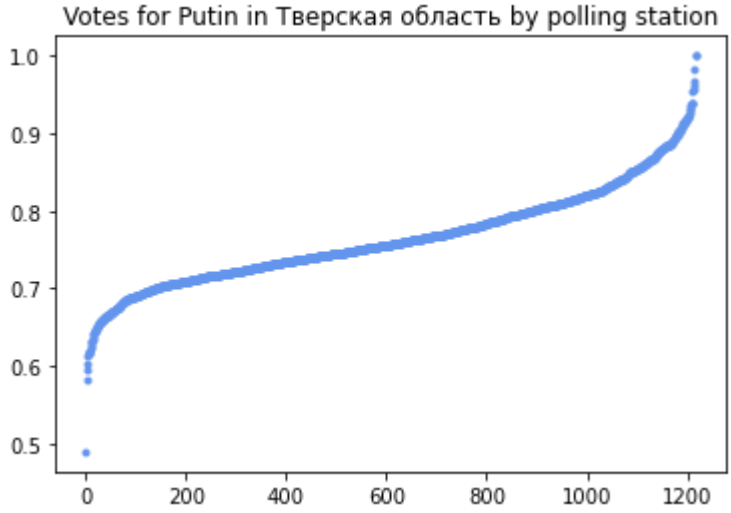

Dependence of votes for Putin on turnout
in Республика Карелия by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
100
200
300
400

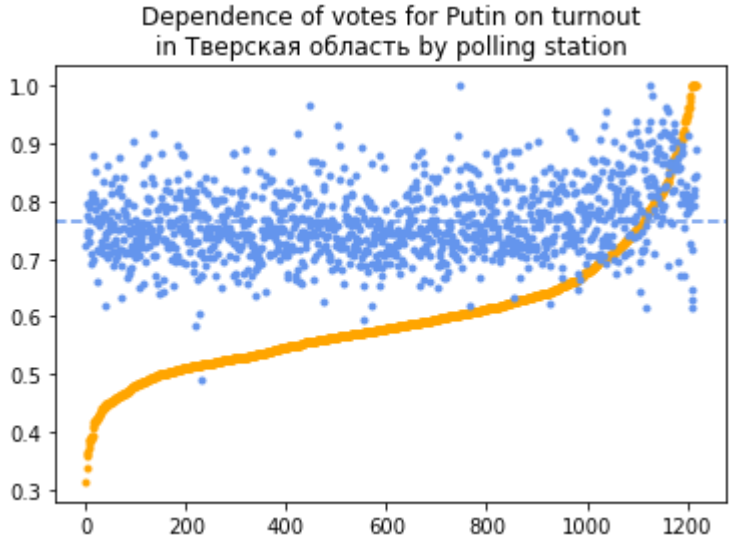

Turnout in Новгородская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
100
200
300
400
500

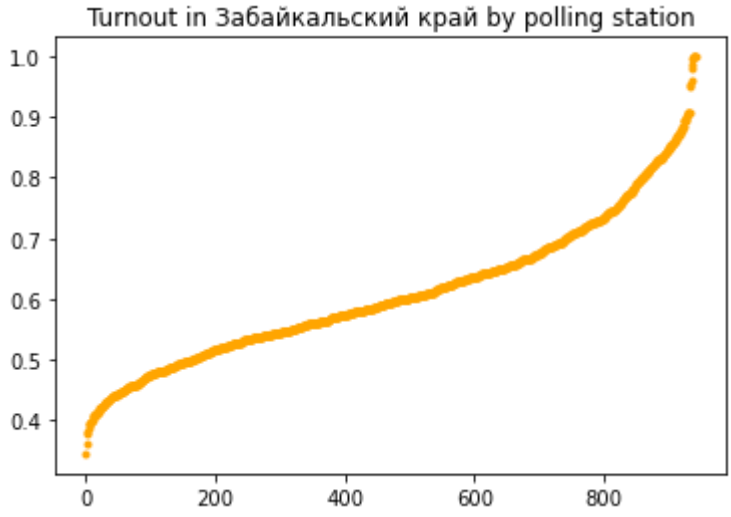

Votes for Putin in Новгородская область by polling station
0.9
0.8
0.7
0.6
0.5
0
100
200
300
400
500

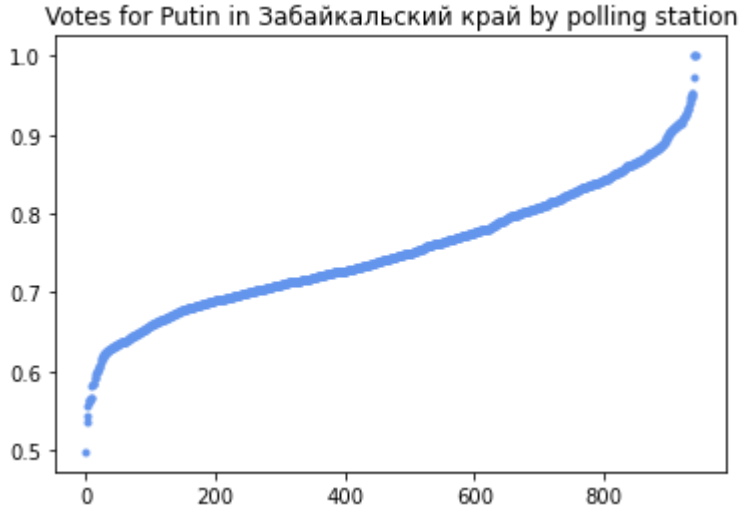

Dependence of votes for Putin on turnout
in Новгородская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
100
200
300
400
500

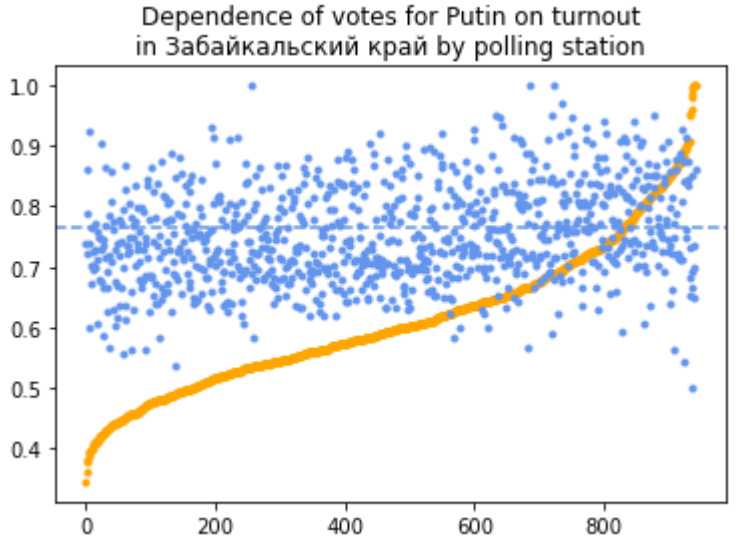

Turnout in Тверская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000
1200


Votes for Putin in Тверская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0
200
400
600
800
1000
1200


Dependence of votes for Putin on turnout
in Тверская область by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000
1200


Turnout in Забайкальский край by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
200
400
600
800

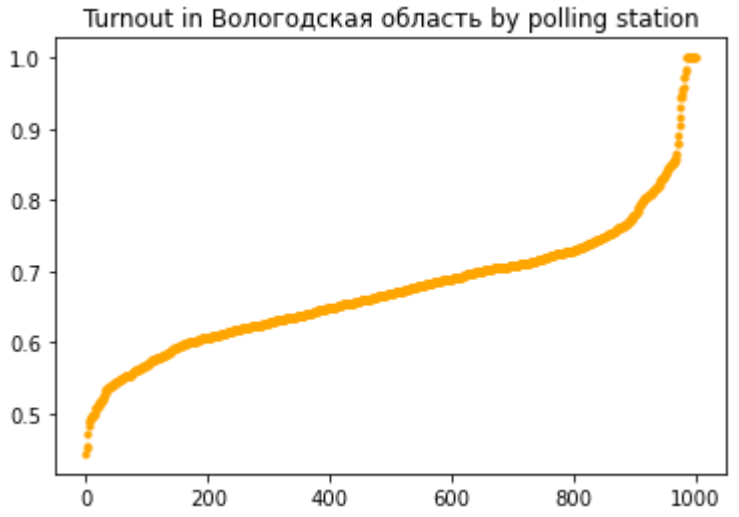


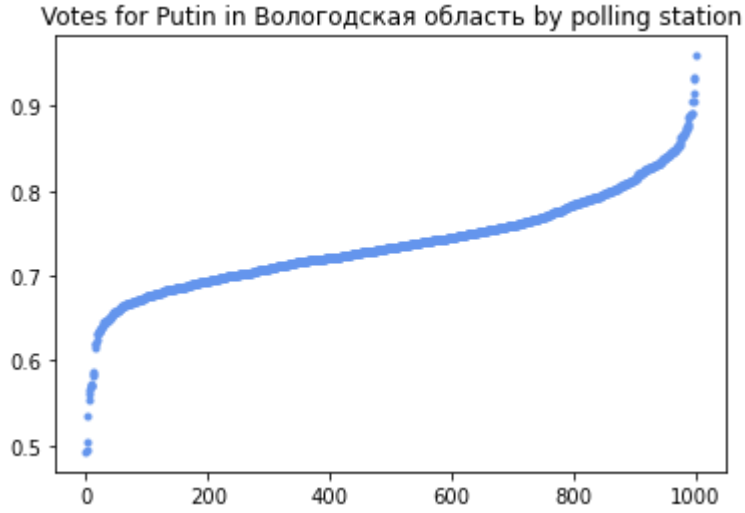


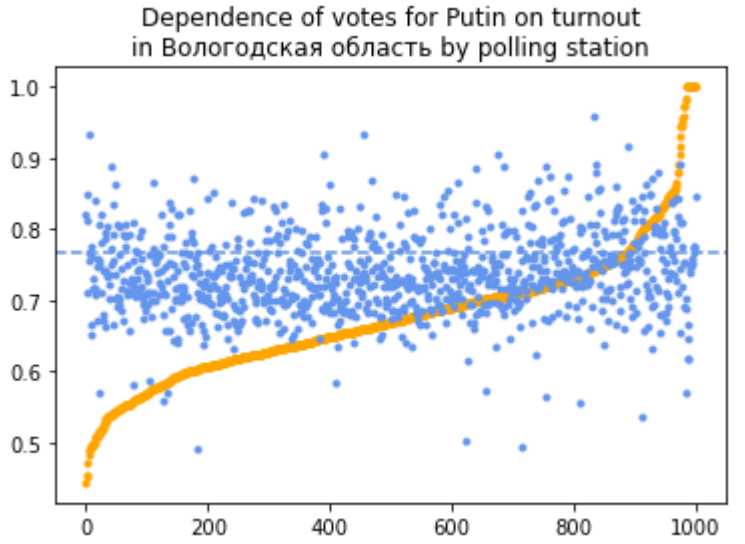
Заметим, что во всех пяти регионах как явка, так и голоса за Путина распределены более-менее равномерно. Голоса расположены по обе стороны от средней линии, независимо от явки

Рассмотрим регион из середины списка. Получаем аналогичные результаты

In [39]:
```
visualize_region_results(df_regions['REGION'][40])
```

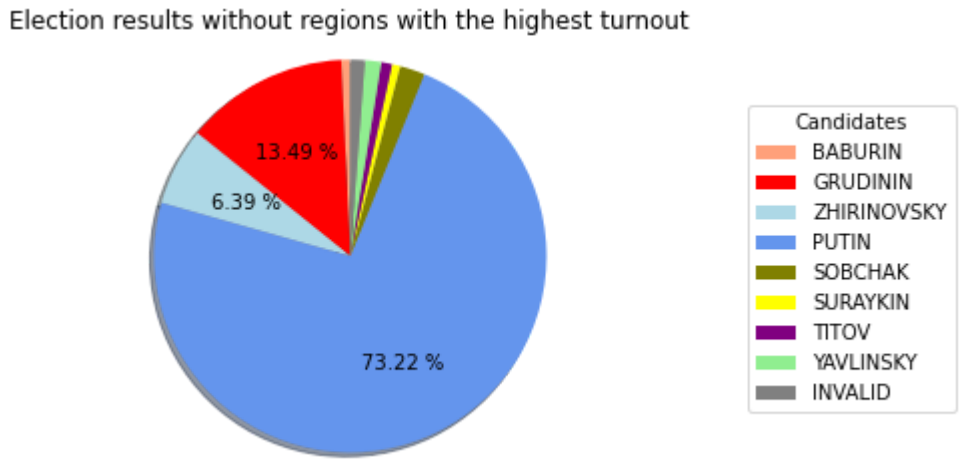






Посмотрим, что будет, если посчитать результаты выборов в той половине регионов, где низкая явка

In [40]:
```
labels = CANDS.copy() + ['INVALID']
start = list(df_regions_sorted.columns).index('BABURIN')
end = list(df_regions_sorted.columns).index('YAVLINSKY') + 1
sizes = [df_regions_sorted[45:][i].sum() for i in df_regions_sorted.colu
sizes.append(df_regions_sorted['N_INVALID'][45:].sum())

_, ax = plt.subplots()
ax.pie(sizes, labels=labels,
        autopct=(lambda x: f'{x:.2f} %' if x > 5.0 else ''),
        shadow=True, startangle=90, labeldistance=None,
        colors=COLORS + ['grey'])
ax.axis('equal')
ax.legend(labels, title="Candidates", loc="center", bbox_to_anchor=(1, 0

plt.title('Election results without regions with the highest turnout')
plt.show()
```

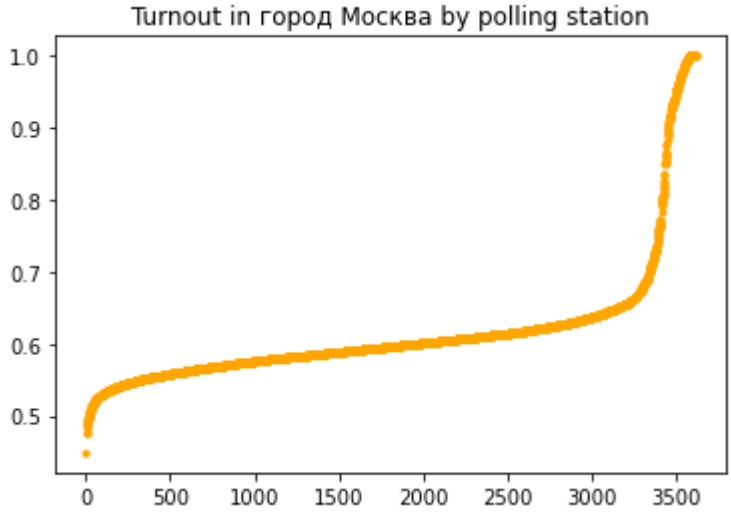


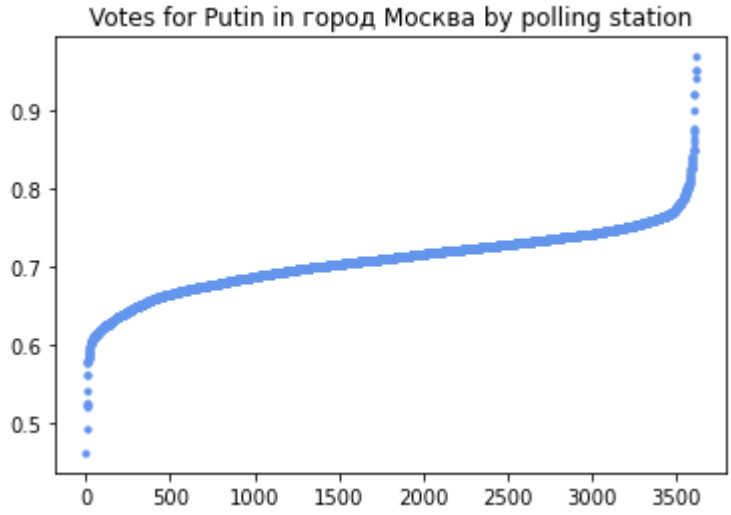
Видим, что Путин потерял больше трёх процентов. Хотя, казалось бы, люди должны голосовать примерно равномерно (во всяком случае, распределение голосов может быть обусловлено ососбенностями региона, а не явкой)

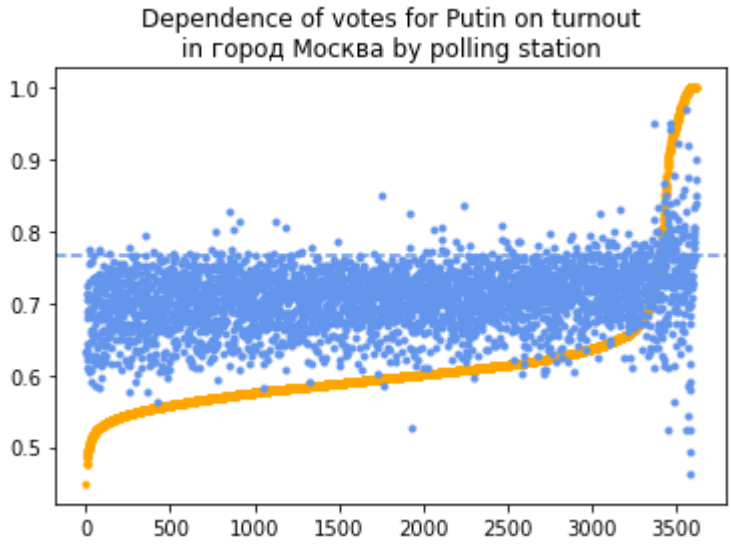
Посмотрим, как голосовали крупнейшие города - Москва и Санкт-Петербург

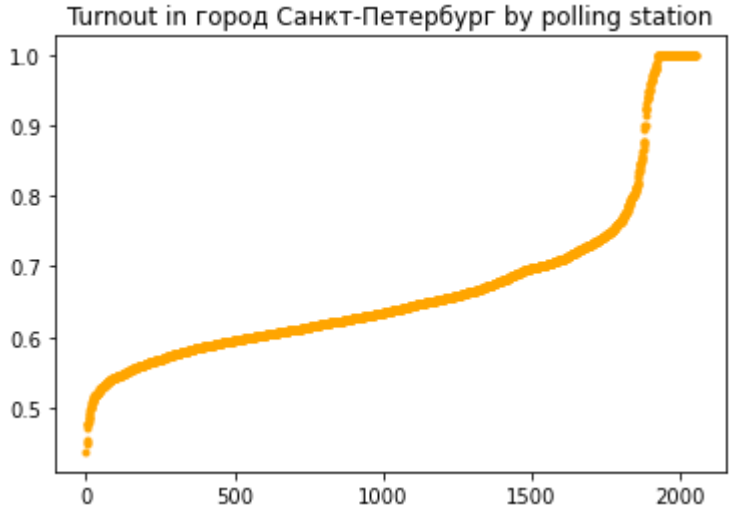
In [41]:
```
visualize_region_results('город Москва')
visualize_region_results('город Санкт-Петербург')
```

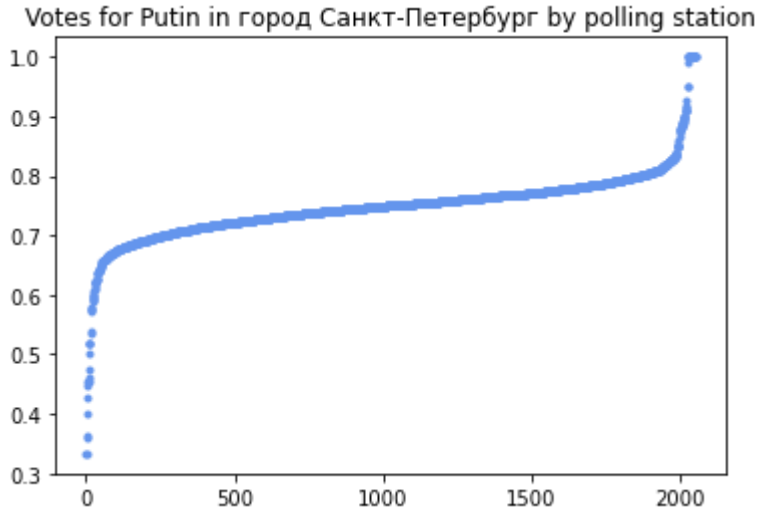
```
Note that the data about this region is incomplete
```

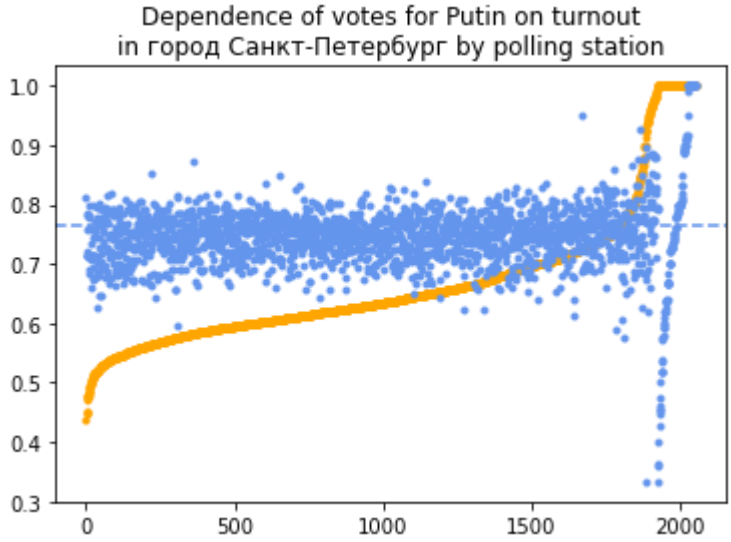





Dependence of votes for Putin on turnout
in город Москва by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0
500
1000
1500
2000
2500
3000
3500


Turnout in город Санкт-Петербург by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0
500
1000
1500
2000

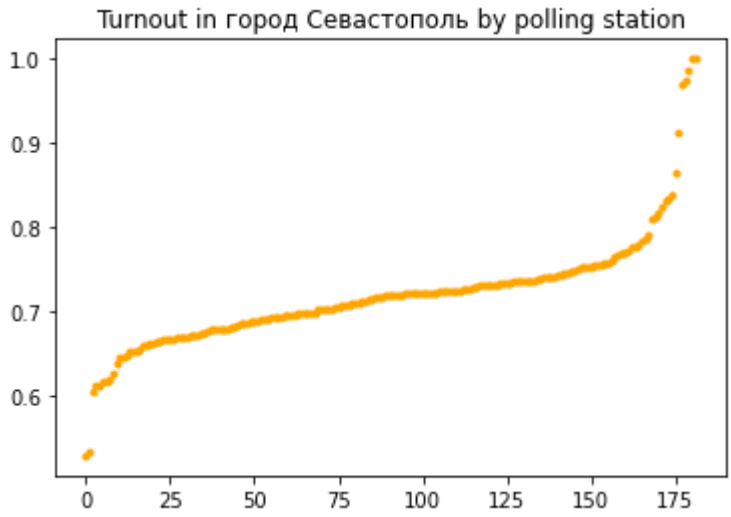

Votes for Putin in город Санкт-Петербург by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
500
1000
1500
2000

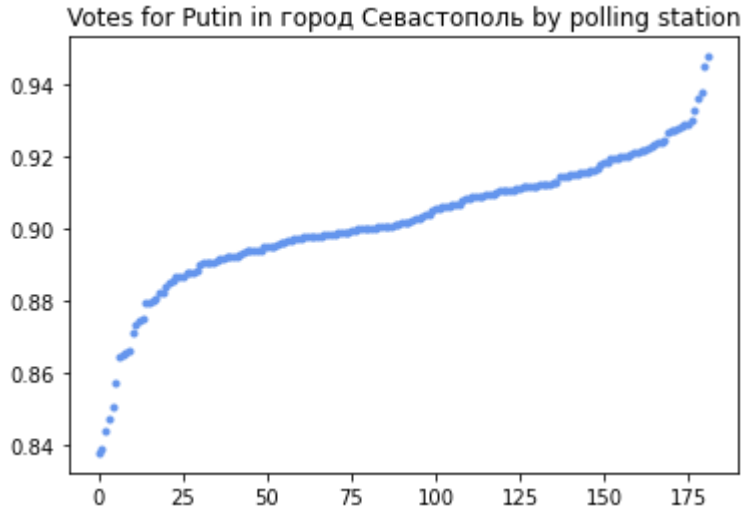


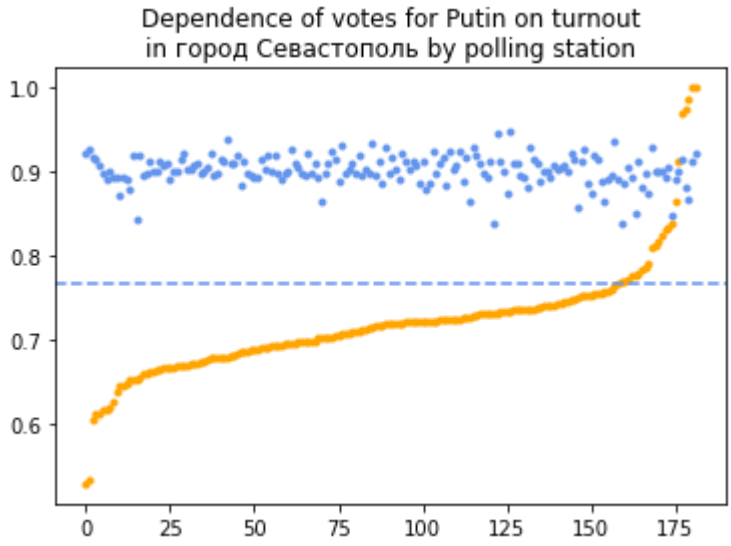
Видим, что явка и особенно количество голосов за Путина ниже, чем в целом по стране. Однако тяжело не заметить большое количество участков со стопроцентной явкой (особенно в Санкт-Петербурге), что вызывает некоторые подозрения

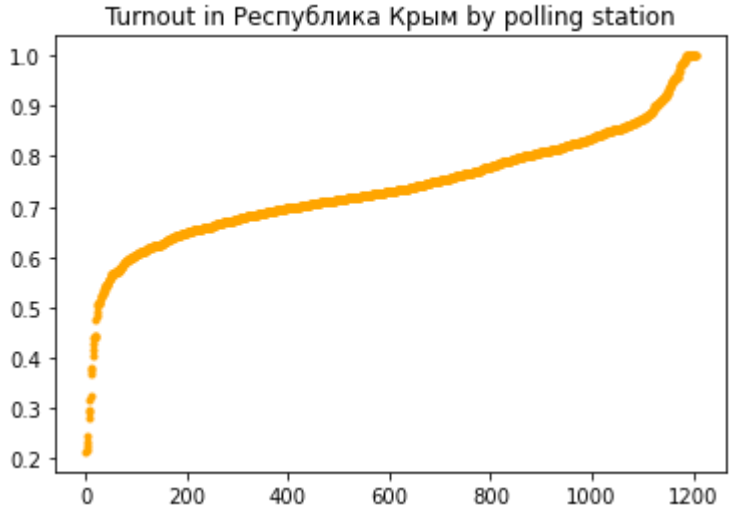
Посмотрим, как прогосовали жители недавно присоединённой территории

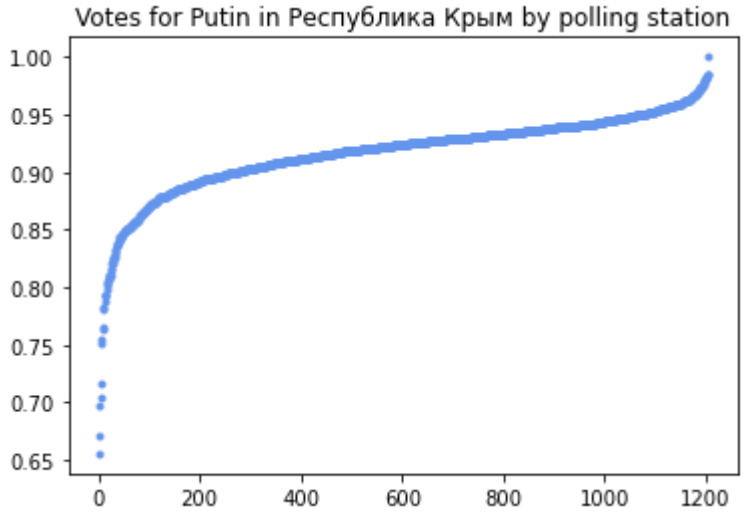
In [42]:
```
visualize_region_results('город Севастополь')
visualize_region_results('Республика Крым')
```

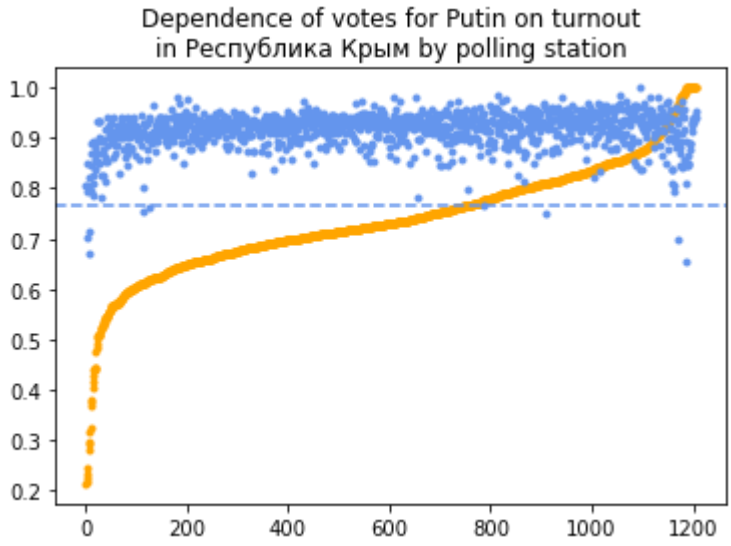





Dependence of votes for Putin on turnout
in город Севастополь by polling station
1.0
0.9
0.8
0.7
0.6
0
25
50
75
100
125
150
175


Turnout in Республика Крым by polling station
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
1200


Votes for Putin in Республика Крым by polling station
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
200
400
600
800
1000
1200



Как и предполагалось, на присоединённой территории видна высокая поддержка Путина и достаточно большая явка

Ранее мы заметили, что в некоторых регионах достаточно высокий процент досрочного голосования. Посмотрим подробнее

In [43]:
```
early_list = []

for _, row in df_regions.iterrows():
    early_list.append(row['N_EARLY']/row['N_VOTED'])

df_regions_early = df_regions.copy()
df_regions_early['EARLY'] = early_list


df_regions_early = df_regions_early.sort_values(by='EARLY', ascending=Fa
df_regions_early
```

Out[43]:

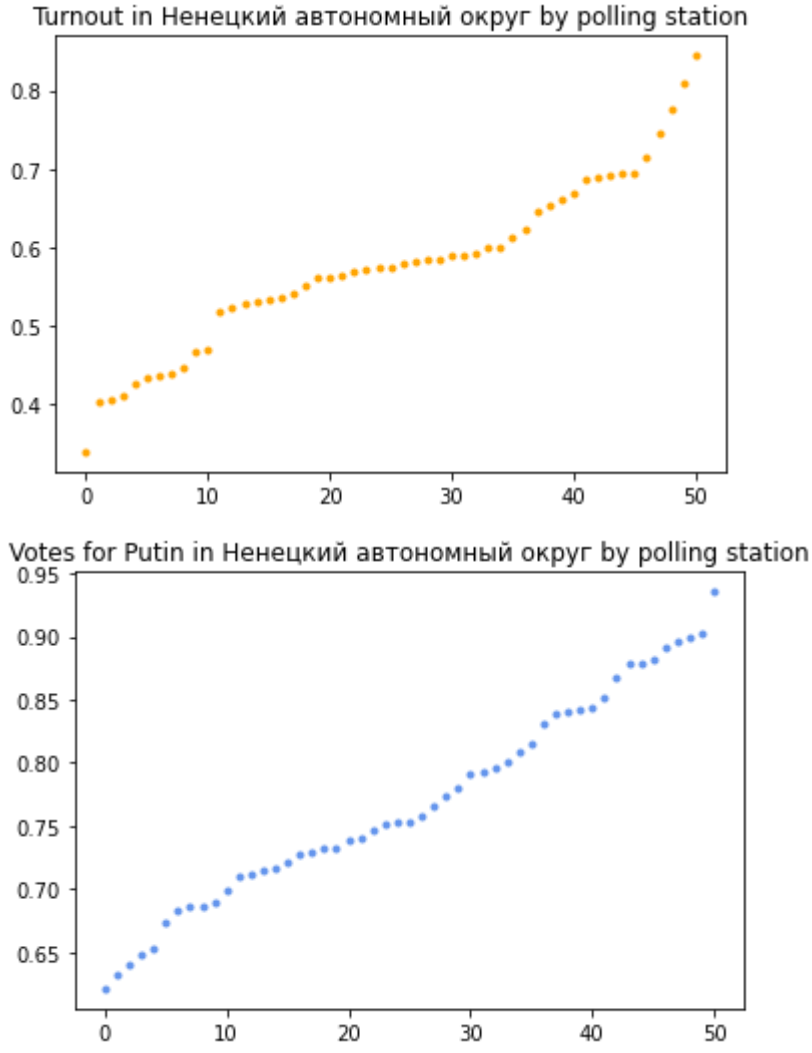
	REGION	N_ALL	N_EARLY	N_OUT	N_INVALID	BABURIN	GRUDININ	ZHIRINO
0	Ненецкий автономный округ	39470	6116	579	267	185	3397	
1	Ямало-Ненецкий автономный округ	370823	41135	4008	2616	1052	19488	
2	Территория за пределами РФ	483957	53482	18026	5801	2010	23871	
3	Чукотский автономный округ	33541	2531	874	293	115	1616	
4	Республика Тыва	175102	6476	11046	1160	406	5762	
...	...	...	...	...	...	...	...	
82	Новосибирская область	2159912	0	64418	16625	8473	213707	
83	Омская область	1535403	0	51132	12487	10496	189303	
84	Республика Татарстан (Татарстан)	2919482	0	104476	18045	13030	204618	

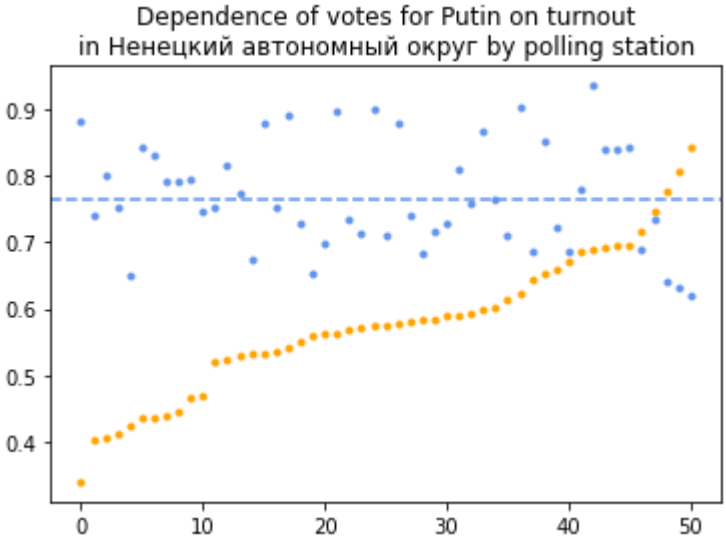
85	Брянская область	978509	0	87912	7177	4472	68375	
86	Город Байконур (Республика Казахстан)	14575	0	168	104	32	1026	

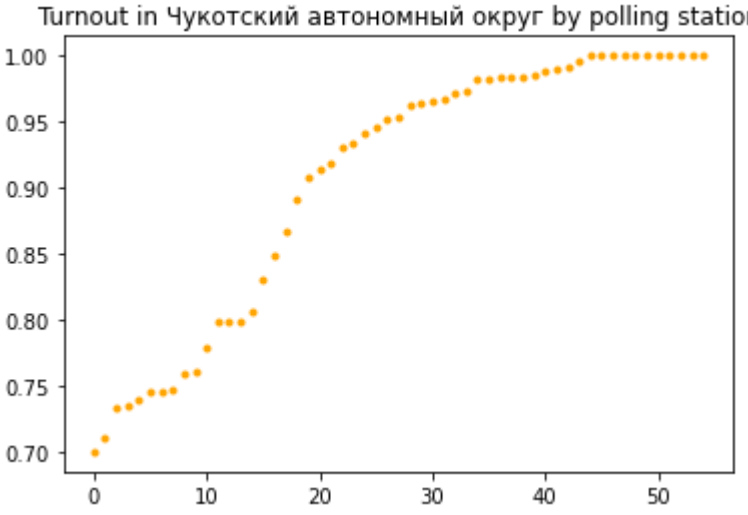
87 rows × 16 columns

Некоторые регионы мы уже рассматривали в связи с высокой явкой. Рассмотрим Ненецкий и Чукотский автономные округи

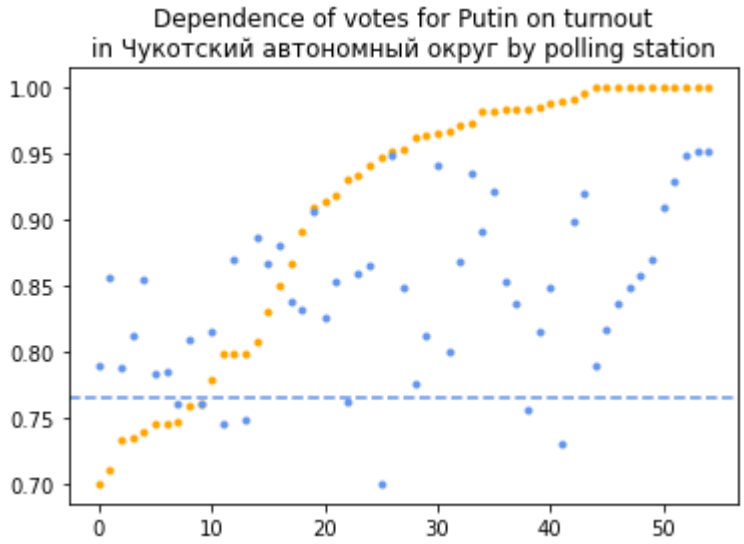
In [44]:
```
visualize_region_results('Ненецкий автономный округ')
visualize_region_results('Чукотский автономный округ')
```




Dependence of votes for Putin on turnout
in Ненецкий автономный округ by polling station
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50


Turnout in Чукотский автономный округ by polling station
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
10
20
30
40
50


Votes for Putin in Чукотский автономный округ by polling station
0.95
0.90
0.85
0.80
0.75
0.70
0
10
20
30
40
50



В Ненецком автономном круге результаты напоминают средние по стране, в то время как на Чукотке мы видим значительно превышающие средние значения явку и поддержку президента

Учитывая, что Грудинин набрал в среднем по стране больше 11 процентов, можно предположить, что в каком-то регионе он победил

In [45]:
```
for i in range(df_regions.shape[0]):
    if df_regions['GRUDININ'][i] > df_regions['PUTIN'][i]:
        print(df_regions['REGION'][i])
```

Видим, что таких регионов нет. Для наглядности воспользуемся геоданными

In [46]:
```
data = {}
with open('../geo.json', 'r') as f:
    data = json.loads(f.read())
```

Для работы с геоданными нужно установить соответствие между ID регионов в файле и в нашей таблице

In [47]:
```
id_list = [-1, -1, 1, 19, 7, 10, 17, 20, 23, 25, 28, 29, 36, 400, 46, 47
df_regions['ID_GEO'] = id_list
```

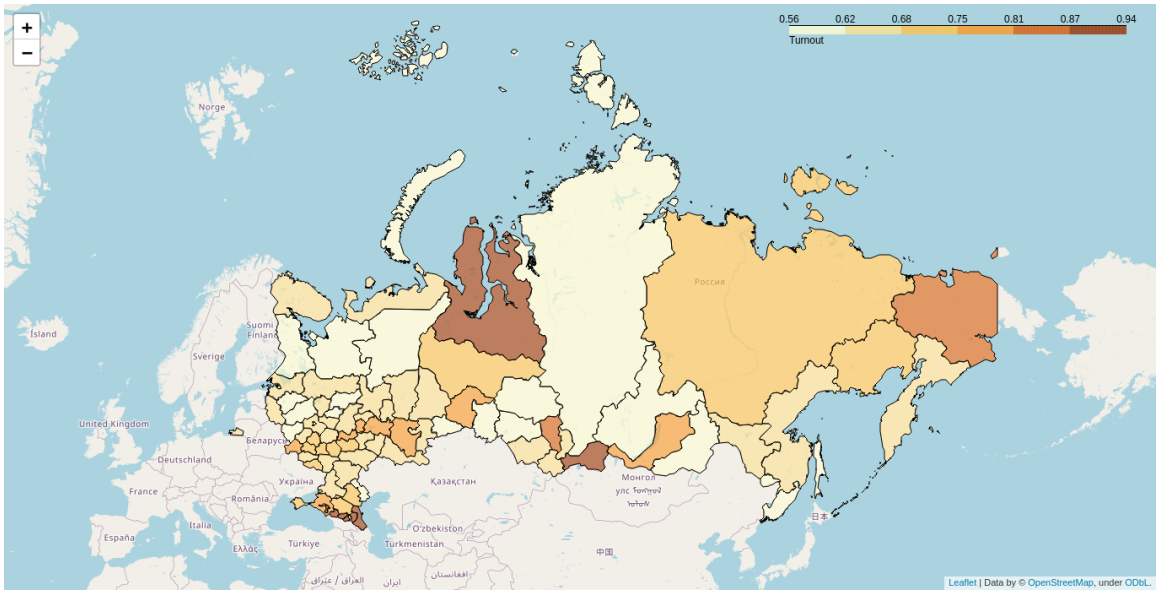
Создадим карту с информацией о явке

In [48]:
```
m = folium.Map(location=[68, 93], zoom_start=3)
folium.Choropleth(
    geo_data=data,
    data=df_regions[2:],
    columns=['ID_GEO', 'TURNOUT'],
    key_on='feature.properties.ID_1',
    fill_color='YlOrBr',
    legend_name='Turnout',
).add_to(m)

m
```

Out[48]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [49]:
```
img_data = m._to_png(5)
img = Image.open(io.BytesIO(img_data))
img
```

Out[49]:



Теперь посмотрим, как выглядит на карте поддержка каждого из кандидатов

In [50]:
```
def draw_map_for_candidate(cand):
    if cand not in CANDS:
        print('You entered incorrect candidate')
        return

    df_regions_cand = df_regions.copy()
    df_regions_cand['CAND'] = [df_regions_cand[cand][i]/df_regions_cand[


    m = folium.Map(location=[68, 93], zoom_start=3)
    folium.Choropleth(
        geo_data=data,
        data=df_regions_cand[2:],
        columns=['ID_GEO', 'CAND'],
        key_on='feature.properties.ID_1',
        fill_color='BuPu',
        legend_name=f'Votes for {cand}',
    ).add_to(m)

    return m
```

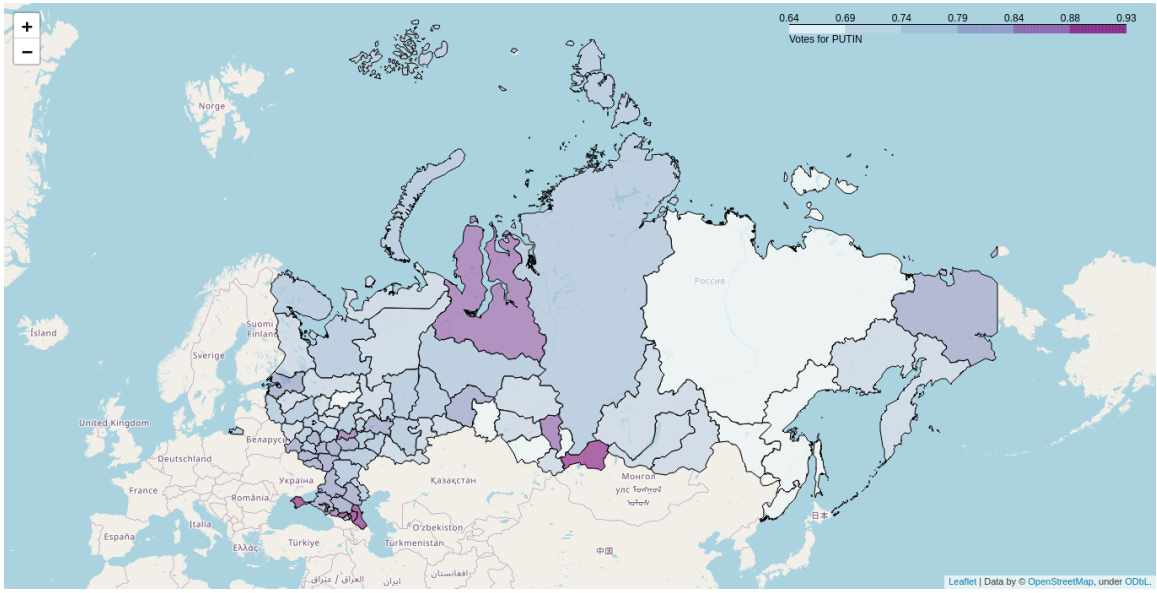
In [51]:
```
map_putin = draw_map_for_candidate('PUTIN')
```

```
map_putin
```

Out[51]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [52]:
```
img_data = map_putin._to_png(5)
img = Image.open(io.BytesIO(img_data))
img
```

Out[52]:



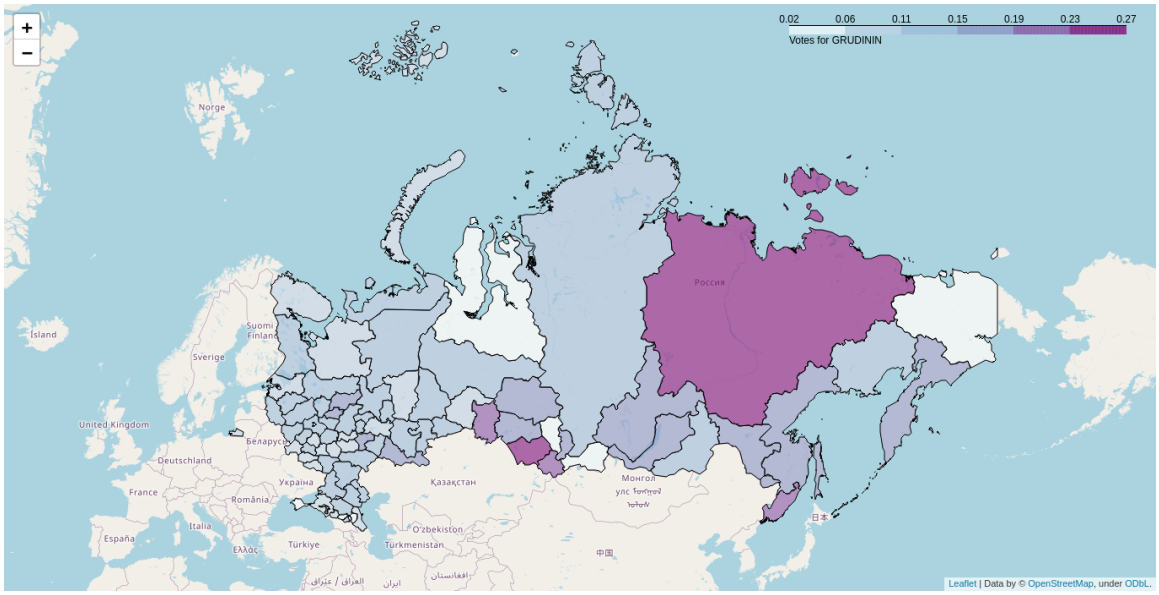
В очередной раз замечаем, что регионы с высокой явкой (коричневые) совпадают с регионами с высокой поддержкой Путина (фиолетовые)

In [53]:
```
map_grudinin = draw_map_for_candidate('GRUDININ')
map_grudinin
```

Out[53]: Make this Notebook Trusted to load map: File -> Trust Notebook

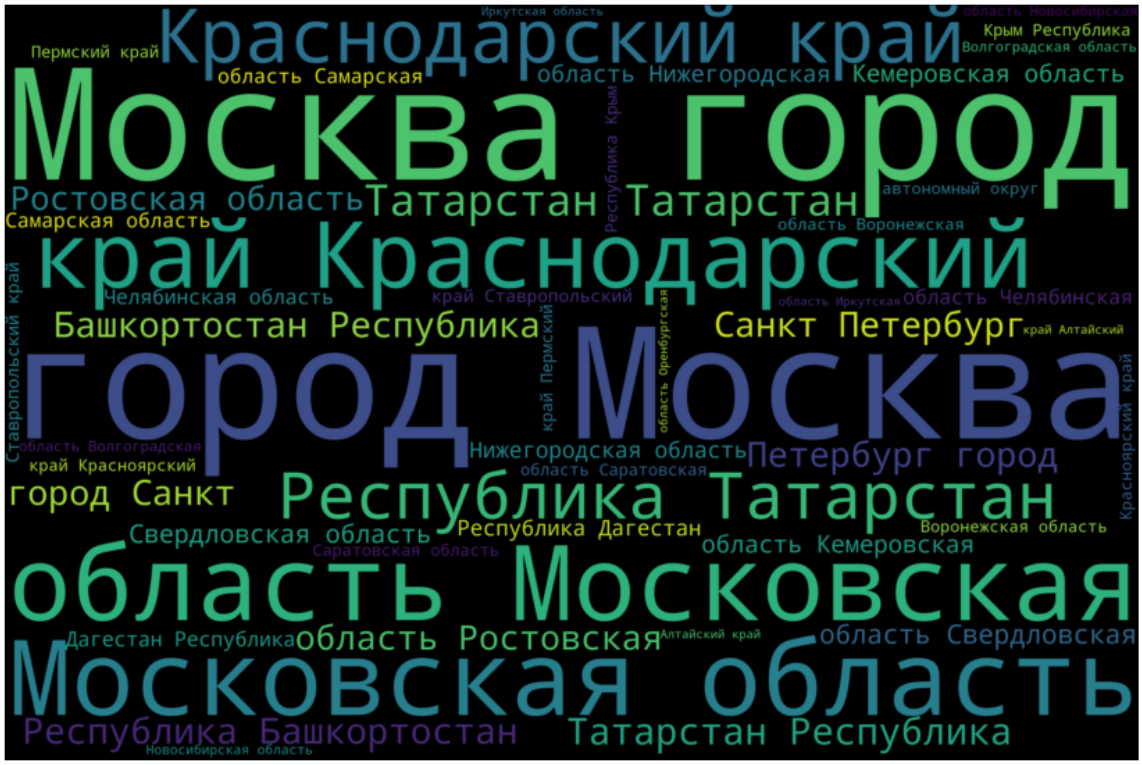
In [54]:
```
img_data = map_grudinin._to_png(5)
img = Image.open(io.BytesIO(img_data))
img
```

Out[54]:



Посмотрим, какие регионы внесли больший вклад в победу Путина (в абсолютных цифрах)

In [55]:
```
text = "";
for i in range(df_regions.shape[0]):
    text += (df_regions['REGION'][i] + ' ')*(df_regions['PUTIN'][i]//100
    text += ' '

wordcloud = WordCloud(width=3000, height=2000, max_words=50).generate(te
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

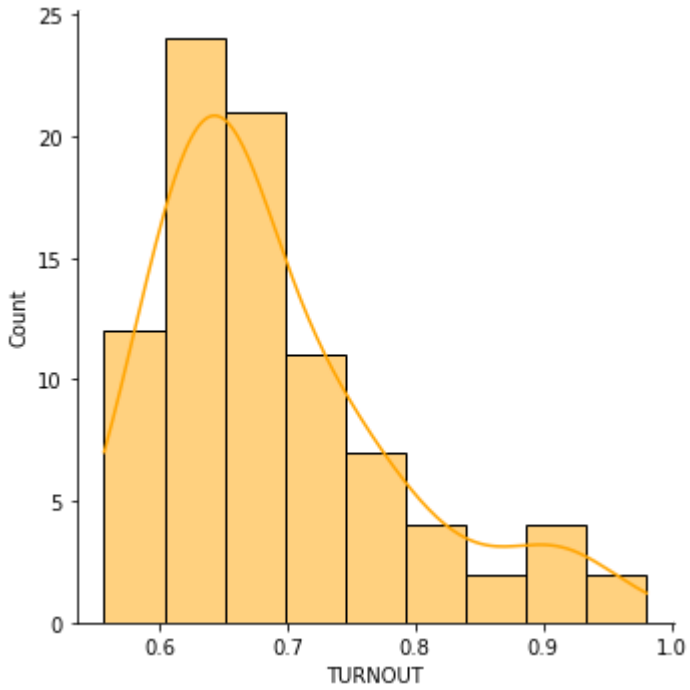


Видим, что большинство голосов победителю достались из Москвы и области, Краснодарского края и республики Татарстан

Посмотрим, наконец, как выглядит распределение явки по регионам и избирательным участкам

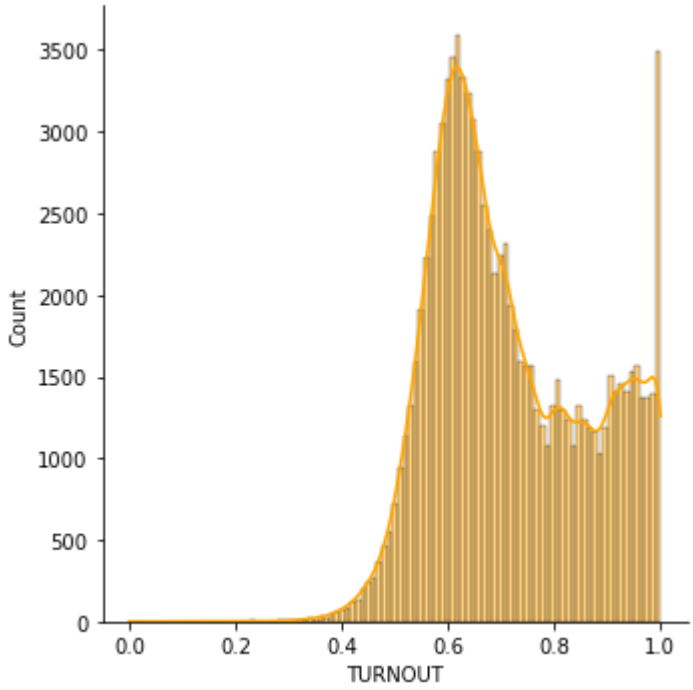
In [56]:
```
sns.displot(data=df_regions, x='TURNOUT', color='orange', kde=True)
```

Out[56]:
```
<seaborn.axisgrid.FacetGrid at 0x7f91a6112ac0>
```



In [57]:
```
sns.displot(data=df, x='TURNOUT', color='orange', kde=True)
```

Out[57]:
```
<seaborn.axisgrid.FacetGrid at 0x7f91a59a9910>
```



Видим увеличение количества участков после 0.9, что не соответствует нормальному распределению. Кроме того, очень заметно большое количество участков с явкой 100 %. Посмотрим на них в таблице

In [58]:
```
df.loc[(df['TURNOUT'] == 1.0) & (df['REGION'] != 'Территория за пределам
```

Out[58]:

	PS_ID	REGION	SUBREGION	N_ALL	N_EARLY	N_OUT	N_INVALID	BABU
456	67	Республика Адыгея (Адыгея)	Красногвардейская	691	0	33	0	
461	72	Республика Адыгея (Адыгея)	Красногвардейская	290	0	9	1	
573	184	Республика Адыгея (Адыгея)	Майкопская городская	434	0	68	5	
574	185	Республика Адыгея (Адыгея)	Майкопская городская	430	0	152	4	
592	203	Республика Адыгея (Адыгея)	Тахтамукайская	593	0	16	6	
...	...	...	...	...	...	...	...	
97524	520	Ямало-Ненецкий автономный округ	Надымская	235	78	6	0	
97525	521	Ямало-Ненецкий автономный округ	Надымская	552	187	8	1	
97535	601	Ямало-Ненецкий автономный округ	Новоуренгойская городская	166	0	96	1	
97671	1111	Ямало-	Тазовская	2778	2727	0	7	

		Ненецкий автономный округ						
97680	1301	Ямало-Ненецкий автономный округ		Ямальская	561	0	33	7

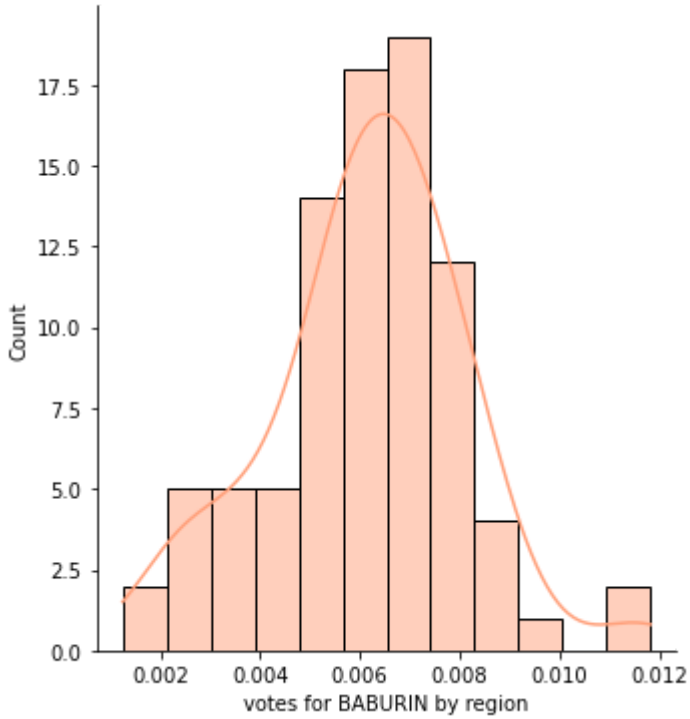
2290 rows × 17 columns

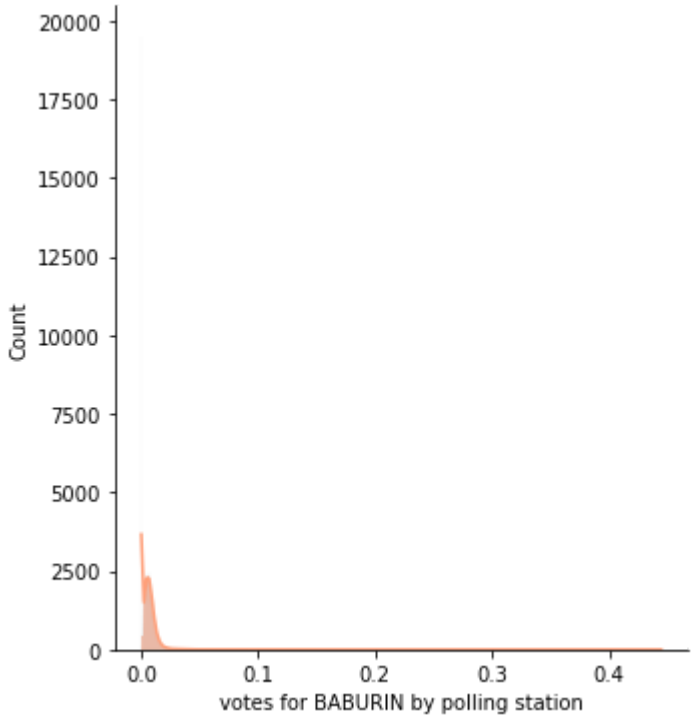
Более чем на 2000 участках явка составила 100 %. Это значительное количество

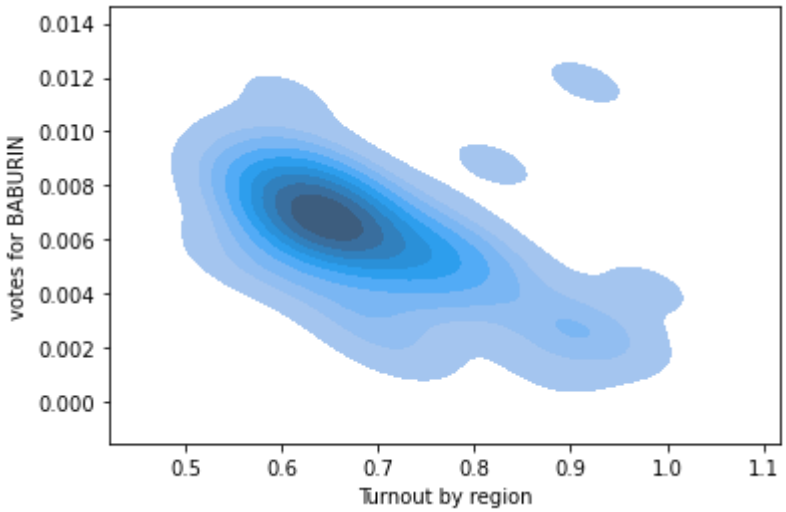
Построим такие же диаграммы для каждого кандидата и добавим ещё одну, показывающую распределение голосов за кандидата в зависимости от явки

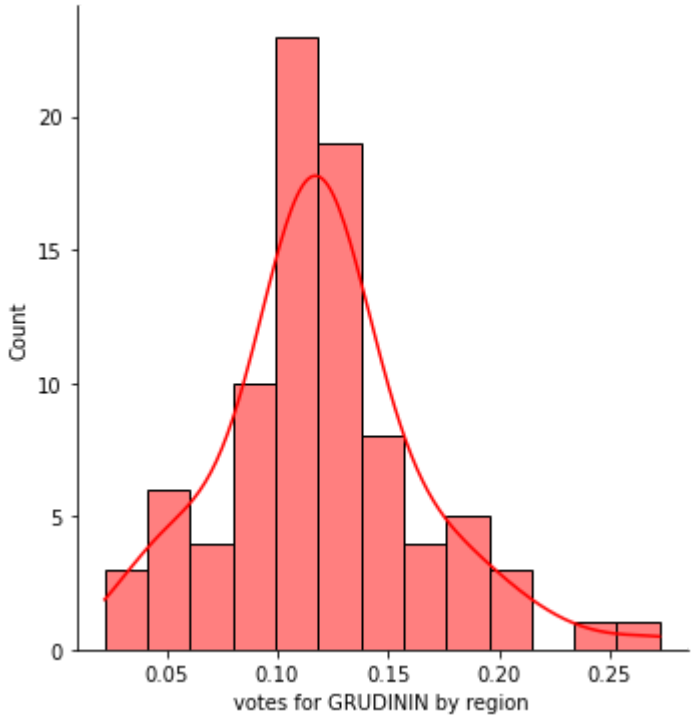
In [59]:
```
def draw_histogram_for_candidate(cand):
    if cand not in CANDS:
        print('You entered incorrect candidate')
        return
    ax = sns.displot(data=df_regions, x=[df_regions[cand][i]/df_regions[
    ax.set(xlabel=f'votes for {cand} by region')
    plt.show()
    ax = sns.displot(data=df, x=[df[cand][i]/df['N_VOTED'][i] if df['N_V
                     color=COLORS[CANDS.index(cand)], kde=True)
    ax.set(xlabel=f'votes for {cand} by polling station')
    plt.show()
    ax = sns.kdeplot(data=df_regions, x='TURNOUT', y=[df_regions[cand][i
    ax.set(xlabel='Turnout by region', ylabel=f'votes for {cand}')
    plt.show()
```

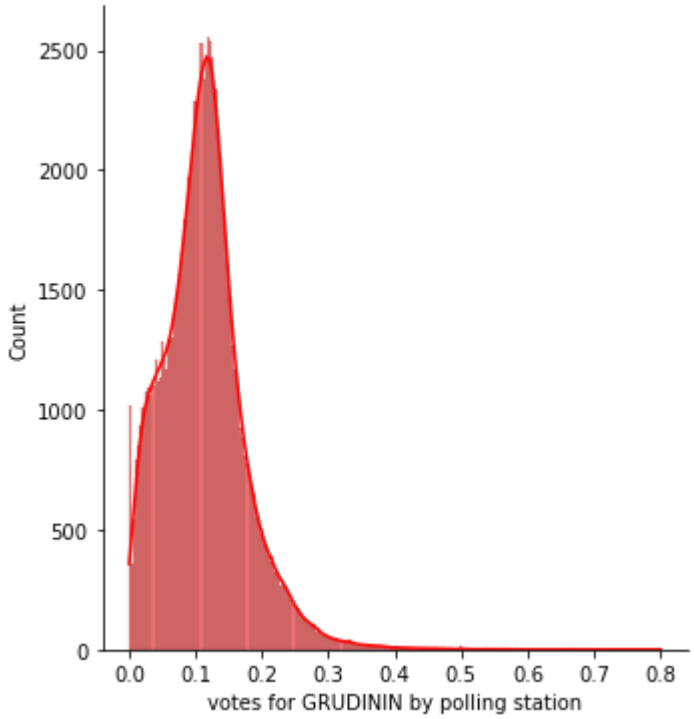
In [60]:
```
for cand in CANDS:
    draw_histogram_for_candidate(cand)
```

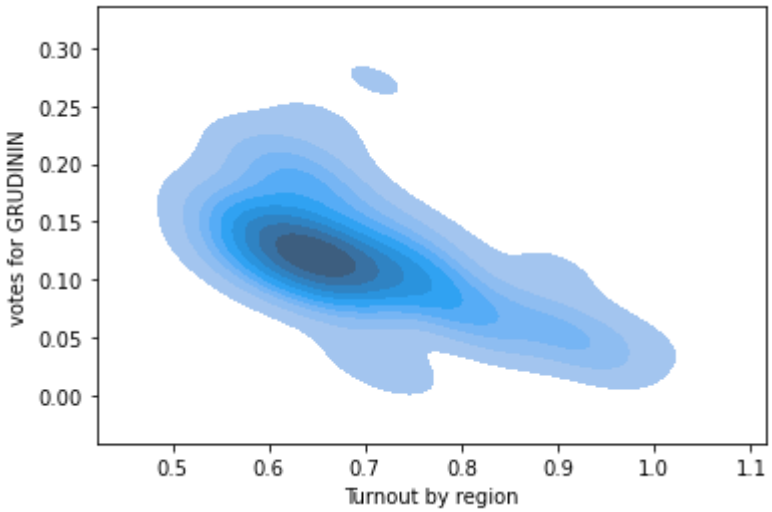


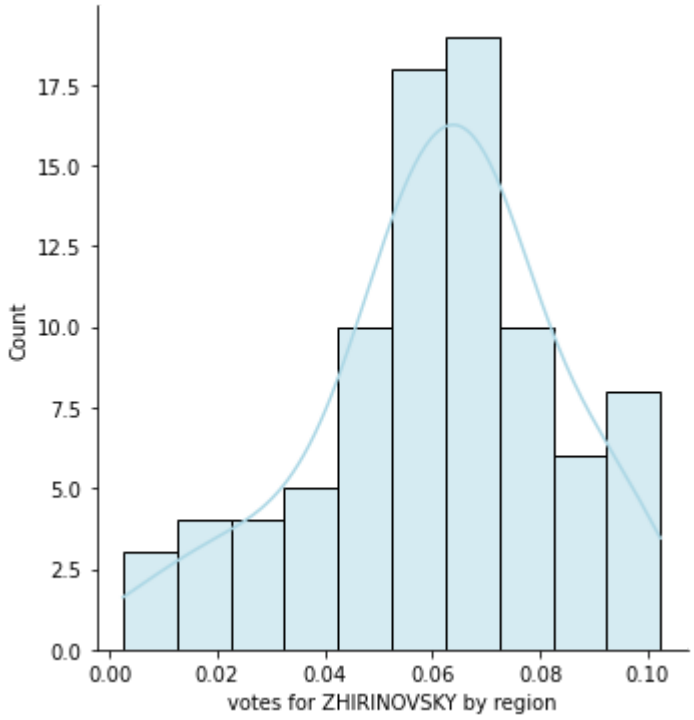

20000
17500
15000
12500
10000
7500
5000
2500
0
Count
0.0
0.1
0.2
0.3
0.4
votes for BABURIN by polling station

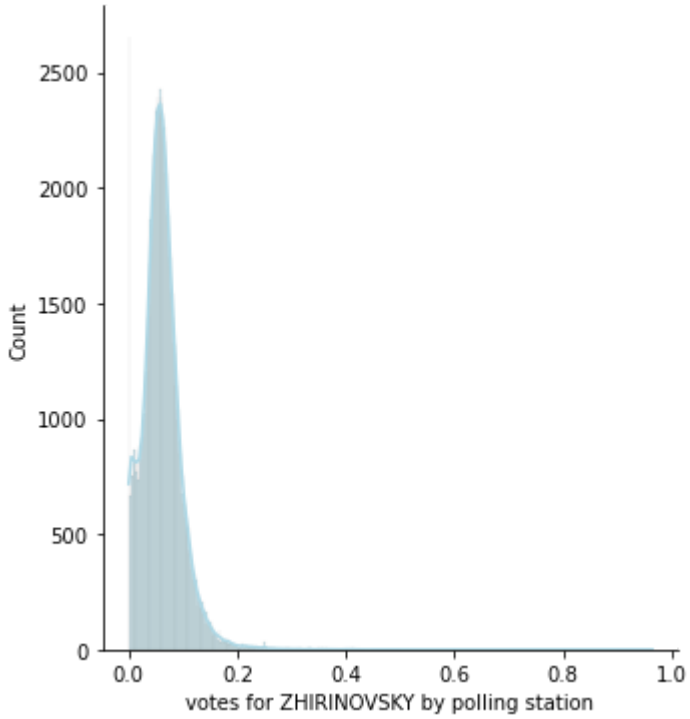

0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
votes for BABURIN
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

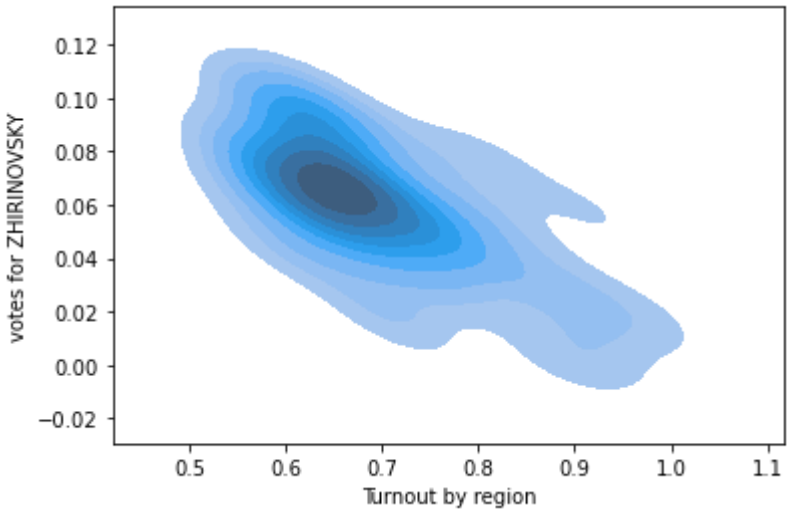

20
15
10
5
0
Count
0.05
0.10
0.15
0.20
0.25
votes for GRUDININ by region

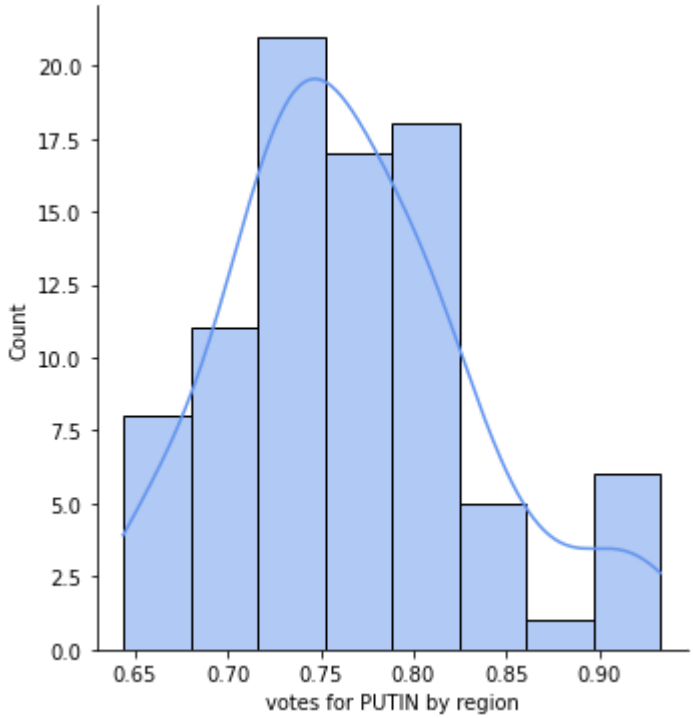

Count
2500
2000
1500
1000
500
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
votes for GRUDININ by polling station

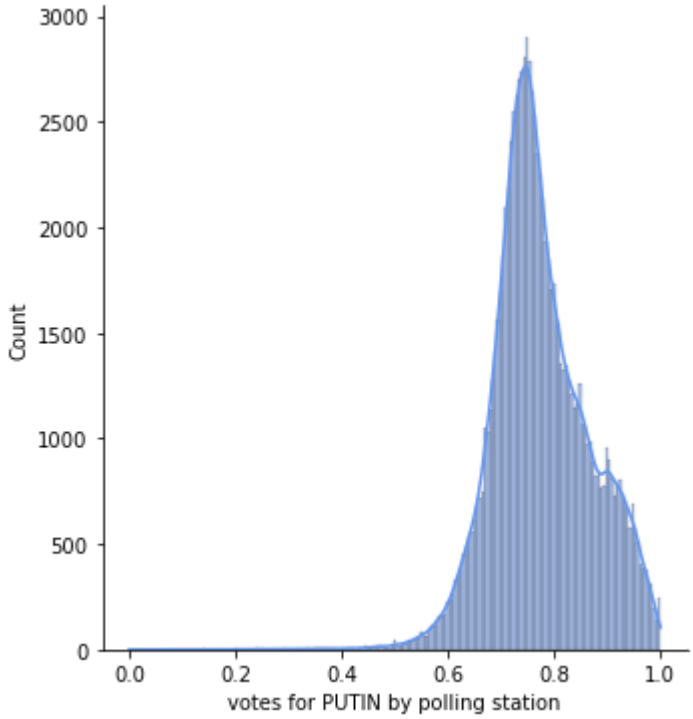

votes for GRUDININ
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

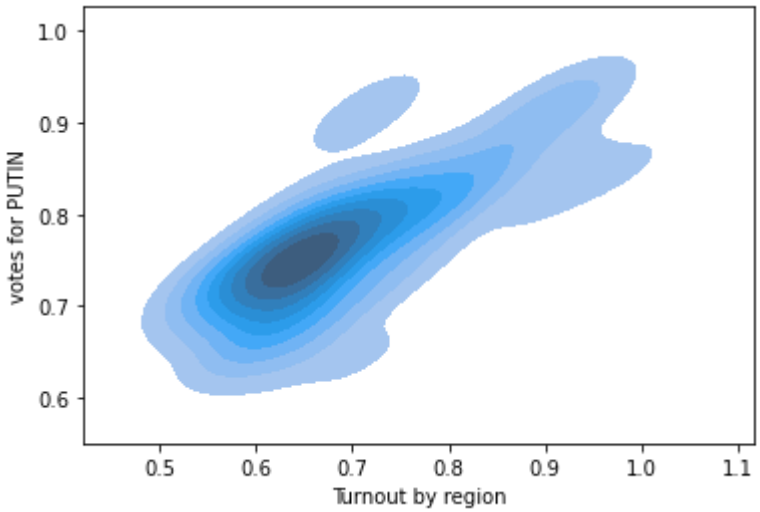

Count
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.00
0.02
0.04
0.06
0.08
0.10
votes for ZHIRINOVSKY by region

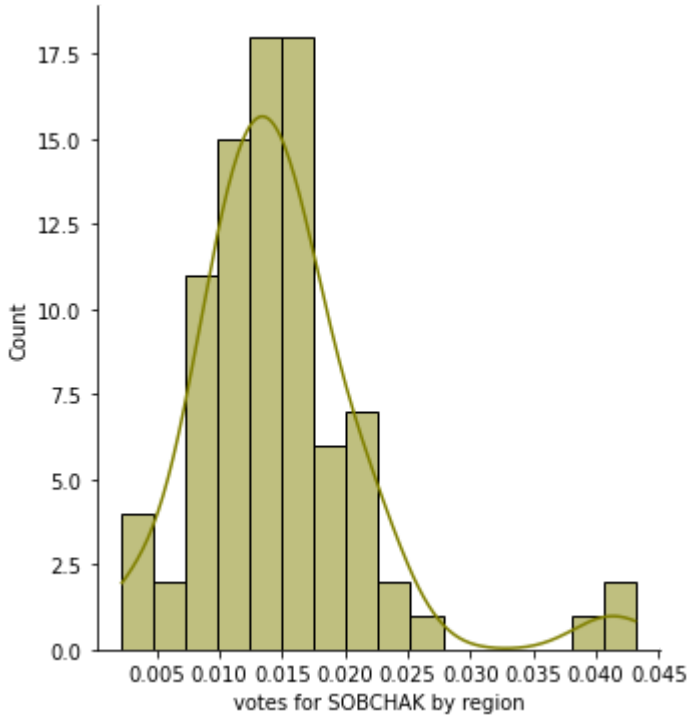

Count
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
votes for ZHIRINOVSKY by polling station

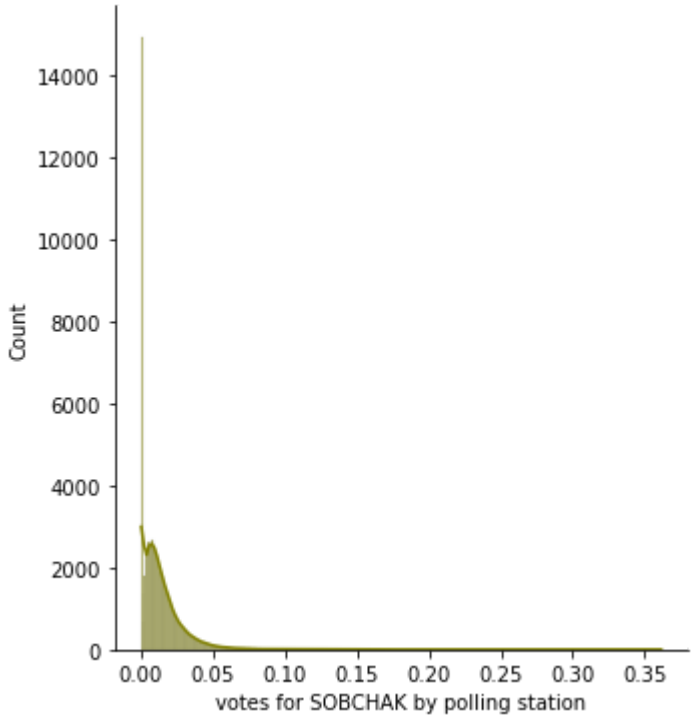

votes for ZHIRINOVSKY
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

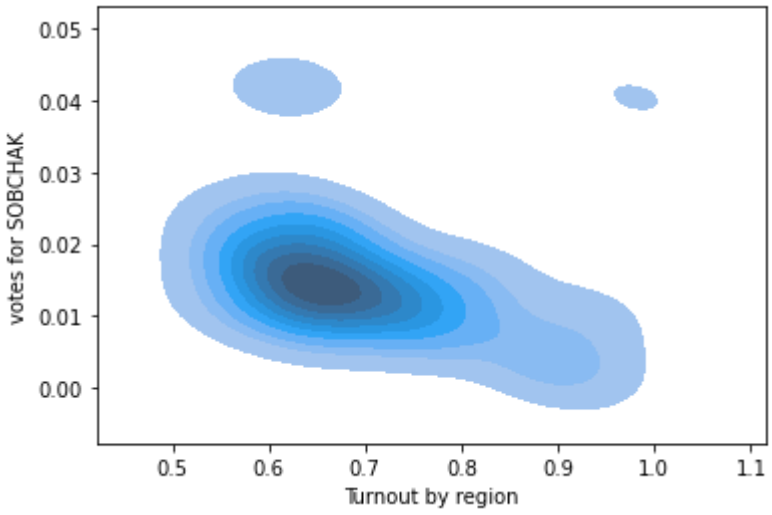

Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.65
0.70
0.75
0.80
0.85
0.90
votes for PUTIN by region

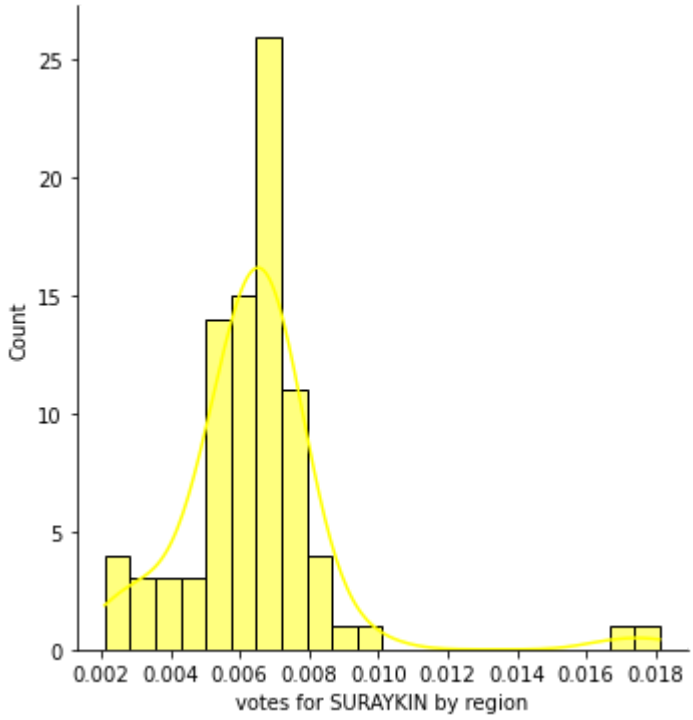

3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
votes for PUTIN by polling station

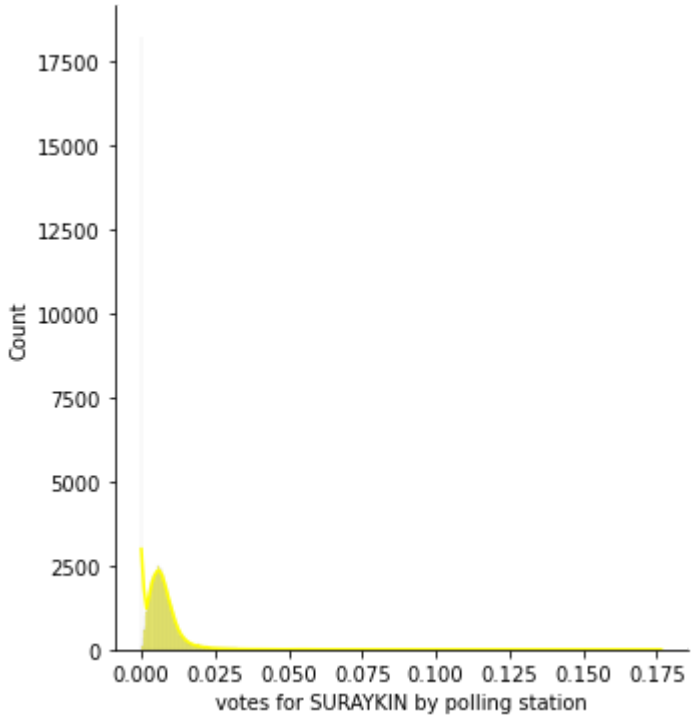

1.0
0.9
0.8
0.7
0.6
votes for PUTIN
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

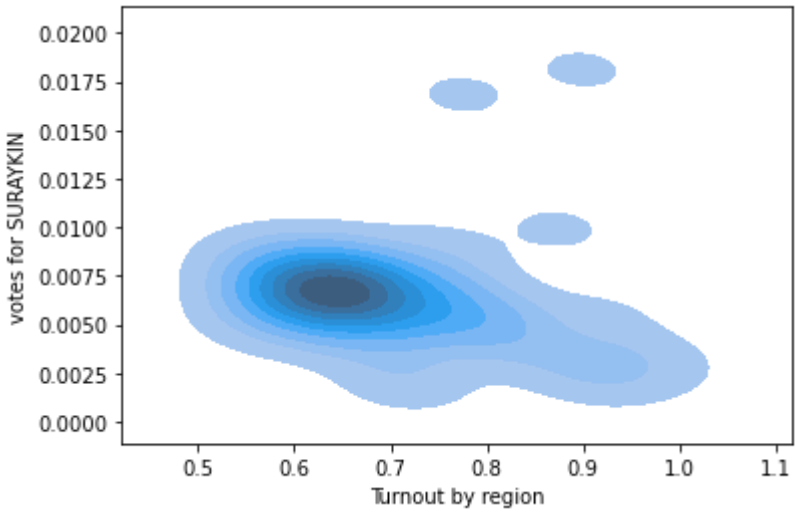

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Count
0.005 0.010 0.015 0.020 0.025 0.030 0.035 0.040 0.045
votes for SOBCHAK by region

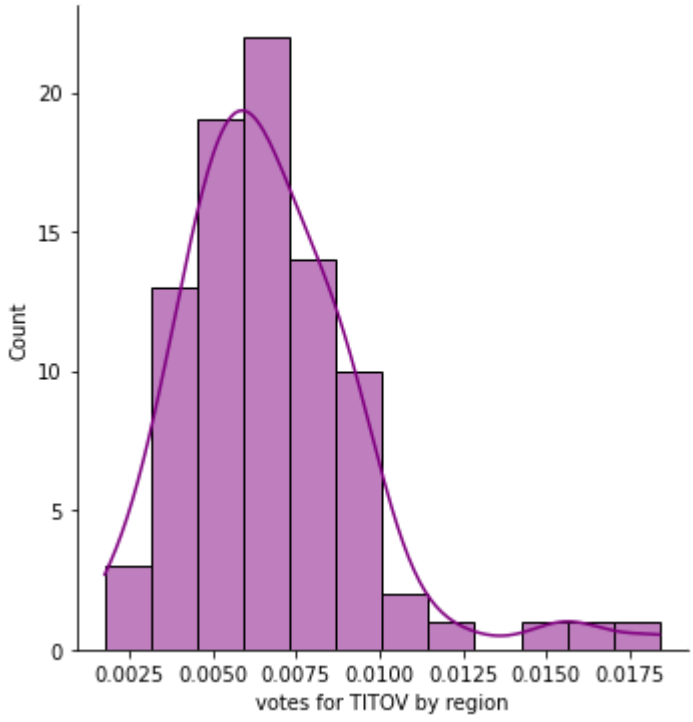

14000
12000
10000
8000
6000
4000
2000
0
Count
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
votes for SOBCHAK by polling station

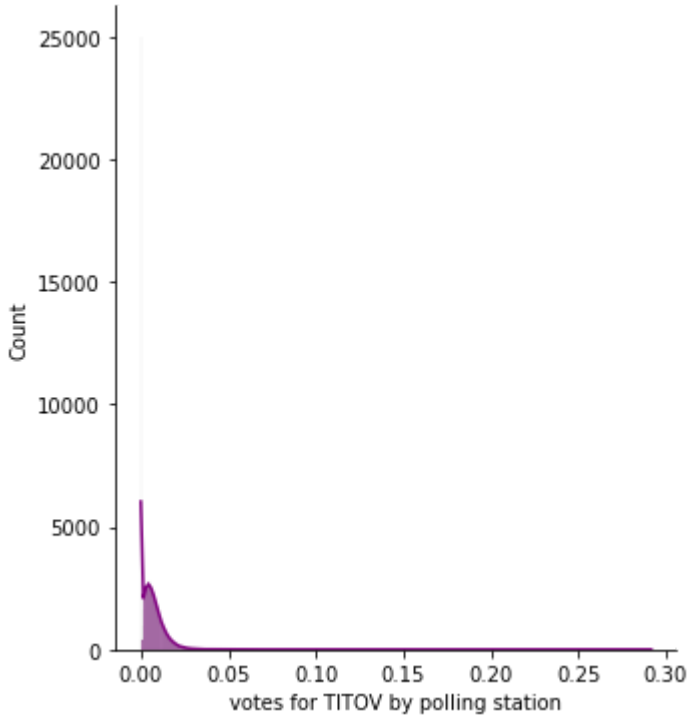

0.05
0.04
0.03
0.02
0.01
0.00
votes for SOBCHAK
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

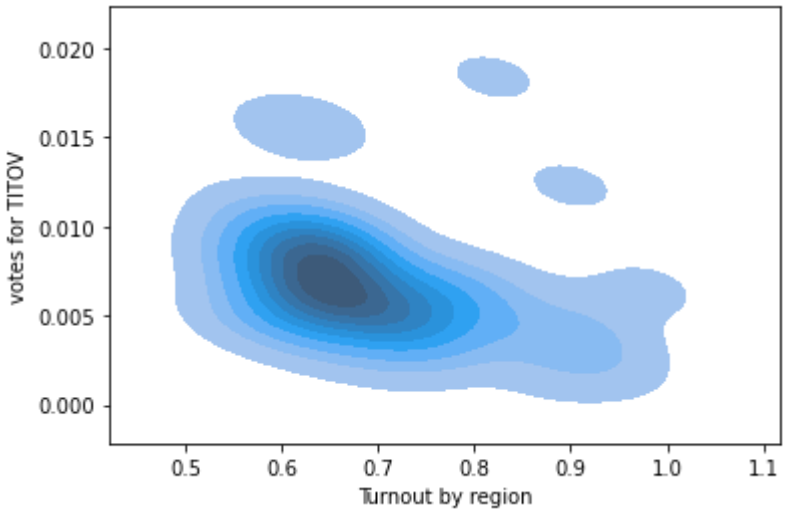

25
20
15
10
5
0
Count
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0.018
votes for SURAYKIN by region

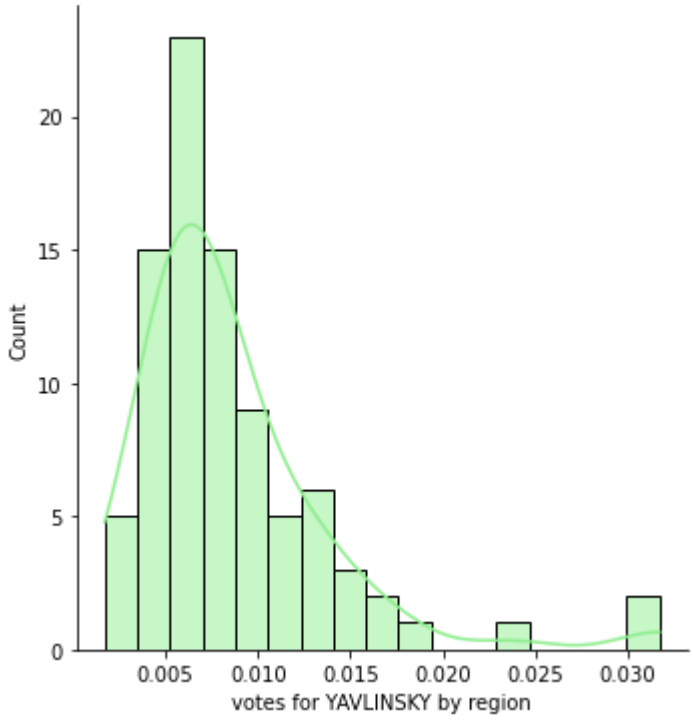

Count
17500
15000
12500
10000
7500
5000
2500
0
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
votes for SURAYKIN by polling station

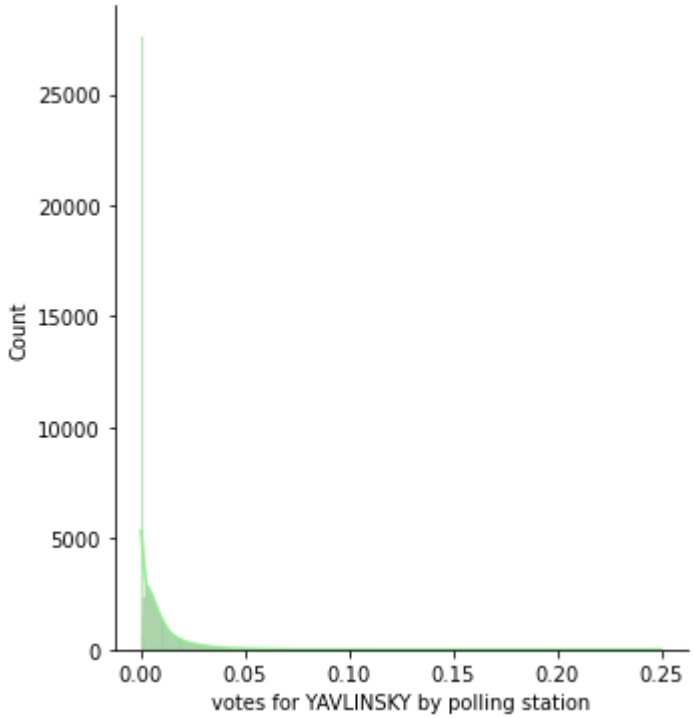

votes for SURAYKIN
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region

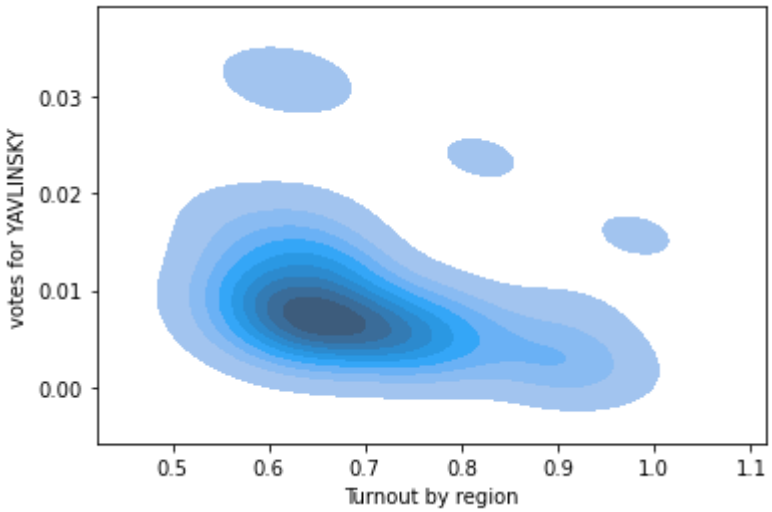

Count
20
15
10
5
0
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
votes for TITOV by region


25000
20000
15000
10000
5000
0
Count
0.00
0.05
0.10
0.15
0.20
0.25
0.30
votes for TITOV by polling station


0.020
0.015
0.010
0.005
0.000
votes for TITOV
0.5
0.6
0.7
0.8
0.9
1.0
1.1
Turnout by region


20
15
10
5
0
Count
0.005
0.010
0.015
0.020
0.025
0.030
votes for YAVLINSKY by region





Видим, что целом распределение по регионам имеет похожую форму у всех кандидатов. На всех графиках мы видим рост последних столбцов, однако он незначителен у всех, кроме Путина, где последний столбец больше предыдущего в 5 раз.\ По последнему графику каждого из кандидатов мы видим, что с увеличением явки падает процент голосов за данного кандидата. У всех, но не у Путина: у него в этом месте наблюдается значительный рост