In [2]:
```
%load_ext autoreload
%autoreload 2
```

In [1]:
```
import calendar
from collections import Counter
from functools import reduce
from operator import itemgetter
from functools import partial

import pandas as pd
import plotly.express as px
import plotly.figure_factory as ff
import plotly.offline as pyo
from mongoengine import connect

from src import settings
from src.data.vacancy import Vacancy
from src.features.clean import remove_html
from src.visualization.statistics import plot_value_counts
```

In [3]:
```
connect(
    host=settings.db_host,
    port=settings.db_port,
    db=settings.db_name
)
```

Out[3]:
```
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True,
read_preference=Primary())
```

In [5]:
```
pyo.init_notebook_mode()
```

In [ ]:
```
df: pd.DataFrame = (
    Vacancy
        .objects
        .to_dataframe(include=[
            '_id',
            'name',
            'description',
            'salary',
            'schedule.name',
            'experience',
            'employment.name',
            'area.name',
            'address.lat',
            'address.lng',
            'address.city',
            'published_at',
            'specializations',
            'employer.name',
            'professional_roles',
            'key_skills',
        ])
)
```

In [ ]:
```
df.set_index('_id', inplace=True)
```

In [ ]:
```
df['description'] = df['description'].map(remove_html)
```

In [ ]:
```
df.columns
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.published_at = pd.to_datetime(df.published_at)

count_by_month = {
    calendar.month_name[month]: sum(df.published_at.dt.month == month) for month in rang
}

px.bar(
    x=count_by_month.keys(),
    y=count_by_month.values(),
    labels={'x': 'Месяц', 'y': 'Количество вакансий'},
    title='Количество вакансий в зависимости от месяца'
)
```

In [ ]:
```
plot_value_counts(
    df['experience.name'],
    x_label='Опыт',
    y_label='Количество вакансий',
    title='Количество вакансий в зависимости от опыта'
).update_xaxes(categoryorder='total descending')
```

In [ ]:
```
plot_value_counts(
    df['schedule.name'],
    x_label='График',
    y_label='Количество вакансий',
    title='Количество вакансий в зависимости от графика работы'
).update_xaxes(categoryorder='total descending')
```

Анализ навыков

In [ ]:
```
key_skills = reduce(set.union, df.key_skills, set())
```

In [ ]:
```
len(key_skills)
```

In [ ]:
```
count_by_key_skill = reduce(Counter.__add__, map(Counter, df.key_skills))
```

In [ ]:
```
ff.create_table([('Навык', 'Количество вакансий')] + count_by_key_skill.most_common(50))
```

Анализ профобластей

In [ ]:
```
df['profarea_names'] = df.specializations.map(lambda specs: list(set(map(itemgetter('pro
```

In [ ]:
```
df.profarea_names.head(10)
```

In [ ]:
```
profareas = reduce(set.union, df.profarea_names, set())
```
In [ ]:
```
len(profareas)
```
In [ ]:
```
count_by_profarea = reduce(Counter.__add__, map(Counter, df.profarea_names))
```
In [ ]:
```
profareas_df = pd.DataFrame(count_by_profarea, index=['Количество вакансий']).T.reset_in
```
In [ ]:
```
ff.create_table(profareas_df)
```
In [ ]:
```
px.bar(
    profareas_df,
    x='Профобласть',
    y='Количество вакансий',
    text_auto='.2s'
).update_xaxes(categoryorder='total descending')
```