

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №1
по курсу «Методы машинного обучения»

Тема: «Разведочный анализ данных. Исследование и визуализация данных.»

ИСПОЛНИТЕЛЬ:

группа ИУ5-21М

Макаров Д.А.
ФИО

подпись

"__" _____ 202__ г.

ПРЕПОДАВАТЕЛЬ:

ФИО

подпись

"__" _____ 202__ г.

Москва - 2020

1) Текстовое описание набора данных

contains information on accidents from each airline

Каждый файл содержит следующие колонки: 'airline', 'avail_seat_km_per_week', 'incidents_85_99', 'fatal_accidents_85_99', 'fatalities_85_99', 'incidents_00_14', 'fatal_accidents_00_14', 'fatalities_00_14'

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [29]:

```
# Будем анализировать данные только на обучающей выборке
data = pd.read_csv('/Users/denis/Downloads/airline-safety.csv', sep=",")
```

2) Основные характеристики датасета

In [30]:

```
# Первые 5 строк датасета
data.head()
```

Out[30]:

	airline	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accidents_00_14
0	Aer Lingus	320906734	2	0	0	0	
1	Aeroflot*	1197672318	76	14	128	6	
2	Aerolineas Argentinas	385803648	6	0	0	1	
3	Aeromexico*	596871813	3	1	64	5	
4	Air Canada	1865253802	2	0	0	2	

In [31]:

```
# Размер датасета - 8143 строк, 7 колонок
data.shape
```

Out[31]:

(56, 8)

In [32]:

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

Всего строк: 56

In [33]:

```
# Список колонок
data.columns
```

Out[33]:

```
Index(['airline', 'avail_seat_km_per_week', 'incidents_85_99',
       'fatal_accidents_85_99', 'fatalities_85_99', 'incidents_00_14',
       'fatal_accidents_00_14', 'fatalities_00_14'],
      dtype='object')
```

In [34]:

```
# Список колонок с типами данных
data.dtypes
```

Out[34]:

```
airline                object
avail_seat_km_per_week    int64
incidents_85_99          int64
fatal_accidents_85_99     int64
fatalities_85_99         int64
incidents_00_14          int64
fatal_accidents_00_14     int64
fatalities_00_14         int64
dtype: object
```

In [35]:

```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
airline - 0
avail_seat_km_per_week - 0
incidents_85_99 - 0
fatal_accidents_85_99 - 0
fatalities_85_99 - 0
incidents_00_14 - 0
fatal_accidents_00_14 - 0
fatalities_00_14 - 0
```

In [36]:

```
# Основные статистические характеристики набора данных
data.describe()
```

Out[36]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accidents_00_14	fatalities_00_14
count	5.600000e+01	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000
mean	1.384621e+09	7.178571	2.178571	112.410714	4.125000	0.660714	5.178571
std	1.465317e+09	11.035656	2.861069	146.691114	4.544977	0.858684	11.035656
min	2.593733e+08	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.740362e+08	2.000000	0.000000	0.000000	1.000000	0.000000	2.000000
50%	8.029089e+08	4.000000	1.000000	48.500000	3.000000	0.000000	4.000000
75%	1.847239e+09	8.000000	3.000000	184.250000	5.250000	1.000000	8.000000
max	7.139291e+09	76.000000	14.000000	535.000000	24.000000	3.000000	53.000000

In [39]:

```
# Определим уникальные значения для целевого признака
data['incidents_85_99'].unique()
```

```
Out[39]:
```

```
array([[ 2, 76,  6,  3, 14,  5,  7, 21,  1,  4,  0, 12, 24,  8, 25, 10, 19,
        16,  9])
```

3) Визуальное исследование датасета

Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

Диаграмма рассеяния

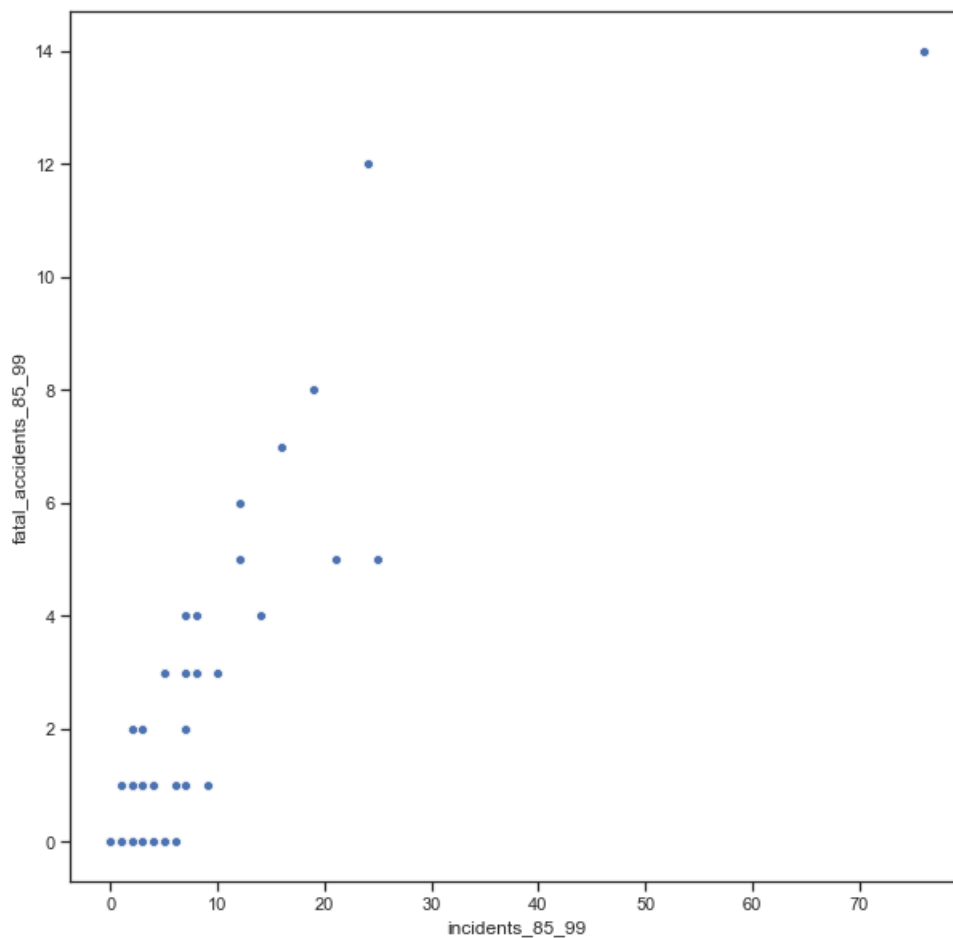
Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости. Не предполагается, что значения упорядочены (например, по времени).

```
In [40]:
```

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='incidents_85_99', y='fatal_accidents_85_99', data=data)
```

```
Out[40]:
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12c2699b0>
```



Можно видеть что между полями Humidity и HumidityRatio присутствует почти линейная зависимость.

Посмотрим насколько на эту зависимость влияет целевой признак.

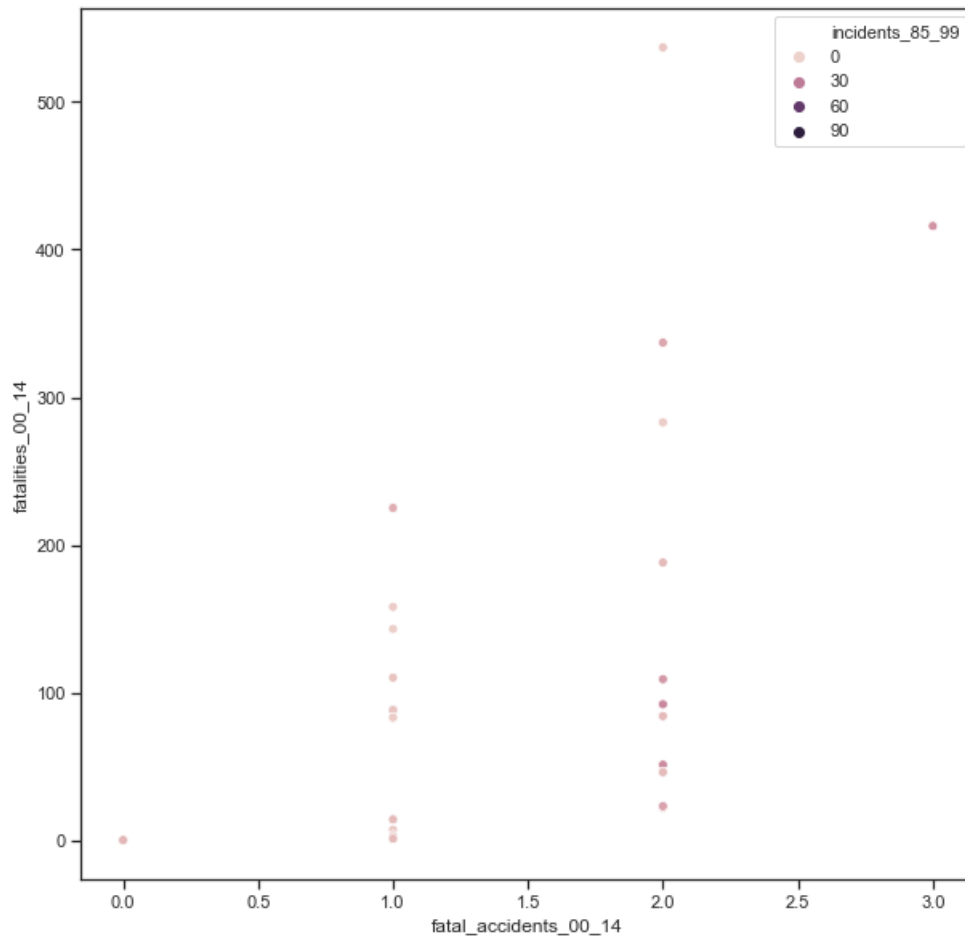
```
In [72]:
```

```
fig, ax = plt.subplots(figsize=(10,10))
```

```
sns.scatterplot(ax=ax, x='fatal_accidents_00_14', y='fatalities_00_14', data=data, hue='incidents_85_99')
```

Out[72]:

<matplotlib.axes._subplots.AxesSubplot at 0x149b9cf28>



Гистограмма

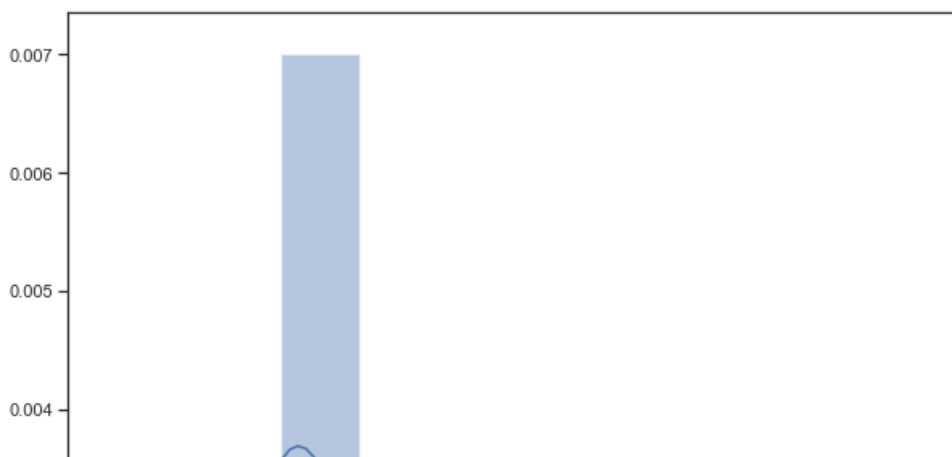
Позволяет оценить плотность вероятности распределения данных.

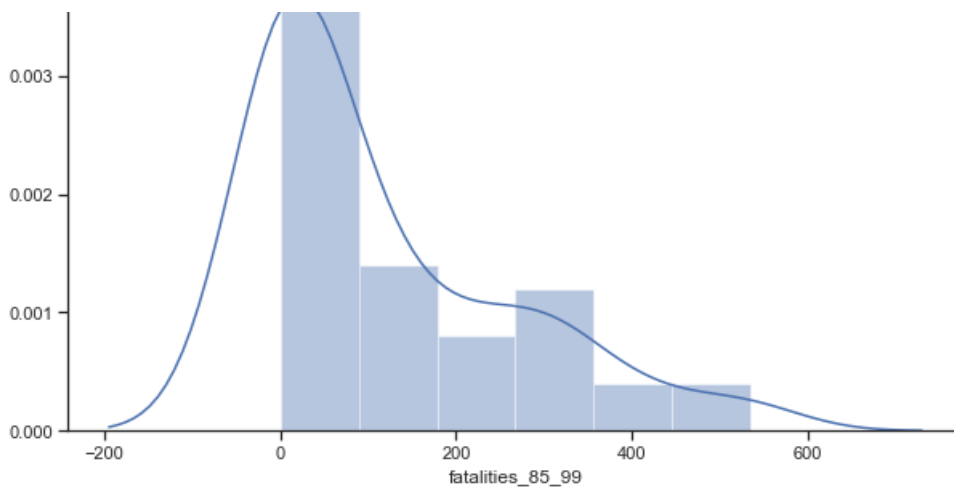
In [44]:

```
fig, ax = plt.subplots(figsize=(10,10))  
sns.distplot(data['fatalities_85_99'])
```

Out[44]:

<matplotlib.axes._subplots.AxesSubplot at 0x12b800c50>





Jointplot

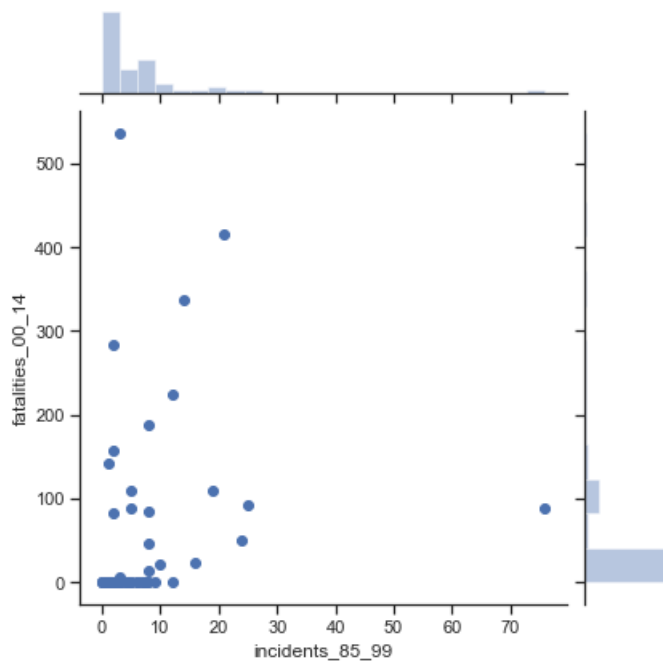
Комбинация гистограмм и диаграмм рассеивания.

In [48]:

```
sns.jointplot(x='incidents_85_99', y='fatalities_00_14', data=data)
```

Out[48]:

<seaborn.axisgrid.JointGrid at 0x142b05978>

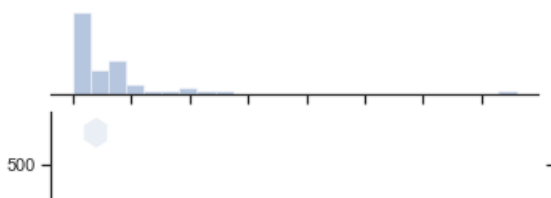


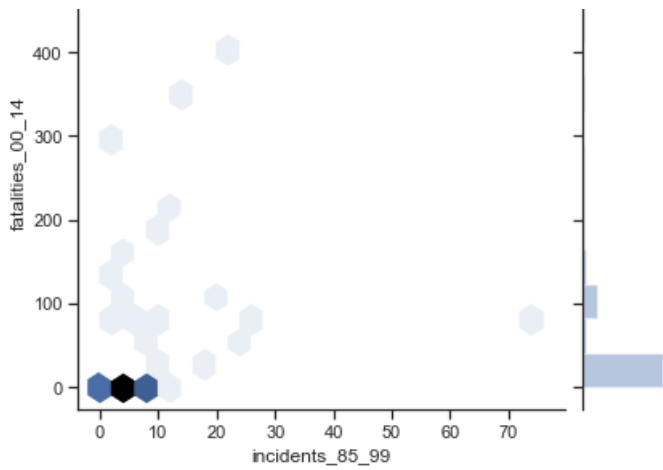
In [49]:

```
sns.jointplot(x='incidents_85_99', y='fatalities_00_14', data=data, kind="hex")
```

Out[49]:

<seaborn.axisgrid.JointGrid at 0x142da0278>



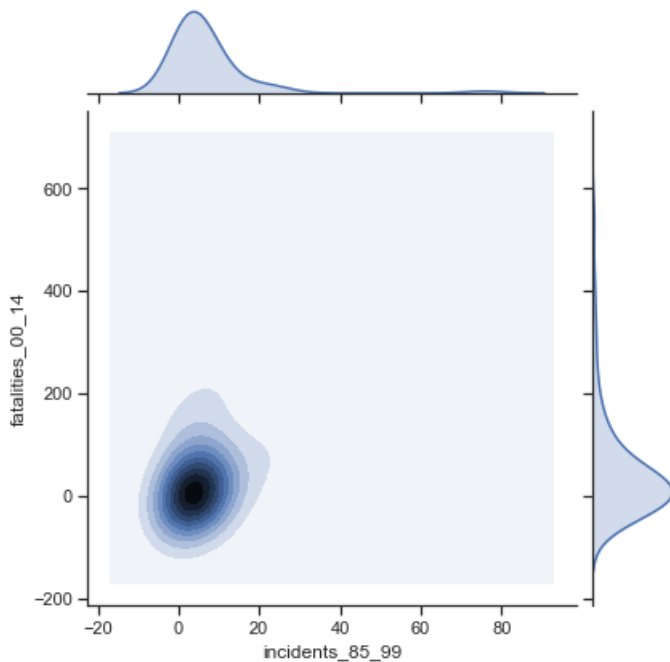


In [50]:

```
sns.jointplot(x='incidents_85_99', y='fatalities_00_14', data=data, kind="kde")
```

Out[50]:

<seaborn.axisgrid.JointGrid at 0x142f90940>



"Парные диаграммы"

Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

Выводится матрица графиков. На пересечении строки и столбца, которые соответствуют двум показателям, строится диаграмма рассеивания. В главной диагонали матрицы строятся гистограммы распределения соответствующих показателей.

In []:

```
sns.pairplot(data)
```

С помощью параметра "hue" возможна группировка по значениям какого-либо признака.

In [52]:

```
sns.pairplot(data, hue="avail_seat_km_per_week")
```

/Users/denis/Documents/MLlabs/env/lib/python3.6/site-packages/seaborn/distributions.py:288:
UserWarning: Data must have variance to compute a kernel density estimate

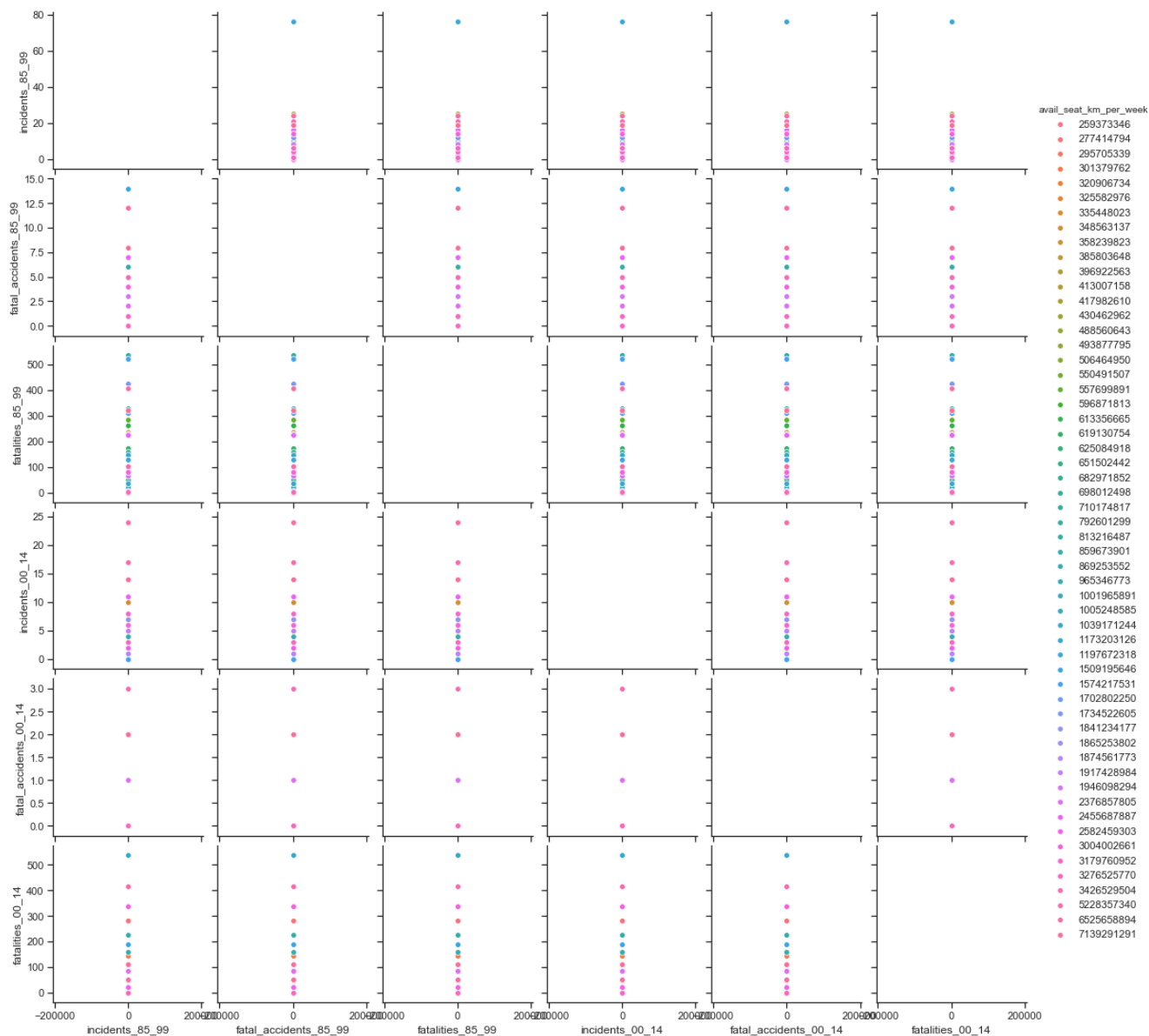
```

/Users/denis/Documents/MLlabs/env/lib/python3.6/site-packages/seaborn/distributions.py:288:
UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)
/Users/denis/Documents/MLlabs/env/lib/python3.6/site-packages/seaborn/distributions.py:288:
UserWarning: Data must have variance to compute a kernel density estimate.
  warnings.warn(msg, UserWarning)

```

Out[52]:

<seaborn.axisgrid.PairGrid at 0x145756710>



Ящик с усами

Отображает одномерное распределение вероятности.

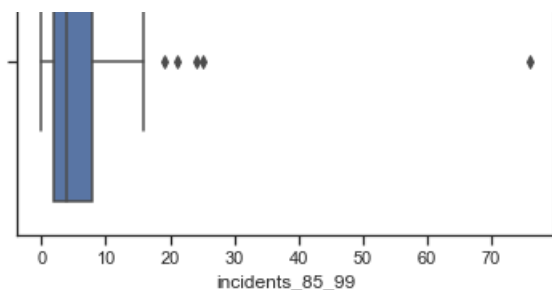
In [53]:

```
sns.boxplot(x=data['incidents_85_99'])
```

Out[53]:

<matplotlib.axes._subplots.AxesSubplot at 0x148eb27b8>



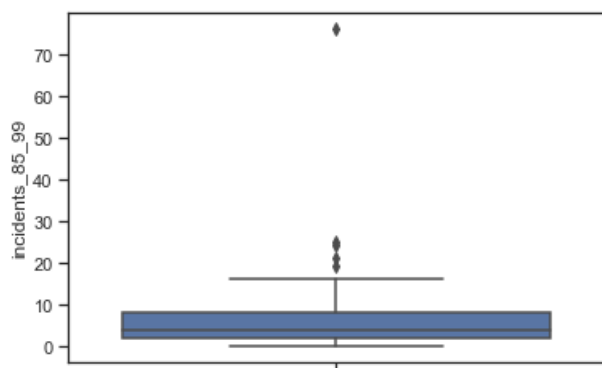


In [54]:

```
# По вертикали
sns.boxplot(y=data['incidents_85_99'])
```

Out[54]:

<matplotlib.axes._subplots.AxesSubplot at 0x14943fe48>

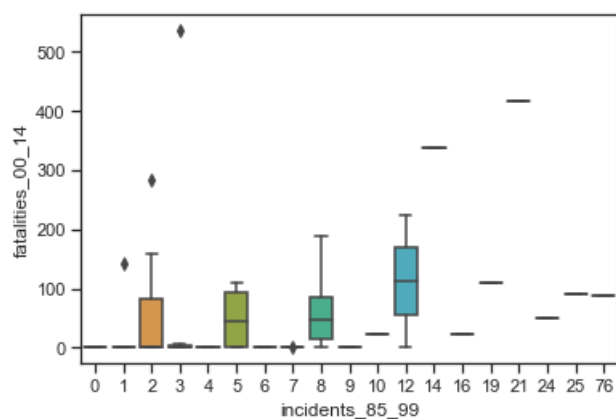


In [56]:

```
# Распределение параметра Humidity сгруппированные по Occurancy.
sns.boxplot(x='incidents_85_99', y='fatalities_00_14', data=data)
```

Out[56]:

<matplotlib.axes._subplots.AxesSubplot at 0x148e1da90>



Violin plot

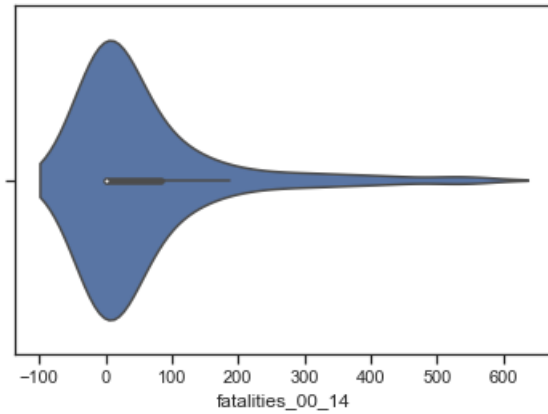
Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности - https://en.wikipedia.org/wiki/Kernel_density_estimation

In [57]:

```
sns.violinplot(x=data['fatalities_00_14'])
```

Out[57]:

<matplotlib.axes._subplots.AxesSubplot at 0x1479f4eb8>

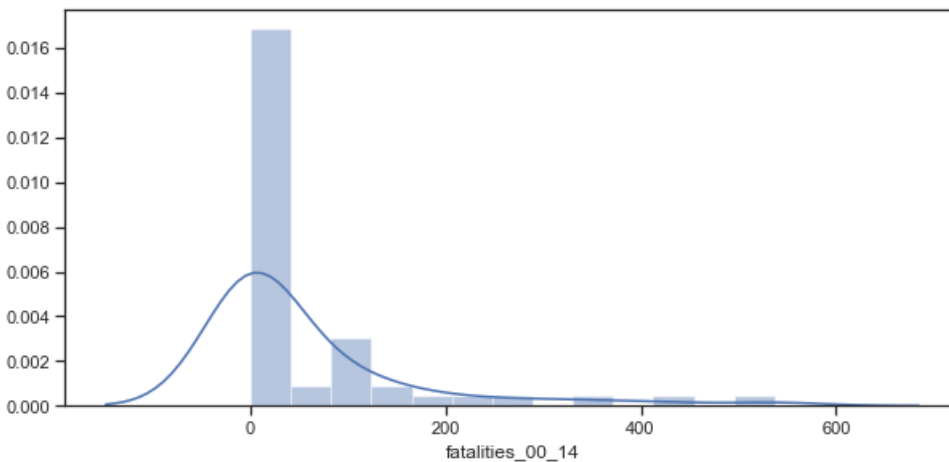
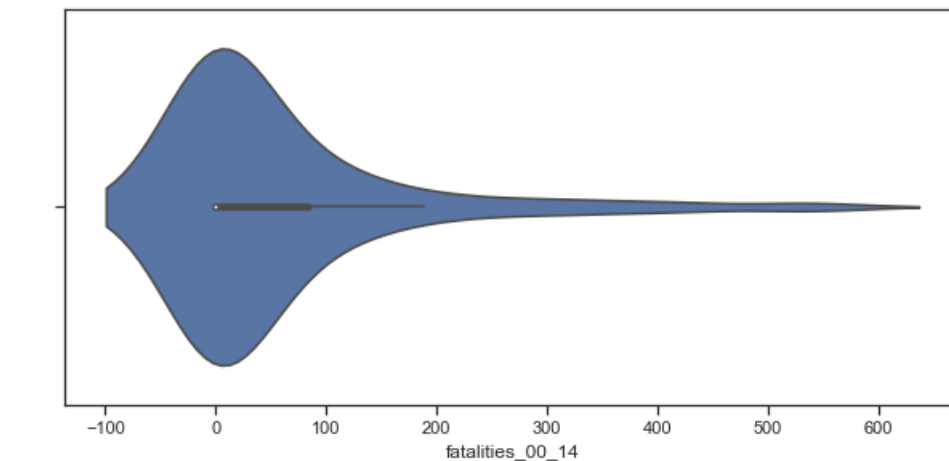


In [58]:

```
fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=data['fatalities_00_14'])
sns.distplot(data['fatalities_00_14'], ax=ax[1])
```

Out[58]:

<matplotlib.axes._subplots.AxesSubplot at 0x14933e6a0>



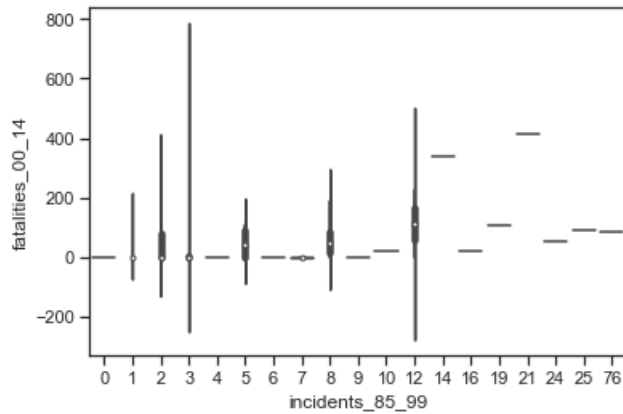
Из приведенных графиков видно, что violinplot действительно показывает распределение плотности.

In [59]:

```
# Распределение параметра Humidity сгруппированные по Occurance.
sns.violinplot(x='incidents_85_99', y='fatalities_00_14', data=data)
```

Out[59]:

<matplotlib.axes._subplots.AxesSubplot at 0x148f493c8>

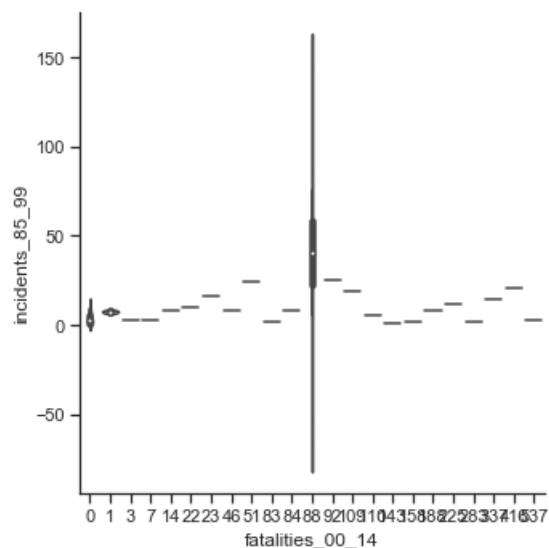


In [60]:

```
sns.catplot(y='incidents_85_99', x='fatalities_00_14', data=data, kind="violin", split=True)
```

Out[60]:

<seaborn.axisgrid.FacetGrid at 0x14737a320>



4) Информация о корреляции признаков

In [61]:

```
data.corr()
```

Out[61]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accidents_00_14
avail_seat_km_per_week	1.000000	0.279538	0.468300	0.209835	0.725917	0.375673
incidents_85_99	0.279538	1.000000	0.856991	0.274394	0.403009	0.390249
fatal_accidents_85_99	0.468300	0.856991	1.000000	0.540866	0.572923	0.498758
fatalities_85_99	0.209835	0.274394	0.540866	1.000000	0.273696	0.228180
incidents_00_14	0.725917	0.403009	0.572923	0.273696	1.000000	0.598071
fatal_accidents_00_14	0.375673	0.390249	0.498758	0.228180	0.598071	1.000000

	fatalities_00_14	0.228484	0.195337	0.186985	0.046979	0.282009	
	avail_seat_km_per_week		incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accide

Корреляционная матрица содержит коэффициенты корреляции между всеми парами признаков.

Корреляционная матрица симметрична относительно главной диагонали. На главной диагонали расположены единицы (корреляция признака самого с собой).

Описание метода corr - <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html>

По умолчанию при построении матрицы используется коэффициент корреляции [Пирсона](#). Возможно также построить корреляционную матрицу на основе коэффициентов корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

In [62]:

```
data.corr(method='pearson')
```

Out[62]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accide
avail_seat_km_per_week	1.000000	0.279538	0.468300	0.209835	0.725917	
incidents_85_99	0.279538	1.000000	0.856991	0.274394	0.403009	
fatal_accidents_85_99	0.468300	0.856991	1.000000	0.540866	0.572923	
fatalities_85_99	0.209835	0.274394	0.540866	1.000000	0.273696	
incidents_00_14	0.725917	0.403009	0.572923	0.273696	1.000000	
fatal_accidents_00_14	0.375673	0.390249	0.498758	0.228180	0.598071	
fatalities_00_14	0.228484	0.195337	0.186985	0.046979	0.282009	

In [63]:

```
data.corr(method='kendall')
```

Out[63]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accide
avail_seat_km_per_week	1.000000	0.199278	0.173397	0.112973	0.345102	
incidents_85_99	0.199278	1.000000	0.709511	0.510457	0.375328	
fatal_accidents_85_99	0.173397	0.709511	1.000000	0.706476	0.265592	
fatalities_85_99	0.112973	0.510457	0.706476	1.000000	0.169407	
incidents_00_14	0.345102	0.375328	0.265592	0.169407	1.000000	
fatal_accidents_00_14	0.133363	0.456350	0.411555	0.241745	0.479243	
fatalities_00_14	0.104203	0.368731	0.325859	0.183673	0.388863	

In [64]:

```
data.corr(method='spearman')
```

Out[64]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14	fatal_accide
avail_seat_km_per_week	1.000000	0.276897	0.231285	0.152488	0.461015	
incidents_85_99	0.276897	1.000000	0.819731	0.658838	0.495822	
fatal_accidents_85_99	0.231285	0.819731	1.000000	0.836670	0.346280	
fatalities_85_99	0.152488	0.658838	0.836670	1.000000	0.218159	
incidents_00_14	0.461015	0.495822	0.346280	0.218159	1.000000	
fatal_accidents_00_14	0.168369	0.551862	0.493827	0.311225	0.556878	

fatalities_00_14 avail_seat_km_per_week incidents_85_99 fatal_accidents_85_99 fatalities_85_99 incidents_00_14 fatal_accidents_00_14

В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

Для визуализации корреляционной матрицы будем использовать "тепловую карту" heatmap которая показывает степень корреляции различными цветами.

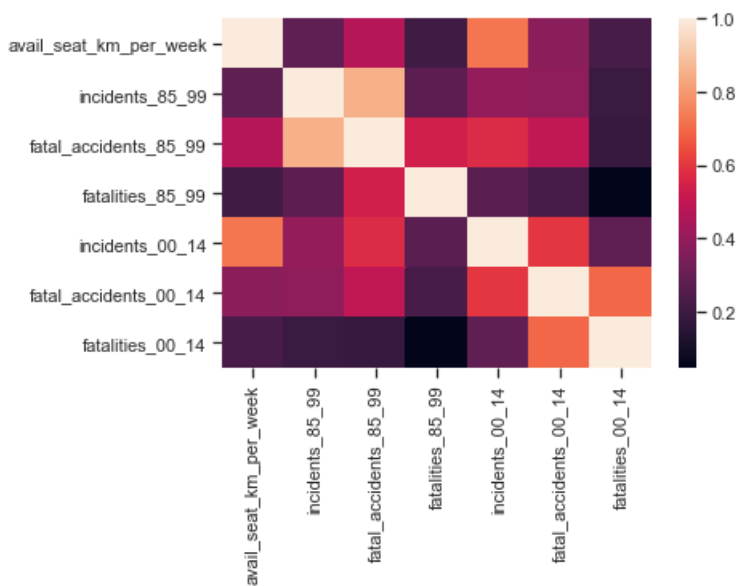
Используем метод heatmap библиотеки seaborn - <https://seaborn.pydata.org/generated/seaborn.heatmap.html>

In [65]:

```
sns.heatmap(data.corr())
```

Out[65]:

<matplotlib.axes._subplots.AxesSubplot at 0x1450f2be0>

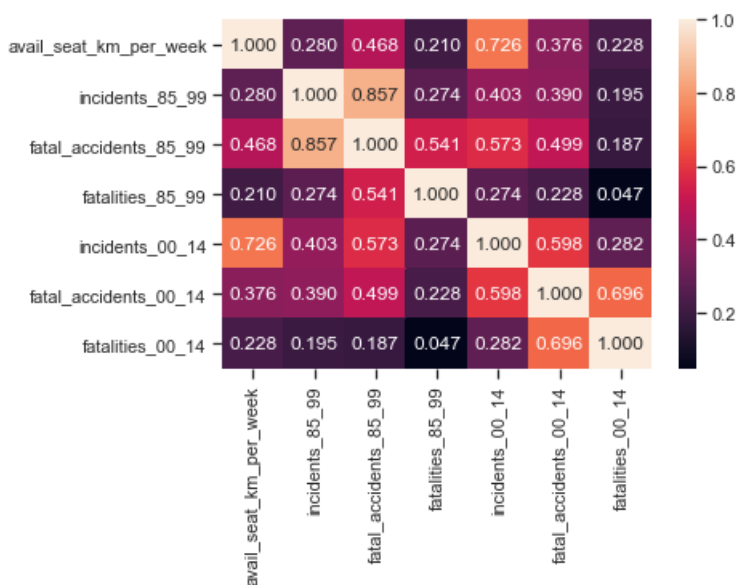


In [66]:

```
# Вывод значений в ячейках
sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

Out[66]:

<matplotlib.axes._subplots.AxesSubplot at 0x147d9fba8>

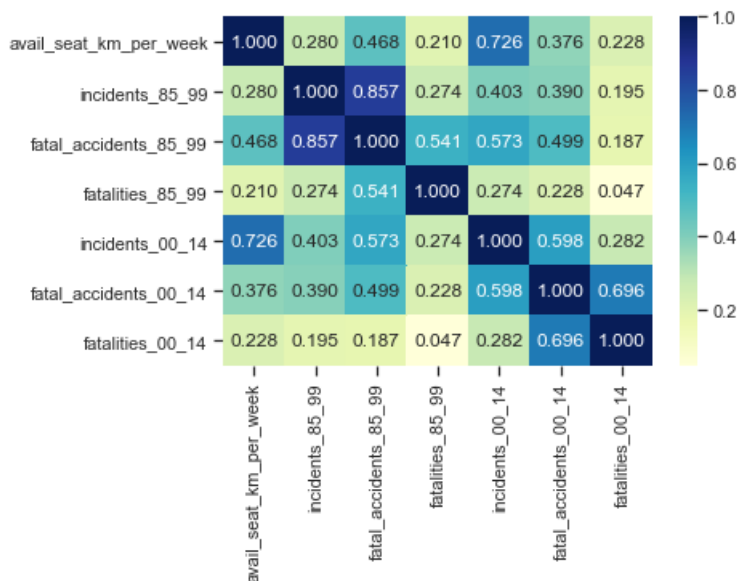


In [67]:

```
# Изменение цветовой гаммы
sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

Out[67]:

<matplotlib.axes._subplots.AxesSubplot at 0x14939a748>

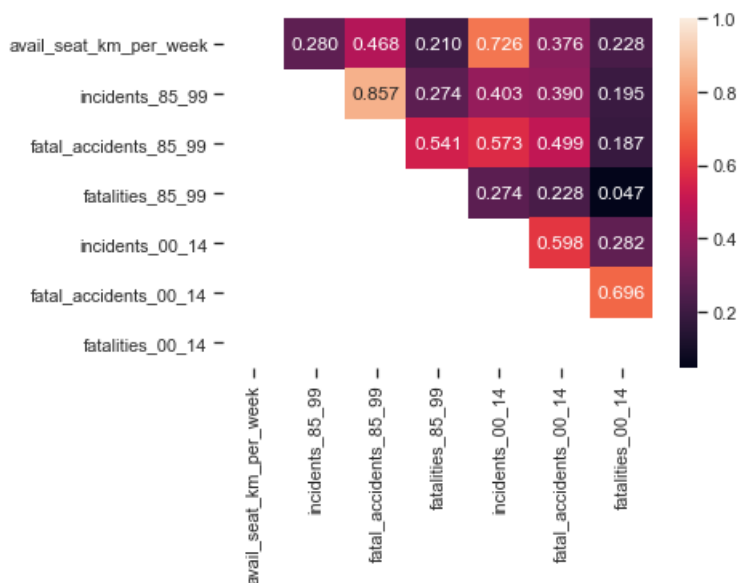


In [68]:

```
# Треугольный вариант матрицы
mask = np.zeros_like(data.corr(), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt='.3f')
```

Out[68]:

<matplotlib.axes._subplots.AxesSubplot at 0x148a5d048>



In [73]:

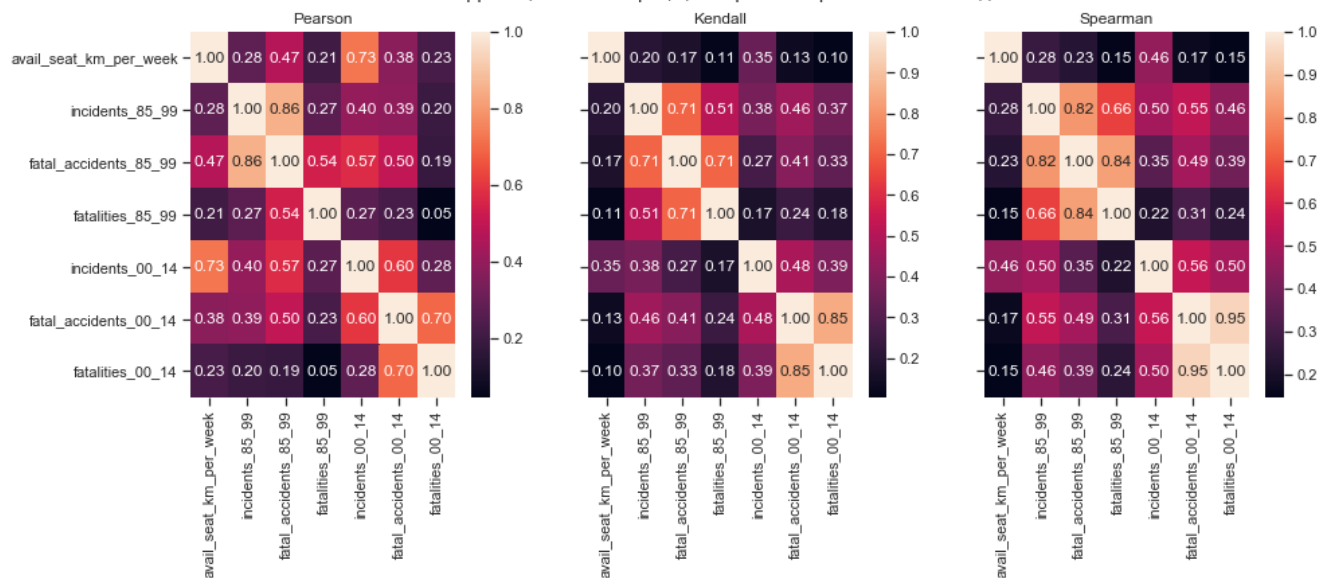
```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(data.corr(method='pearson'), ax=ax[0], annot=True, fmt='.2f')
```

```

sns.heatmap(data.corr(method='kendall'), ax=ax[1], annot=True, fmt='.2f')
sns.heatmap(data.corr(method='spearman'), ax=ax[2], annot=True, fmt='.2f')
fig.suptitle('Корреляционные матрицы, построенные различными методами')
ax[0].title.set_text('Pearson')
ax[1].title.set_text('Kendall')
ax[2].title.set_text('Spearman')

```

Корреляционные матрицы, построенные различными методами



In []: