НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Факультет математики

Ковалев Евгений, Сухарев Иван

Домашняя работа №5

Частые (под)последовательности

3 курс, майнор «Интеллектуальный анализ данных»,
группа ИАД-4

Москва, 2017 г.

1. Поиск частых событий

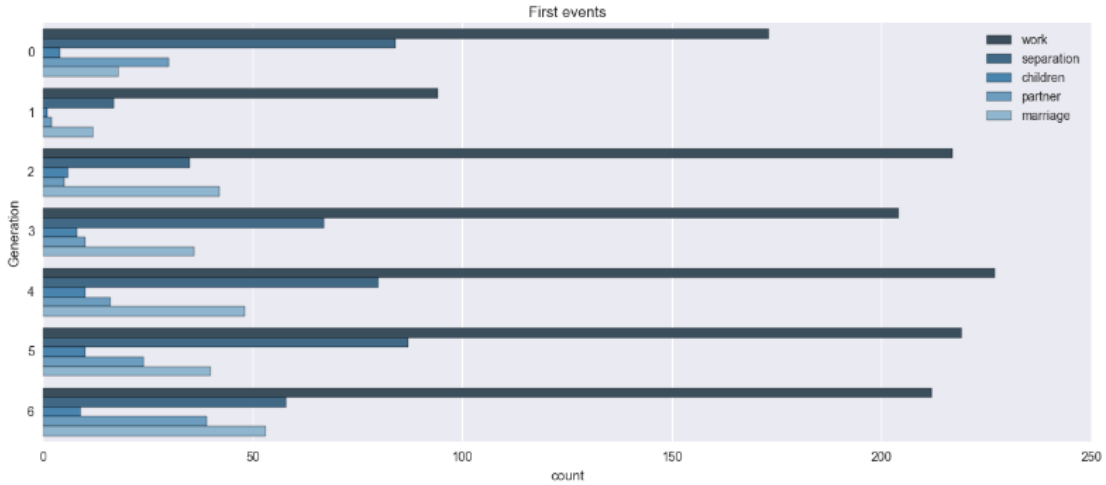
Код для предобработки данных на Python:

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
import seaborn as sns

sns.set_style('darkgrid')

%matplotlib inline
```

In [2]:
```
with open('socAttrAndSeqFusion(full).txt') as f:
    data = f.readlines()
data = list(map(lambda x: re.sub('-1|-2|\n', '', x).split(), data))
data = [x for x in data if len(x) > 4]
zipped_data = list(zip(*data))
first_events = zipped_data[4]
generations = zipped_data[1]
generations = list(map(lambda x: re.sub('generation=', '', x), generations))
```

1) У всех поколений самым частым первым событием является работа.

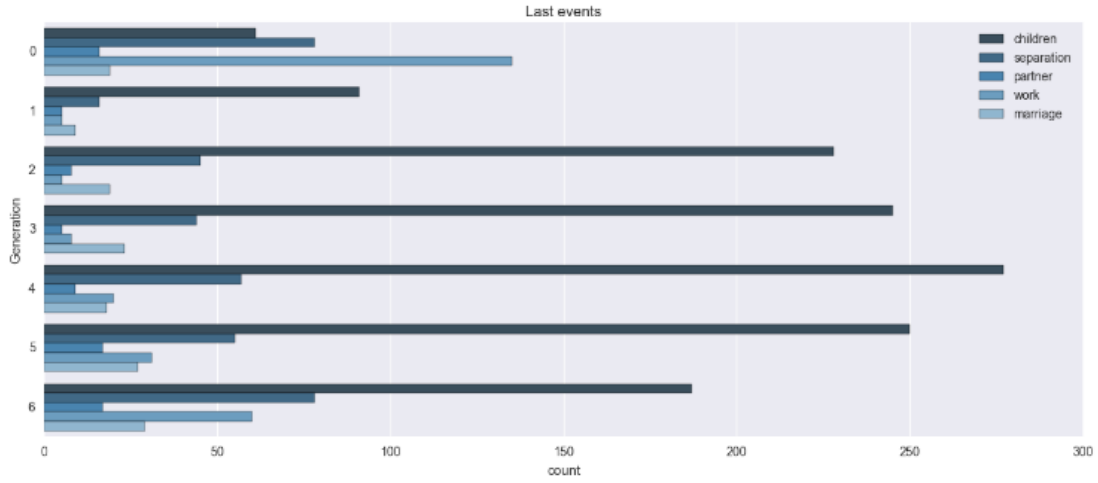
In [3]:
```
plt.figure(figsize=(15, 6))
plt.title('First events')
plt.ylabel('Generation')
sns.countplot(y=generations, hue=first_events, palette='Blues_d');
```



2) У поколения «0» самым частым последним событием является работа. У всех остальных поколений самым частым последним событием являются дети.

In [4]:
```
last_events = [x[-1] for x in data]
```

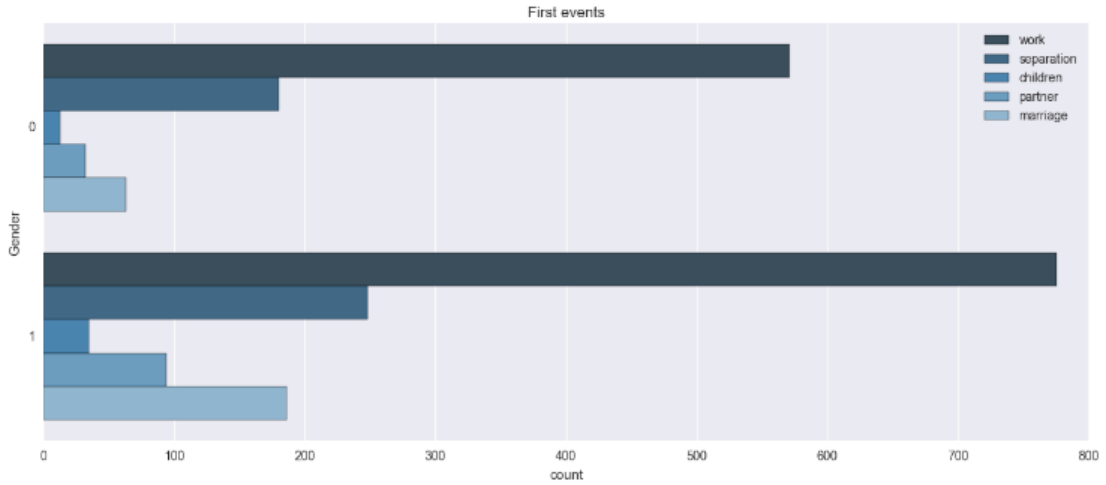
In [5]:
```
plt.figure(figsize=(15, 6))
plt.title('Last events')
plt.ylabel('Generation')
sns.countplot(y=generations, hue=last_events, palette='Blues_d');
```



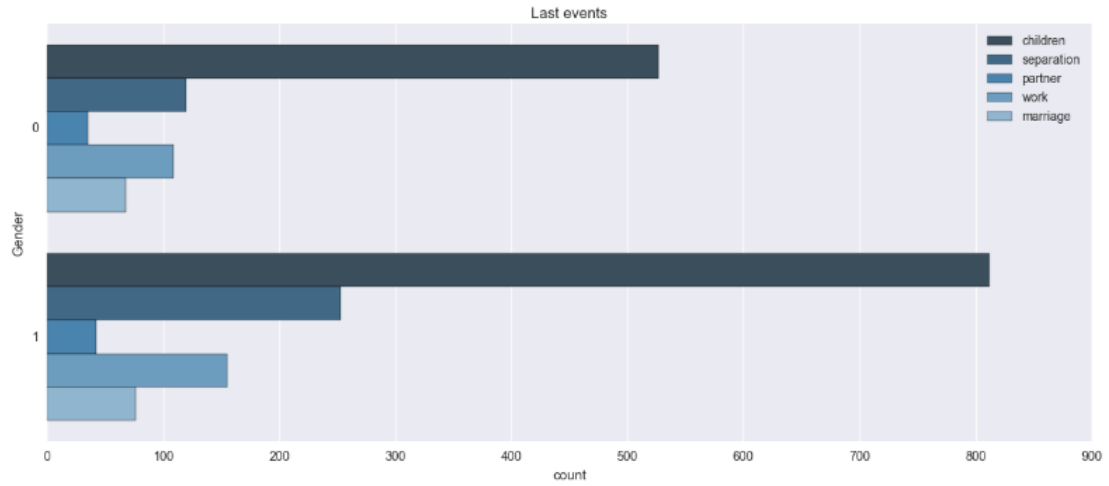
3) У обоих полов самым частым первым событием является работа, а самым частым последним — дети.

In [6]:
```
genders = zipped_data[0]
genders = list(map(lambda x: re.sub('gender=', '', x), genders))
```

In [7]:
```
plt.figure(figsize=(15, 6))
plt.title('First events')
plt.ylabel('Gender')
sns.countplot(y=genders, hue=first_events, palette='Blues_d');
```



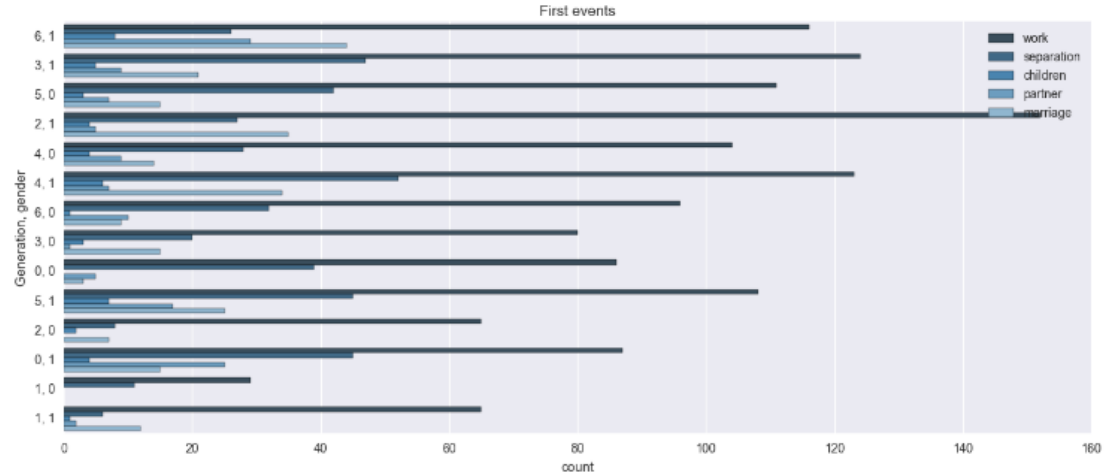
```
In [8]: plt.figure(figsize=(15, 6))
        plt.title('Last events')
        plt.ylabel('Gender')
        sns.countplot(y=genders, hue=last_events, palette='Blues_d');
```



У всех комбинаций поколений-полов самым частым первым событием является работа.

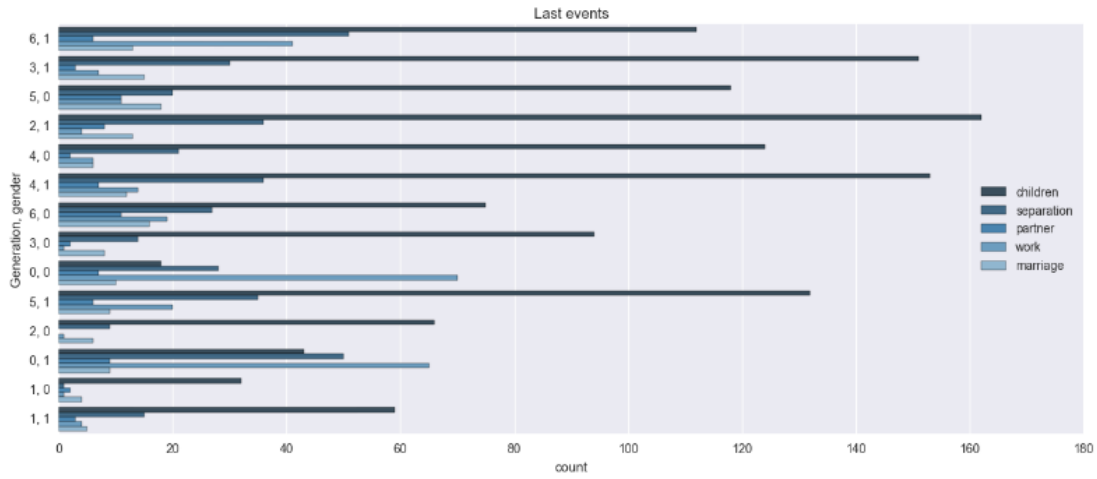
```
In [9]: generations_genders = list(zip(generations, genders))
        generations_genders = list(map(lambda x: ', '.join(x), generations_genders))
```

```
In [10]: plt.figure(figsize=(15, 6))
         plt.title('First events')
         plt.ylabel('Generation, gender')
         sns.countplot(y=generations_genders, hue=first_events, palette='Blues_d');
```



У комбинаций поколений-полов «0»-«0» и «0»-«1» самым частым последним событием является работа. У всех остальных комбинаций самым частым последним событием являются дети.

```
In [11]: plt.figure(figsize=(15, 6))
         plt.title('Last events')
         plt.ylabel('Generation, gender')
         sns.countplot(y=generations_genders, hue=last_events, palette='Blues_d');
```



2.

1) Из формулировки задания непонятно, как выражается поддержка — в долях, процентах или количествах объектов, поэтому возьмем, на наш взгляд, самое логичное — 1% (в долях это будет 100%, что даст везде нули, а в количестве объектов частотность последовательности будет означать, что она встречается хотя бы раз, и это не кажется осмысленным).

   - Частых последовательностей: 125.
   - Частых замкнутых последовательностей: 122.

2) Код для разделения данных по полам на Python:

In [12]:
```
gender0 = []
gender1 = []
with open('socAttrAndSeqFusion(full).txt') as f:
    data = f.readlines()
for x in data:
    sequence = ' '.join(x.split()[8:])
    gender = x.split('gender=')[1][0]
    if gender == '0':
        gender0.append(sequence)
    else:
        gender1.append(sequence)
```

In [13]:
```
output0 = open('gender0.txt', 'w')
output0.write('\n'.join(gender0))
output0.close()
```

In [14]:
```
output1 = open('gender1.txt', 'w')
output1.write('\n'.join(gender1))
output1.close()
```

Пол «0»:

- Наиболее частая последовательность длины 2: work, children ($support = 514$)
- Наиболее частая последовательность длины 3: work, marriage, children ($support = 362$)
- Наиболее частая последовательность длины 4: work, separation, marriage, children ($support = 95$)
- Наиболее частая последовательность длины 5: work, separation, partner, marriage, children ($support = 10$)

Пол «1»:

- Наиболее частая последовательность длины 2: marriage, children ($support = 782$)
- Наиболее частая последовательность длины 3: work, marriage, children ($support = 480$)
- Наиболее частая последовательность длины 4: work, marriage, separation, children ($support = 156$)
- Наиболее частая последовательность длины 5: work, partner, marriage, separation, children ($support = 17$)

3) Код для разделения данных по поколениям и полам на Python:

In [15]:
```
generation_gender = [[] for i in range(14)]
with open('socAttrAndSeqFusion(full).txt') as f:
    data = f.readlines()
for x in data:
    sequence = ' '.join(x.split()[8:])
    generation = int(x.split('generation=')[1][0])
    gender = int(x.split('gender=')[1][0])
    generation_gender[7 * gender + generation].append(sequence)
```

In [16]:
```
for i in range(len(generation_gender)):
    output = open('generation' + str(i % 7) + '_gender' + str(i // 7) + '.txt', 'w')
    output.write('\n'.join(generation_gender[i]))
    output.close()
```

Комбинация поколения-пола «0»-«0»:

- Частых последовательностей: 67
- Частых замкнутых последовательностей: 56
- Наиболее частая последовательность: work ($support = 118$)

Комбинация поколения-пола «0»-«1»:

- Частых последовательностей: 74
- Частых замкнутых последовательностей: 73
- Наиболее частая последовательность: work ($support = 130$)

Комбинация поколения-пола «1»-«0»:

- Частых последовательностей: 99
- Частых замкнутых последовательностей: 41
- Наиболее частая последовательность: marriage ($support = 38$)

Комбинация поколения-пола «1»-«1»:

- Частых последовательностей: 114
- Частых замкнутых последовательностей: 58

- Наиболее частая последовательность: marriage ($support = 73$)

Комбинация поколения-пола «2»-«0»:

- Частых последовательностей: 98
- Частых замкнутых последовательностей: 62
- Наиболее частая последовательность: marriage ($support = 79$)

Комбинация поколения-пола «2»-«1»:

- Частых последовательностей: 110
- Частых замкнутых последовательностей: 94
- Наиболее частая последовательность: marriage ($support = 208$)

Комбинация поколения-пола «3»-«0»:

- Частых последовательностей: 96
- Частых замкнутых последовательностей: 77
- Наиболее частая последовательность: marriage ($support = 113$)

Комбинация поколения-пола «3»-«1»:

- Частых последовательностей: 115
- Частых замкнутых последовательностей: 103
- Наиболее частая последовательность: marriage ($support = 191$)

Комбинация поколения-пола «4»-«0»:

- Частых последовательностей: 139
- Частых замкнутых последовательностей: 121
- Наиболее частые последовательности: children и marriage ($support = 152$)

Комбинация поколения-пола «4»-«1»:

- Частых последовательностей: 125
- Частых замкнутых последовательностей: 113
- Наиболее частая последовательность: marriage ($support = 206$)

Комбинация поколения-пола «5»-«0»:

- Частых последовательностей: 166
- Частых замкнутых последовательностей: 136
- Наиболее частая последовательность: work ($support = 156$)

Комбинация поколения-пола «5»-«1»:

- Частых последовательностей: 151
- Частых замкнутых последовательностей: 133
- Наиболее частая последовательность: children ($support = 189$)

Комбинация поколения-пола «6»-«0»:

- Частых последовательностей: 142
- Частых замкнутых последовательностей: 123
- Наиболее частая последовательность: work ($support = 132$)

Комбинация поколения-пола «6»-«1»:

- Частых последовательностей: 157
- Частых замкнутых последовательностей: 143
- Наиболее частая последовательность: children ($support = 182$)

4) Для поиска ассоциативных правил воспользуемся алгоритмом FPGrowth в SPMF, а для этого сначала занумеруем события (переведем «children» в «1», «marriage» в «2», «partner» в «3», «separation» в «4», «work» в «5») и отсортируем полученные данные для каждого человека. Код для предобработки данных на Python:

```
In [17]: events = ['children', 'marriage', 'partner', 'separation', 'work']

def gender_event2number(gender):
    with open('gender' + gender + '.txt') as f:
        data = f.readlines()
    data = list(map(lambda x: re.sub('-1|-2|\n', '', x).split(), data))
    data = list(map(lambda x: sorted([str(events.index(event) + 1) for event in x]), data))
    output = open('gender' + gender + '_new.txt', 'w')
    output.write('\n'.join([' '.join(x) for x in data]))
    output.close()

gender_event2number('0')
gender_event2number('1')
```

Ассоциативные правила для пола «0» (минимальная поддержка 1%, минимальная достоверность 0.5):

2 ==> 1 #SUP: 615 #CONF: 0.9138187221396731

1 ==> 2 #SUP: 615 #CONF: 0.9564541213063764

3 ==> 1 #SUP: 168 #CONF: 0.7368421052631579

4 ==> 1 #SUP: 534 #CONF: 0.8030075187969925

1 ==> 4 #SUP: 534 #CONF: 0.8304821150855366

5 ==> 1 #SUP: 559 #CONF: 0.7453333333333333

1 ==> 5 #SUP: 559 #CONF: 0.8693623639191291

3 ==> 2 #SUP: 179 #CONF: 0.7850877192982456

4 ==> 2 #SUP: 559 #CONF: 0.8406015037593985

2 ==> 4 #SUP: 559 #CONF: 0.8306092124814265

5 ==> 2 #SUP: 585 #CONF: 0.78

2 ==> 5 #SUP: 585 #CONF: 0.8692421991084696

3 ==> 4 #SUP: 192 #CONF: 0.8421052631578947

3 ==> 5 #SUP: 203 #CONF: 0.8903508771929824

5 ==> 4 #SUP: 599 #CONF: 0.7986666666666666

4 ==> 5 #SUP: 599 #CONF: 0.9007518796992481

2 3 ==> 1 #SUP: 156 #CONF: 0.8715083798882681

1 3 ==> 2 #SUP: 156 #CONF: 0.9285714285714286

3 ==> 1 2 #SUP: 156 #CONF: 0.6842105263157895

2 4 ==> 1 #SUP: 513 #CONF: 0.9177101967799642

1 4 ==> 2 #SUP: 513 #CONF: 0.9606741573033708

1 2 ==> 4 #SUP: 513 #CONF: 0.8341463414634146

4 ==> 1 2 #SUP: 513 #CONF: 0.7714285714285715

2 ==> 1 4 #SUP: 513 #CONF: 0.7622585438335809

1 ==> 2 4 #SUP: 513 #CONF: 0.7978227060653188

2 5 ==> 1 #SUP: 538 #CONF: 0.9196581196581196

1 5 ==> 2 #SUP: 538 #CONF: 0.962432915921288

1 2 ==> 5 #SUP: 538 #CONF: 0.8747967479674796

5 ==> 1 2 #SUP: 538 #CONF: 0.7173333333333334

2 ==> 1 5 #SUP: 538 #CONF: 0.799405646359584

1 ==> 2 5 #SUP: 538 #CONF: 0.8367029548989113

3 4 ==> 1 #SUP: 141 #CONF: 0.734375

1 3 ==> 4 #SUP: 141 #CONF: 0.8392857142857143

3 ==> 1 4 #SUP: 141 #CONF: 0.618421052631579

3 5 ==> 1 #SUP: 150 #CONF: 0.7389162561576355

1 3 ==> 5 #SUP: 150 #CONF: 0.8928571428571429

3 ==> 1 5 #SUP: 150 #CONF: 0.6578947368421053

4 5 ==> 1 #SUP: 488 #CONF: 0.8146911519198664

1 5 ==> 4 #SUP: 488 #CONF: 0.8729874776386404

1 4 ==> 5 #SUP: 488 #CONF: 0.9138576779026217

5 ==> 1 4 #SUP: 488 #CONF: 0.6506666666666666

4 ==> 1 5 #SUP: 488 #CONF: 0.7338345864661654

1 ==> 4 5 #SUP: 488 #CONF: 0.7589424572317263

3 4 ==> 2 #SUP: 149 #CONF: 0.7760416666666666

2 3 ==> 4 #SUP: 149 #CONF: 0.8324022346368715

3 ==> 2 4 #SUP: 149 #CONF: 0.6535087719298246

3 5 ==> 2 #SUP: 158 #CONF: 0.7783251231527094

2 3 ==> 5 #SUP: 158 #CONF: 0.88268156424581

3 ==> 2 5 #SUP: 158 #CONF: 0.6929824561403509

4 5 ==> 2 #SUP: 510 #CONF: 0.8514190317195326

2 5 ==> 4 #SUP: 510 #CONF: 0.8717948717948718

2 4 ==> 5 #SUP: 510 #CONF: 0.9123434704830053

5 ==> 2 4 #SUP: 510 #CONF: 0.68

4 ==> 2 5 #SUP: 510 #CONF: 0.7669172932330827

2 ==> 4 5 #SUP: 510 #CONF: 0.7578008915304606

3 5 ==> 4 #SUP: 175 #CONF: 0.8620689655172413

3 4 ==> 5 #SUP: 175 #CONF: 0.9114583333333334

3 ==> 4 5 #SUP: 175 #CONF: 0.7675438596491229

2 3 4 ==> 1 #SUP: 129 #CONF: 0.8657718120805369

1 3 4 ==> 2 #SUP: 129 #CONF: 0.9148936170212766

1 2 3 ==> 4 #SUP: 129 #CONF: 0.8269230769230769

3 4 ==> 1 2 #SUP: 129 #CONF: 0.671875

2 3 ==> 1 4 #SUP: 129 #CONF: 0.7206703910614525

1 3 ==> 2 4 #SUP: 129 #CONF: 0.7678571428571429

3 ==> 1 2 4 #SUP: 129 #CONF: 0.5657894736842105

2 3 5 ==> 1 #SUP: 139 #CONF: 0.879746835443038

1 3 5 ==> 2 #SUP: 139 #CONF: 0.9266666666666666

1 2 3 ==> 5 #SUP: 139 #CONF: 0.8910256410256411

3 5 ==> 1 2 #SUP: 139 #CONF: 0.6847290640394089

2 3 ==> 1 5 #SUP: 139 #CONF: 0.776536312849162

1 3 ==> 2 5 #SUP: 139 #CONF: 0.8273809523809523

3 ==> 1 2 5 #SUP: 139 #CONF: 0.6096491228070176

2 4 5 ==> 1 #SUP: 470 #CONF: 0.9215686274509803

1 4 5 ==> 2 #SUP: 470 #CONF: 0.9631147540983607

1 2 5 ==> 4 #SUP: 470 #CONF: 0.8736059479553904

1 2 4 ==> 5 #SUP: 470 #CONF: 0.9161793372319688

4 5 ==> 1 2 #SUP: 470 #CONF: 0.7846410684474123

2 5 ==> 1 4 #SUP: 470 #CONF: 0.8034188034188035

2 4 ==> 1 5 #SUP: 470 #CONF: 0.8407871198568873

1 5 ==> 2 4 #SUP: 470 #CONF: 0.8407871198568873

1 4 ==> 2 5 #SUP: 470 #CONF: 0.8801498127340824

1 2 ==> 4 5 #SUP: 470 #CONF: 0.7642276422764228

5 ==> 1 2 4 #SUP: 470 #CONF: 0.6266666666666667

4 ==> 1 2 5 #SUP: 470 #CONF: 0.706766917293233

2 ==> 1 4 5 #SUP: 470 #CONF: 0.6983655274888558

1 ==> 2 4 5 #SUP: 470 #CONF: 0.7309486780715396

3 4 5 ==> 1 #SUP: 129 #CONF: 0.7371428571428571

1 3 5 ==> 4 #SUP: 129 #CONF: 0.86

1 3 4 ==> 5 #SUP: 129 #CONF: 0.9148936170212766

3 5 ==> 1 4 #SUP: 129 #CONF: 0.6354679802955665

3 4 ==> 1 5 #SUP: 129 #CONF: 0.671875

1 3 ==> 4 5 #SUP: 129 #CONF: 0.7678571428571429

3 ==> 1 4 5 #SUP: 129 #CONF: 0.5657894736842105

3 4 5 ==> 2 #SUP: 136 #CONF: 0.7771428571428571

2 3 5 ==> 4 #SUP: 136 #CONF: 0.8607594936708861

2 3 4 ==> 5 #SUP: 136 #CONF: 0.912751677852349

3 5 ==> 2 4 #SUP: 136 #CONF: 0.6699507389162561

3 4 ==> 2 5 #SUP: 136 #CONF: 0.7083333333333334

2 3 ==> 4 5 #SUP: 136 #CONF: 0.7597765363128491

3 ==> 2 4 5 #SUP: 136 #CONF: 0.5964912280701754

2 3 4 5 ==> 1 #SUP: 118 #CONF: 0.8676470588235294

1 3 4 5 ==> 2 #SUP: 118 #CONF: 0.9147286821705426

1 2 3 5 ==> 4 #SUP: 118 #CONF: 0.8489208633093526

1 2 3 4 ==> 5 #SUP: 118 #CONF: 0.9147286821705426

3 4 5 ==> 1 2 #SUP: 118 #CONF: 0.6742857142857143

2 3 5 ==> 1 4 #SUP: 118 #CONF: 0.7468354430379747

2 3 4 ==> 1 5 #SUP: 118 #CONF: 0.7919463087248322

1 3 5 ==> 2 4 #SUP: 118 #CONF: 0.7866666666666666

1 3 4 ==> 2 5 #SUP: 118 #CONF: 0.8368794326241135

1 2 3 ==> 4 5 #SUP: 118 #CONF: 0.7564102564102564

3 5 ==> 1 2 4 #SUP: 118 #CONF: 0.5812807881773399

3 4 ==> 1 2 5 #SUP: 118 #CONF: 0.6145833333333334

2 3 ==> 1 4 5 #SUP: 118 #CONF: 0.659217877094972

1 3 ==> 2 4 5 #SUP: 118 #CONF: 0.7023809523809523

3 ==> 1 2 4 5 #SUP: 118 #CONF: 0.5175438596491229

Ассоциативные правила для пола «1» (минимальная поддержка 1%, минимальная достоверность 0.5):

2 ==> 1 #SUP: 1017 #CONF: 0.9373271889400921

1 ==> 2 #SUP: 1017 #CONF: 0.9228675136116152

3 ==> 1 #SUP: 282 #CONF: 0.8417910447761194

4 ==> 1 #SUP: 898 #CONF: 0.8560533841754051

1 ==> 4 #SUP: 898 #CONF: 0.8148820326678766

5 ==> 1 #SUP: 898 #CONF: 0.8208409506398537

1 ==> 5 #SUP: 898 #CONF: 0.8148820326678766

3 ==> 2 #SUP: 271 #CONF: 0.808955223880597

4 ==> 2 #SUP: 896 #CONF: 0.8541468064823642

2 ==> 4 #SUP: 896 #CONF: 0.8258064516129032

5 ==> 2 #SUP: 894 #CONF: 0.8171846435100548

2 ==> 5 #SUP: 894 #CONF: 0.823963133640553

3 ==> 4 #SUP: 289 #CONF: 0.8626865671641791

3 ==> 5 #SUP: 287 #CONF: 0.8567164179104477

5 ==> 4 #SUP: 896 #CONF: 0.8190127970749543

4 ==> 5 #SUP: 896 #CONF: 0.8541468064823642

2 3 ==> 1 #SUP: 249 #CONF: 0.9188191881918819

1 3 ==> 2 #SUP: 249 #CONF: 0.8829787234042553

3 ==> 1 2 #SUP: 249 #CONF: 0.7432835820895523

2 4 ==> 1 #SUP: 847 #CONF: 0.9453125

1 4 ==> 2 #SUP: 847 #CONF: 0.9432071269487751

1 2 ==> 4 #SUP: 847 #CONF: 0.8328416912487709

4 ==> 1 2 #SUP: 847 #CONF: 0.8074356530028599

2 ==> 1 4 #SUP: 847 #CONF: 0.7806451612903226

1 ==> 2 4 #SUP: 847 #CONF: 0.7686025408348457

2 5 ==> 1 #SUP: 837 #CONF: 0.9362416107382551

1 5 ==> 2 #SUP: 837 #CONF: 0.9320712694877505

1 2 ==> 5 #SUP: 837 #CONF: 0.8230088495575221

5 ==> 1 2 #SUP: 837 #CONF: 0.7650822669104205

2 ==> 1 5 #SUP: 837 #CONF: 0.7714285714285715

1 ==> 2 5 #SUP: 837 #CONF: 0.7595281306715064

3 4 ==> 1 #SUP: 244 #CONF: 0.8442906574394463

1 3 ==> 4 #SUP: 244 #CONF: 0.8652482269503546

3 ==> 1 4 #SUP: 244 #CONF: 0.7283582089552239

3 5 ==> 1 #SUP: 244 #CONF: 0.8501742160278746

1 3 ==> 5 #SUP: 244 #CONF: 0.8652482269503546

3 ==> 1 5 #SUP: 244 #CONF: 0.7283582089552239

4 5 ==> 1 #SUP: 780 #CONF: 0.8705357142857143

1 5 ==> 4 #SUP: 780 #CONF: 0.8685968819599109

1 4 ==> 5 #SUP: 780 #CONF: 0.8685968819599109

5 ==> 1 4 #SUP: 780 #CONF: 0.7129798903107861

4 ==> 1 5 #SUP: 780 #CONF: 0.7435653002859867

1 ==> 4 5 #SUP: 780 #CONF: 0.7078039927404719

3 4 ==> 2 #SUP: 232 #CONF: 0.8027681660899654

2 3 ==> 4 #SUP: 232 #CONF: 0.8560885608856088

3 ==> 2 4 #SUP: 232 #CONF: 0.6925373134328359

3 5 ==> 2 #SUP: 231 #CONF: 0.8048780487804879

2 3 ==> 5 #SUP: 231 #CONF: 0.8523985239852399

3 ==> 2 5 #SUP: 231 #CONF: 0.6895522388059702

4 5 ==> 2 #SUP: 781 #CONF: 0.8716517857142857

2 5 ==> 4 #SUP: 781 #CONF: 0.8736017897091722

2 4 ==> 5 #SUP: 781 #CONF: 0.8716517857142857

5 ==> 2 4 #SUP: 781 #CONF: 0.7138939670932358

4 ==> 2 5 #SUP: 781 #CONF: 0.7445185891325071

2 ==> 4 5 #SUP: 781 #CONF: 0.719815668202765

3 5 ==> 4 #SUP: 253 #CONF: 0.8815331010452961

3 4 ==> 5 #SUP: 253 #CONF: 0.8754325259515571

3 ==> 4 5 #SUP: 253 #CONF: 0.755223880597015

2 3 4 ==> 1 #SUP: 215 #CONF: 0.9267241379310345

1 3 4 ==> 2 #SUP: 215 #CONF: 0.8811475409836066

1 2 3 ==> 4 #SUP: 215 #CONF: 0.8634538152610441

3 4 ==> 1 2 #SUP: 215 #CONF: 0.7439446366782007

2 3 ==> 1 4 #SUP: 215 #CONF: 0.7933579335793358

1 3 ==> 2 4 #SUP: 215 #CONF: 0.7624113475177305

3 ==> 1 2 4 #SUP: 215 #CONF: 0.6417910447761194

2 3 5 ==> 1 #SUP: 214 #CONF: 0.9264069264069265

1 3 5 ==> 2 #SUP: 214 #CONF: 0.8770491803278688

1 2 3 ==> 5 #SUP: 214 #CONF: 0.8594377510040161

3 5 ==> 1 2 #SUP: 214 #CONF: 0.7456445993031359

2 3 ==> 1 5 #SUP: 214 #CONF: 0.7896678966789668

1 3 ==> 2 5 #SUP: 214 #CONF: 0.7588652482269503

3 ==> 1 2 5 #SUP: 214 #CONF: 0.6388059701492538

2 4 5 ==> 1 #SUP: 738 #CONF: 0.9449423815620999

1 4 5 ==> 2 #SUP: 738 #CONF: 0.9461538461538461

1 2 5 ==> 4 #SUP: 738 #CONF: 0.8817204301075269

1 2 4 ==> 5 #SUP: 738 #CONF: 0.8713105076741441

4 5 ==> 1 2 #SUP: 738 #CONF: 0.8236607142857143

2 5 ==> 1 4 #SUP: 738 #CONF: 0.825503355704698

2 4 ==> 1 5 #SUP: 738 #CONF: 0.8236607142857143

1 5 ==> 2 4 #SUP: 738 #CONF: 0.821826280623608

1 4 ==> 2 5 #SUP: 738 #CONF: 0.821826280623608

1 2 ==> 4 5 #SUP: 738 #CONF: 0.7256637168141593

5 ==> 1 2 4 #SUP: 738 #CONF: 0.6745886654478976

4 ==> 1 2 5 #SUP: 738 #CONF: 0.7035271687321258

2 ==> 1 4 5 #SUP: 738 #CONF: 0.680184331797235

1 ==> 2 4 5 #SUP: 738 #CONF: 0.6696914700544465

3 4 5 ==> 1 #SUP: 217 #CONF: 0.857707509881423

1 3 5 ==> 4 #SUP: 217 #CONF: 0.889344262295082

1 3 4 ==> 5 #SUP: 217 #CONF: 0.889344262295082

3 5 ==> 1 4 #SUP: 217 #CONF: 0.7560975609756098

3 4 ==> 1 5 #SUP: 217 #CONF: 0.7508650519031141

1 3 ==> 4 5 #SUP: 217 #CONF: 0.7695035460992907

3 ==> 1 4 5 #SUP: 217 #CONF: 0.6477611940298508

3 4 5 ==> 2 #SUP: 204 #CONF: 0.8063241106719368

2 3 5 ==> 4 #SUP: 204 #CONF: 0.8831168831168831

2 3 4 ==> 5 #SUP: 204 #CONF: 0.8793103448275862

3 5 ==> 2 4 #SUP: 204 #CONF: 0.710801393728223

3 4 ==> 2 5 #SUP: 204 #CONF: 0.7058823529411765

2 3 ==> 4 5 #SUP: 204 #CONF: 0.7527675276752768

3 ==> 2 4 5 #SUP: 204 #CONF: 0.608955223880597

2 3 4 5 ==> 1 #SUP: 191 #CONF: 0.9362745098039216

1 3 4 5 ==> 2 #SUP: 191 #CONF: 0.880184331797235

1 2 3 5 ==> 4 #SUP: 191 #CONF: 0.8925233644859814

1 2 3 4 ==> 5 #SUP: 191 #CONF: 0.8883720930232558

3 4 5 ==> 1 2 #SUP: 191 #CONF: 0.7549407114624506

2 3 5 ==> 1 4 #SUP: 191 #CONF: 0.8268398268398268

2 3 4 ==> 1 5 #SUP: 191 #CONF: 0.8232758620689655

1 3 5 ==> 2 4 #SUP: 191 #CONF: 0.7827868852459017

1 3 4 ==> 2 5 #SUP: 191 #CONF: 0.7827868852459017

1 2 3 ==> 4 5 #SUP: 191 #CONF: 0.7670682730923695

3 5 ==> 1 2 4 #SUP: 191 #CONF: 0.6655052264808362

3 4 ==> 1 2 5 #SUP: 191 #CONF: 0.6608996539792388

2 3 ==> 1 4 5 #SUP: 191 #CONF: 0.7047970479704797

1 3 ==> 2 4 5 #SUP: 191 #CONF: 0.6773049645390071

3 ==> 1 2 4 5 #SUP: 191 #CONF: 0.5701492537313433

Три самых достоверных правила для пола «0»:

1 4 5 ==> 2 #SUP: 470 #CONF: 0.9631147540983607

1 5 ==> 2 #SUP: 538 #CONF: 0.962432915921288

1 4 ==> 2 #SUP: 513 #CONF: 0.9606741573033708

Три самых достоверных правила для пола «1»:

1 4 5 ==> 2 #SUP: 738 #CONF: 0.9461538461538461

2 4 ==> 1 #SUP: 847 #CONF: 0.9453125

2 4 5 ==> 1 #SUP: 738 #CONF: 0.9449423815620999

3.

1) Поиск частых множеств и ассоциативных правил можно применить при стимулировании продаж, например, книг. Объектами будут покупатели, а признаками — книги из каталога сайта. Соответственно, по найденным частым множествам и ассоциативным правилам будет понятнее, какие книги берут вместе и что рекомендовать покупателям.

2) Поиск частых (под)последовательностей можно применить при анализе и предупреждении самых разных заболеваний. Объектами будут пациенты, а событиями — их заболевания. Выявление частых (под)последовательностей в подобной базе данных может указать на влияние одних заболеваний на другие, что повлечет за собой соответствующие исследования в данном направлении и предупреждение, например, рака или инсульта.

3) Поиск частых (под)графов можно применить при анализе распространения эпидемии. Вершинами будут переносчики эпидемии с некоторыми признаками, а ребрами будут обозначены контакты. Выявление частых (под)графов позволит заметить закономерности при передаче инфекции, сконцентрироваться на ее конкретных шаблонах и, возможно, продвинуться в исследованиях, повернув их в более правильном направлении.