马的疝病分析

一 数据准备

首先将原始的 txt 文件转换成 csv 文件，以便于对数据进行分析：

```
fp_origin = open("horse-colic.txt", 'r')
fp_modified = open("horse-colic.csv", 'w')

line = fp_origin.readline()
while(line):
    temp = line.strip().split()
    temp = ','.join(temp)+'\n'
    fp_modified.write(temp)
    line = fp_origin.readline()

fp_origin.close()
fp_modified.close()
```

然后读取数据，并且将属性分为标称属性和数值属性：

```
# 属性
attribute = ["surgery","Age","Hospital Number","rectal temperature"," pulse",
"respiratory rate","temperature of extremities","peripheral pulse","mucous membranes",
"capillary refill time","pain","peristalsis","abdominal distension","nasogastric tube",
"nasogastric reflux","nasogastric reflux PH","rectal examination","abdomen",
"packed cell volume","total protein","abdominocentesis appearance",
"abdomcentesis total protein","outcome","surgical lesion","t1","t2","t3","cp_data"]
# 数值属性
name_value = ["rectal temperature"," pulse","respiratory rate","nasogastric reflux PH",
"packed cell volume","total protein","abdomcentesis total protein"]
# 标称属性
name_category = ["surgery","Age","Hospital Number","temperature of extremities",
"peripheral pulse","mucous membranes","capillary refill time","pain","peristalsis",
"abdominal distension","nasogastric tube","nasogastric reflux","rectal examination",
"abdomen","abdominocentesis appearance","outcome","surgical lesion","cp_data"]
# 读取数据
data_origin = pd.read_csv("horse-colic.csv",names = attribute,na_values = "?")
# 将字符数据转换为category
for item in name_category:
    data_origin[item] = data_origin[item].astype('category')
```

二 数据摘要和可视化

2.1 数据摘要

2.1.1 标称属性

对标称属性，给出每个可能取值的频数。使用 pandas 中的 value_counts 函数统计每个标称属性的取值频数：

```
for item in name_category:
    print (item, '的频数为: \n', pd.value_counts(data_origin[item].values), '\n')
```

统计结果为：

Surgery 的频数为：

1.0	214
2.0	152

Age 的频数为：

1	340
9	28

temperature of extremities 的频数为：

3.0	135
1.0	95
2.0	39
4.0	34

peripheral pulse 的频数为：

1.0	151
3.0	116
4.0	12
2.0	6

mucous membranes 的频数为：

1.0	98
3.0	81
4.0	50
2.0	38
5.0	28
6.0	25

capillary refill time 的频数为：

1.0	232
2.0	96
3.0	2

pain 的频数为：

3.0	82
2.0	77
5.0	50
1.0	49
4.0	47

peristalsis 的频数为：
3.0 154
4.0 91
1.0 49
2.0 22

abdominal distension 的频数为：
1.0 101
3.0 85
2.0 75
4.0 42

nasogastric tube 的频数为：
2.0 121
1.0 89
3.0 27

nasogastric reflux 的频数为：
1.0 141
3.0 49
2.0 45

rectal examination 的频数为：
4.0 97
1.0 68
3.0 61
2.0 14

abdomen 的频数为：
5.0 96
4.0 55
1.0 31
2.0 24
3.0 19

abdominocentesis appearance 的频数为：
2.0 62
3.0 60
1.0 52

outcome 的频数为：
1.0 225
2.0 89
3.0 52

surgical lesion 的频数为：
1 232
2 136

cp_data 的频数为：
2 244
1 124

2.1.2 数值属性

对于数值属性，给出最大、最小、均值、中位数、四分位数及缺失值的个数：

```
# 最大值
data_show = pd.DataFrame(data = data_origin[name_value].max(), columns = ['max'])
# 最小值
data_show['min'] = data_origin[name_value].min()
# 均值
data_show['mean'] = data_origin[name_value].mean()
# 中位数
data_show['median'] = data_origin[name_value].median()
# 四分位数
data_show['quartile'] = data_origin[name_value].describe().loc['25%']
# 缺失值个数
data_show['missing'] = data_origin[name_value].describe().loc['count'].apply(lambda x : 368-x)
```

结果如下：

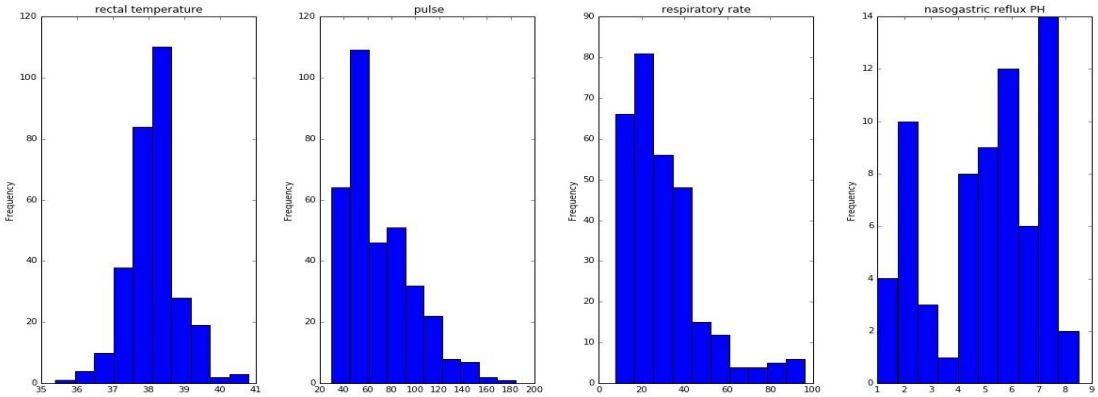
	max	min	mean	median	quartile	missing
rectal temperature	40.8	35.4	38.134448	38.1	NaN	69.0
pulse	184.0	30.0	70.757310	60.0	NaN	26.0
respiratory rate	96.0	8.0	30.521886	28.0	NaN	71.0
nasogastric reflux PH	8.5	1.0	4.962319	5.4	NaN	299.0
packed cell volume	75.0	4.0	45.656798	44.0	NaN	37.0
total protein	89.0	3.3	24.771077	7.5	NaN	43.0
abdomcentesis total protein	10.1	0.1	2.948120	2.1	NaN	235.0

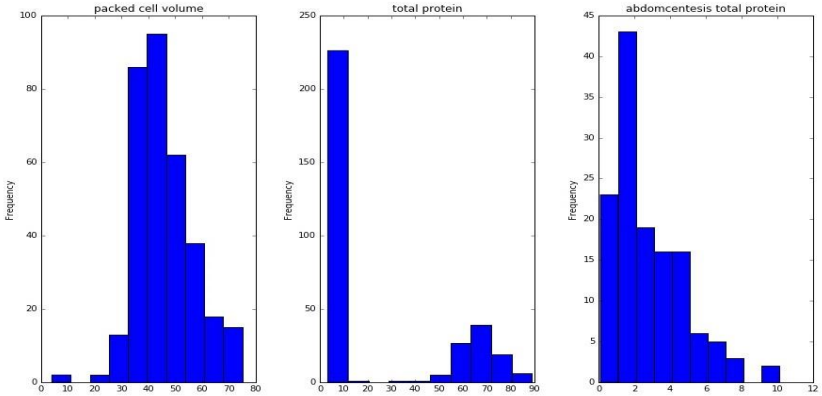
2.2 数据可视化

2.2.1 直方图和 qq 图

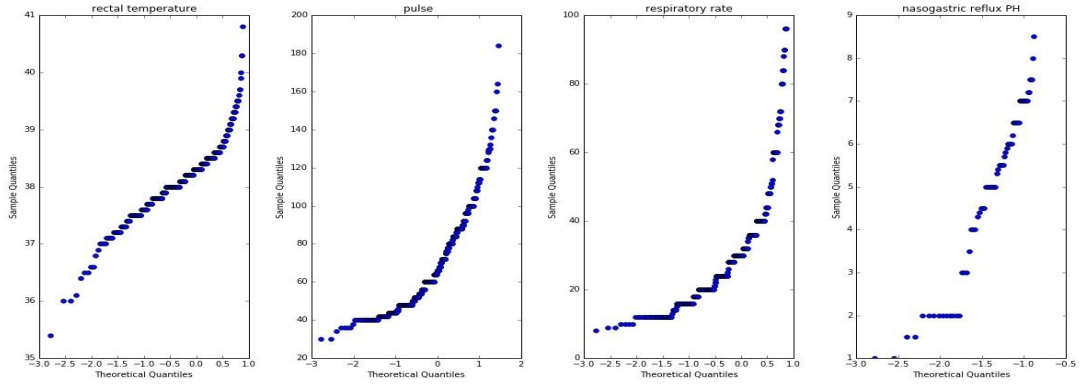
绘制直方图：

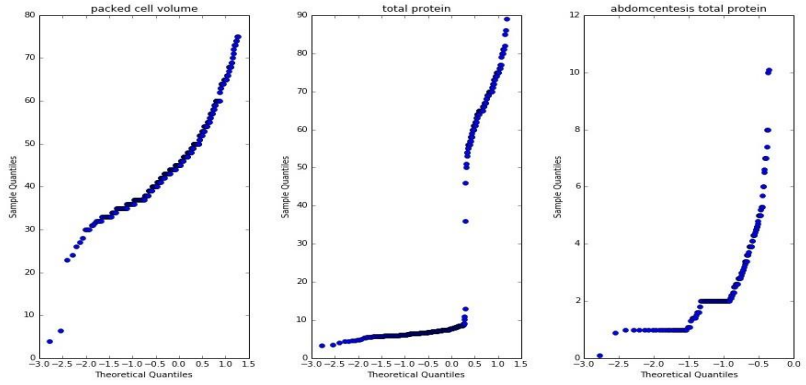
```
fig = plt.figure(figsize = (20,20))
i = 1
for item in name_value:
    ax = fig.add_subplot(2, 4, i)
    data_origin[item].plot(kind = 'hist', title = item, ax = ax)
    i += 1
plt.subplots_adjust(wspace = 0.3, hspace = 0.3)
fig.savefig('histogram.jpg')
```





绘制 qq 图，并且根据 qq 图判断是否为正态分布：



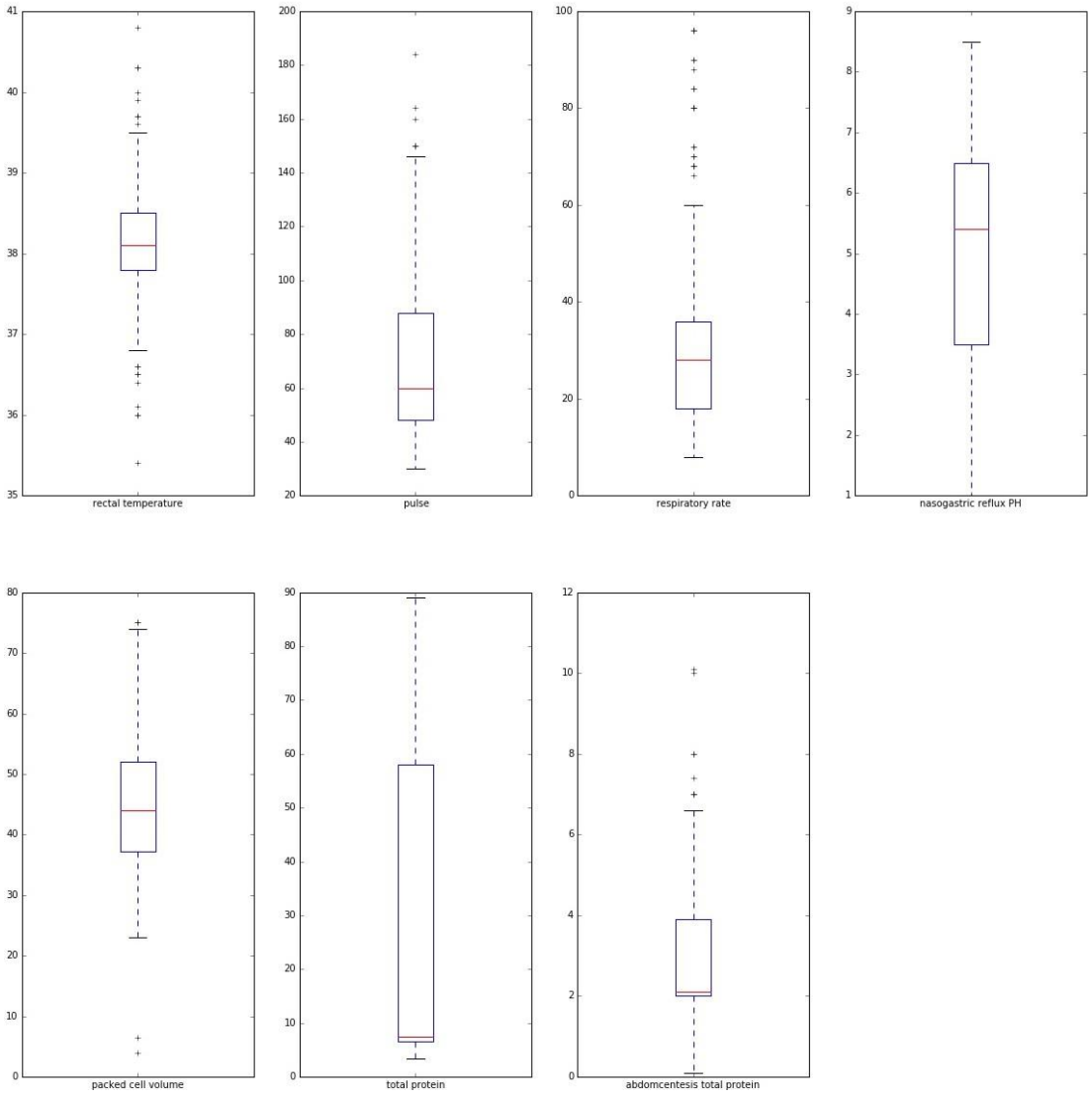


通过 qq 图可以看到 rectal temperature 属性和 packed volume 属性满足正态分布。

2.2.2 盒图

绘制盒图，对离群值进行识别：

```
fig = plt.figure(figsize = (20,20))
i = 1
for item in name_value:
    ax = fig.add_subplot(2, 4, i)
    data_origin[item].plot(kind = 'box')
    i += 1
fig.savefig('boxplot.jpg')
```



三 处理缺失数据

3.1 将缺失部分剔除

使用 dropna 函数，将有缺失值的数据整条删除，得到新的数据集：

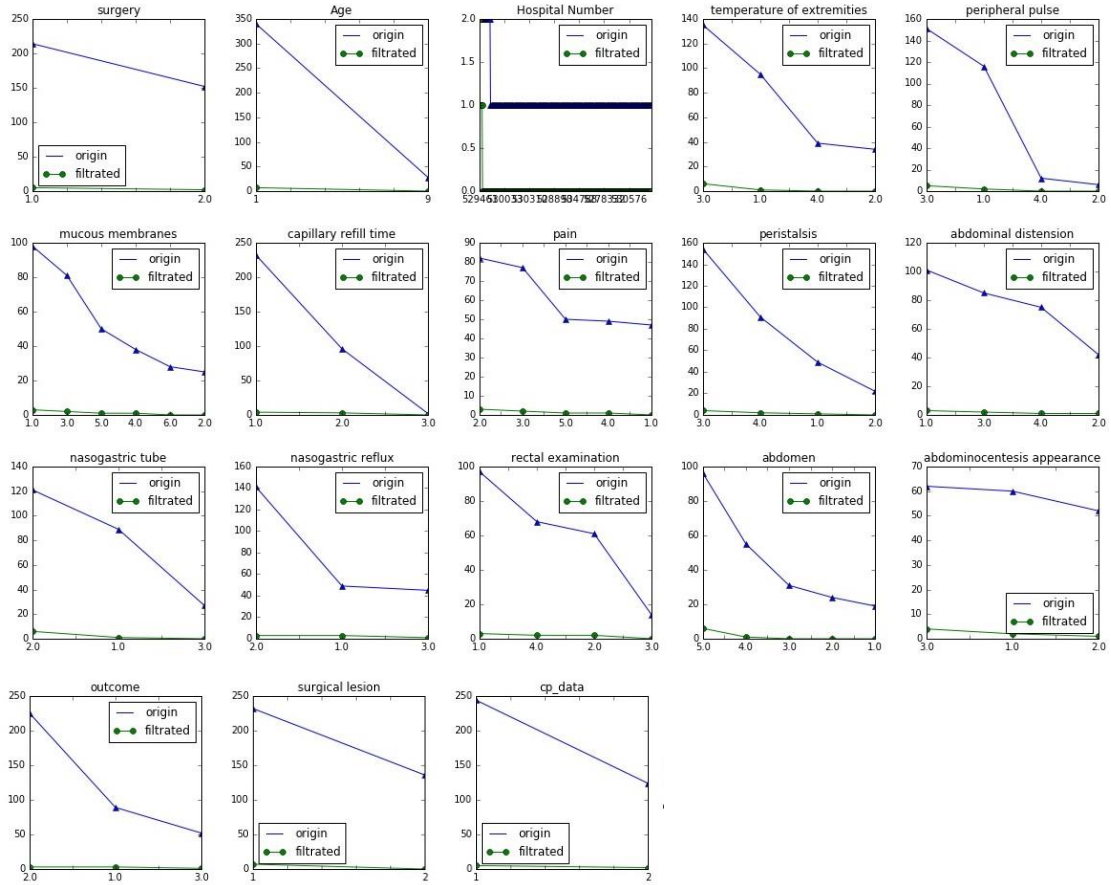
```
data_filtrated = data_origin.dropna()
```

然后可视化的对新旧数据集进行比较。

通过折线图比较新旧数据集的标称属性：

```
# 对标称属性，绘制折线图
for item in name_category:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    pd.value_counts(data_origin[item].values).plot(ax = ax, marker = '^', label = 'origin', legend = True)
    pd.value_counts(data_filtrated[item].values).plot(ax = ax, marker = 'o', label = 'filtrated', legend = True)
    i += 1
```

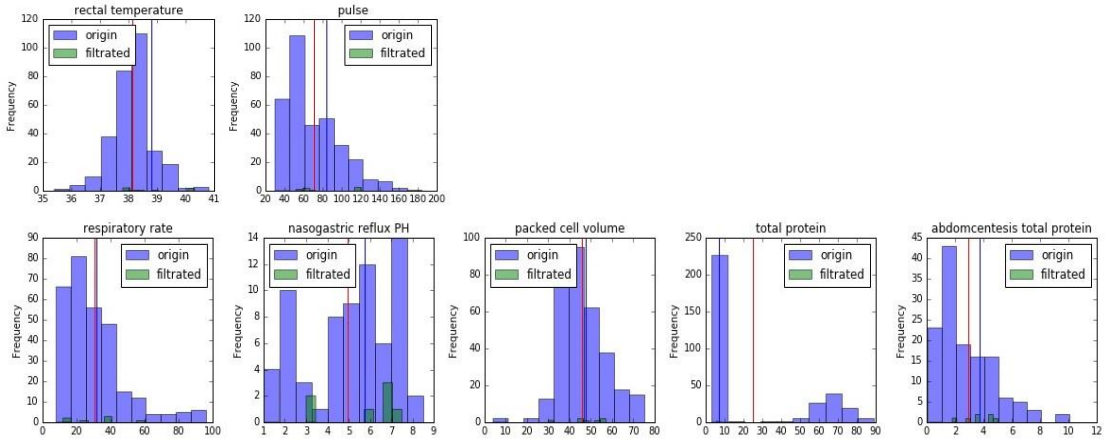
通过折线图可以看到，新的数据集数据量明显减少：



通过直方图图比较新旧数据集的标称属性：

```
# 对数值属性，绘制直方图
for item in name_value:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'filtrated', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
```

在直方图中，红色的垂线是旧数据集的均值，蓝色的垂线是新数据集的均值，可以看到，虽然新旧数据集在样本数目上差别很大，但是除了 total protein 属性之外，其余属性的数据分布非常相似，而且均值变化不大



3.2 用最高频率值来填补缺失值

分别对每个属性列进行处理，首先找到每个属性中出现次数最多的值，再用这个值填充这个属性中所有的缺失值：

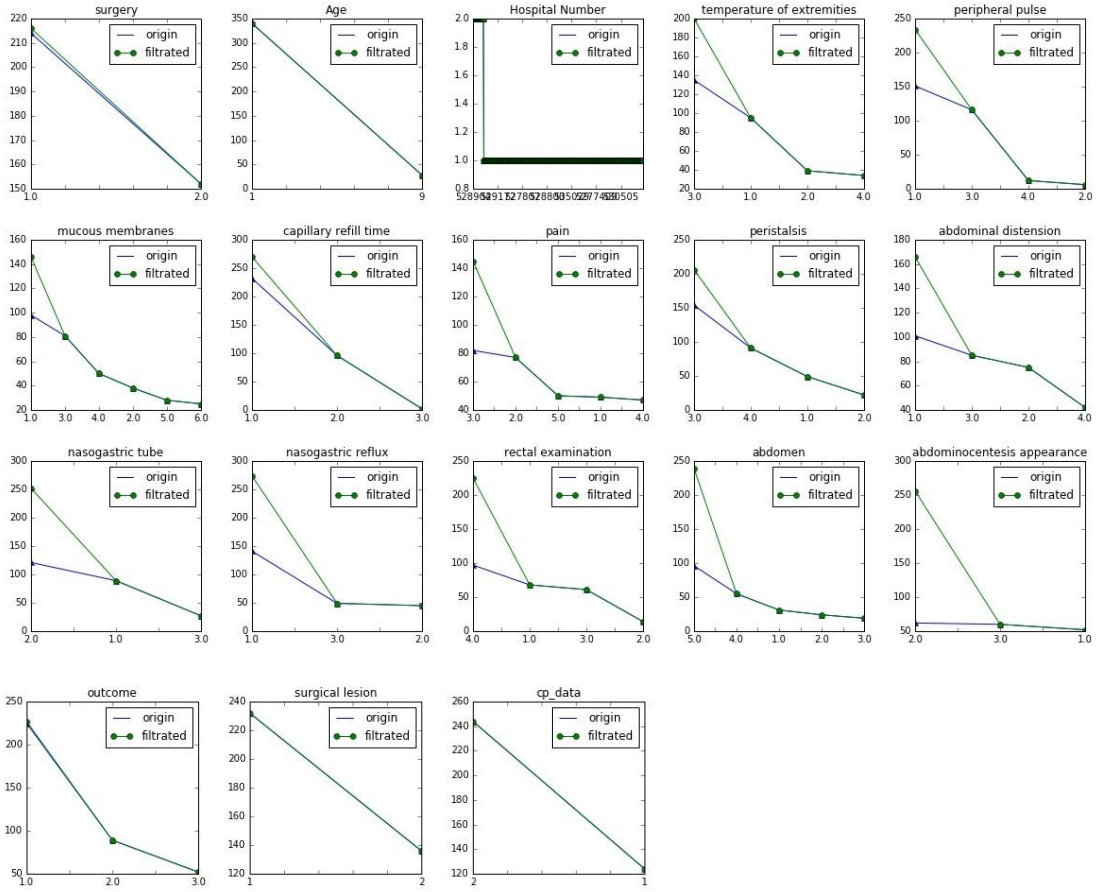
```
# 对每一列数据，分别进行处理
for item in attribute:
    # 计算最高频率的值
    most_frequent_value = data_filtrated[item].value_counts().idxmax()
    # 替换缺失值
    data_filtrated[item].fillna(value = most_frequent_value, inplace = True)
```

然后可视化的对新旧数据集进行比较。

通过折线图比较新旧数据集的标称属性：

```
# 对标称属性，绘制折线图
for item in name_category:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    pd.value_counts(data_origin[item].values).plot(ax = ax, marker = '^', label = 'origin', legend = True)
    pd.value_counts(data_filtrated[item].values).plot(ax = ax, marker = 'o', label = 'filtrated', legend = True)
    i += 1
```

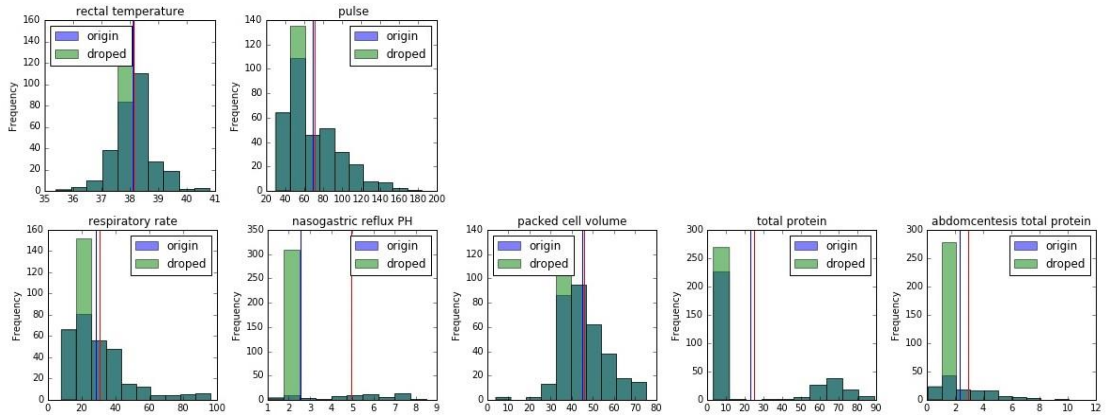
通过折线图可以看到，只有 surgery、Age、Hospital Number、outcome、surgical lesion、cp_data 这几个属性在新旧数据集上大概相同，其余的新数据上的数量都明显大于旧数据集，说明这些属性缺失比较严重：



通过直方图图比较新旧数据集的标称属性：

```
# 对数值属性，绘制直方图
for item in name_value:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
```

从直方图中可以看出，补全的都是出现最高频率的值，并且除了属性 nasogastric reflux PH，其余的属性的均值都几乎保持不变：



3.3 通过属性的相关关系来填补缺失值

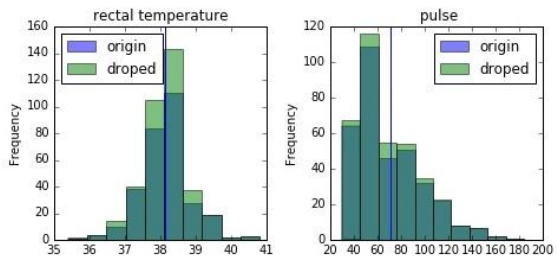
使用 pandas 中的 interpolate()函数，分别对每个数值属性进行插值计算，并用得到的插值填充缺失值：

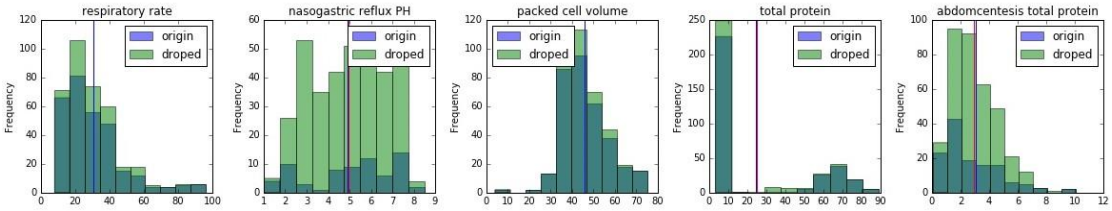
```
for item in name_value:
    data_filtrated[item].interpolate(inplace = True)
```

因为只填充了数值属性的缺失值，故我们只通过数值属性的直方图来可视化比较新旧数据集：

```
# 对数值属性，绘制直方图
for item in name_value:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
```

从直方图中可以看出，对于每个数值属性，新的数据集几乎都添加了多个值，数据分布和旧数据集几乎相同，且新旧数据集的均值几乎相等：





3.3 通过数据对象之间的相似性来填补缺失值

计算每两条数据之间的相似度，对每条数据填充缺失值时，选择与其最相似的一条数据的同一属性的值进行填充：

```
# 构造分数表
score = {}
range_length = len(data_origin)
for i in range(0, range_length):
    score[i] = {}
    for j in range(0, range_length):
        score[i][j] = 0
test = 0;
# 在处理后的数据中，对每两条数据条目计算差异性得分，分值越高差异性越大
for i in range(0, range_length):
    for j in range(i, range_length):
        for item in name_category:
            if data_norm.iloc[i][item] != data_norm.iloc[j][item]:
                score[i][j] += 1
        for item in name_value:
            temp = abs(data_norm.iloc[i][item] - data_norm.iloc[j][item])
            score[i][j] += temp
        score[j][i] = score[i][j]
    test = i
```

```
# 对有缺失值的条目，用和它相似度最高（得分最低）的数据条目中对应属性的值替换
for index in nan_list:
    best_friend = sorted(score[index].items(), key=operator.itemgetter(1), reverse = False)[1][0]
    for item in name_value:
        if pd.isnull(data_filtrated.iloc[index][item]):
            if pd.isnull(data_origin.iloc[best_friend][item]):
                data_filtrated.ix[index, item] = data_origin[item].value_counts().idxmax()
            else:
                data_filtrated.ix[index, item] = data_origin.iloc[best_friend][item]
```

通过数值属性的直方图来可视化比较新旧数据集：

```
# 对数值属性，绘制直方图
for item in name_value:
    ax = fig.add_subplot(5, 5, i)
    ax.set_title(item)
    data_origin[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'origin', legend = True)
    data_filtrated[item].plot(ax = ax, alpha = 0.5, kind = 'hist', label = 'droped', legend = True)
    ax.axvline(data_origin[item].mean(), color = 'r')
    ax.axvline(data_filtrated[item].mean(), color = 'b')
    i += 1
```

通过直方图可以看出，每个数值属性填充的缺失值大部分都集中在出现频率较高的值上，除了属性 nasogastric reflux PH，新旧数据集的均值几乎相同：