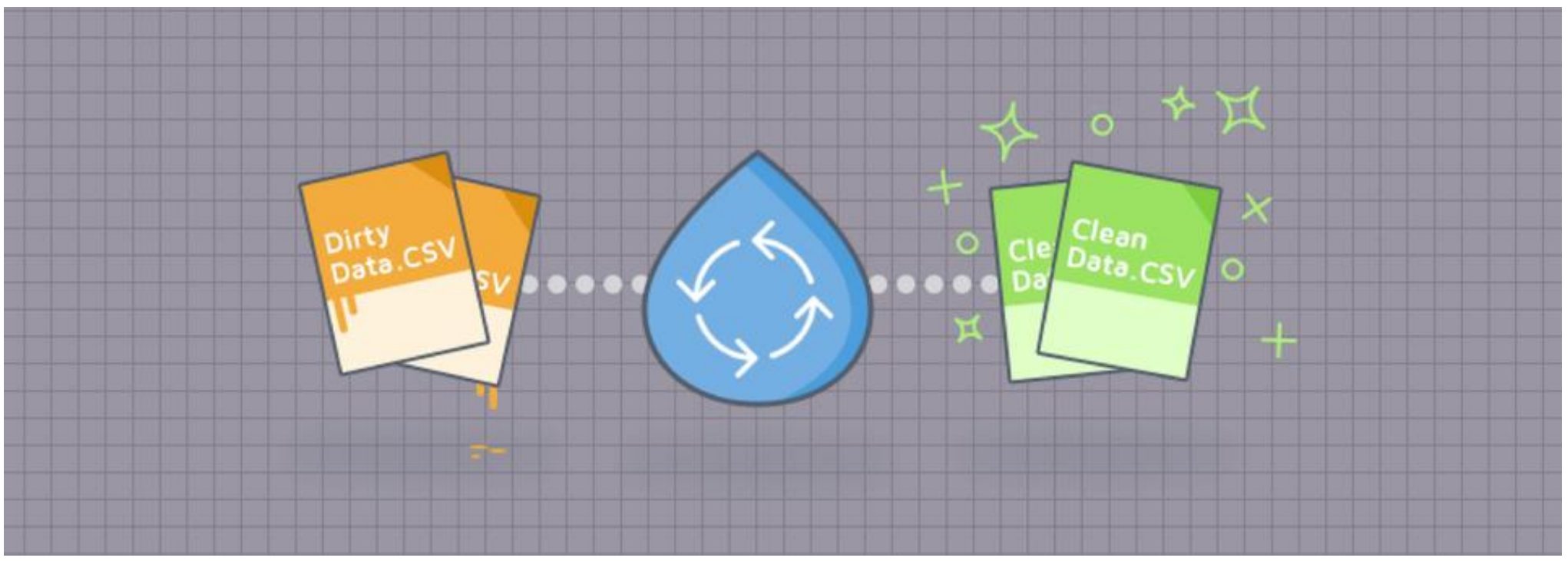
# 数据预处理基本方法及实践

作者：李沛尧

日期：2019-10-20




## 学习目标

1. 理解数据预处理的重要性
2. 掌握数据预处理的基本步骤及方法
3. 了解如何在Python中对数据进行流程化预处理

## 1. 为什么我们需要进行数据预处理?

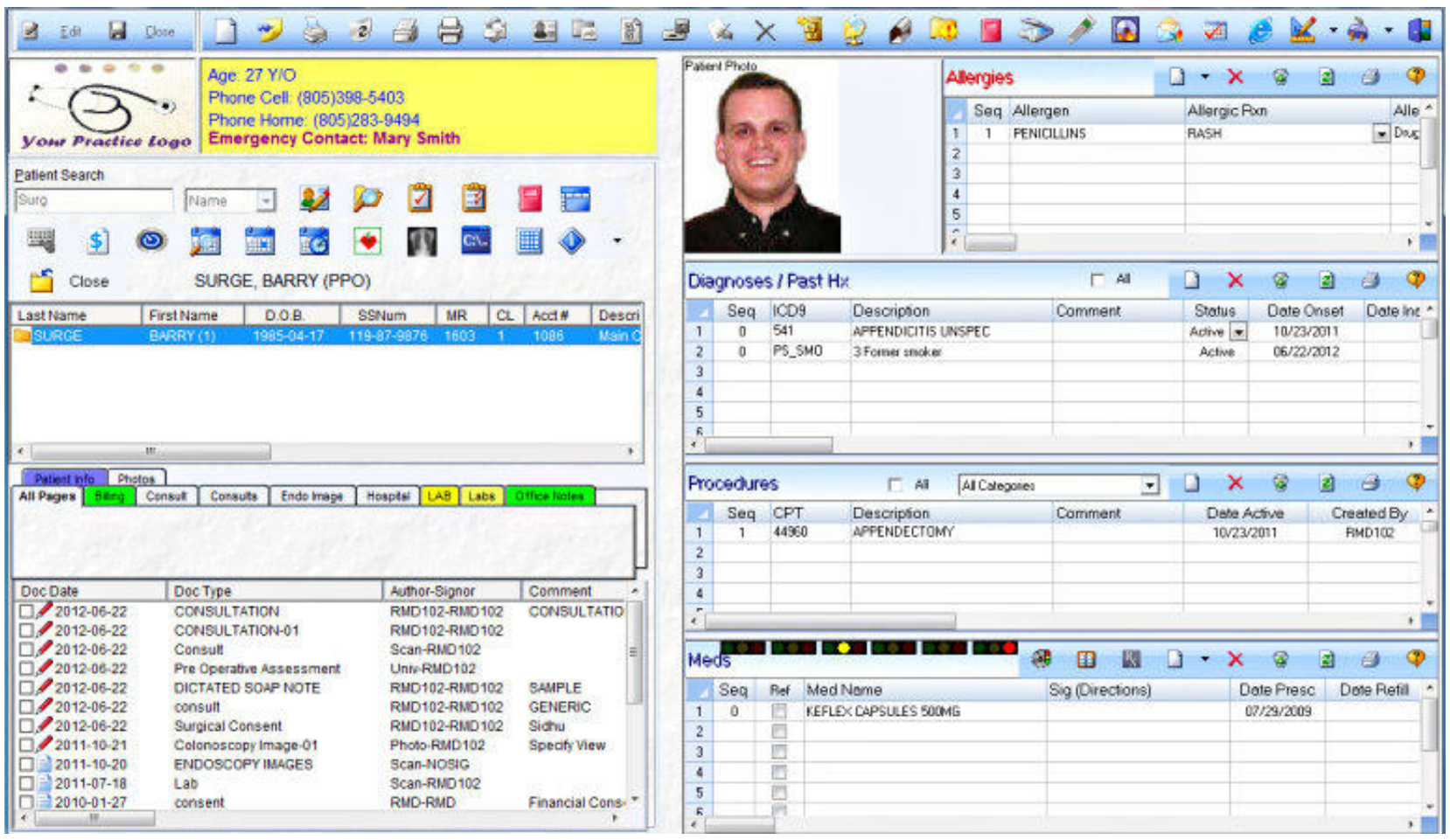
### 基本定义

数据预处理是指

> 在进行统计分析之前，把通过“**数据提取**”导出的原始数据转化为“**干净**”和“**整洁**”的数据集。

### 实践意义

使用EHR进行的研究通常涉及健康记录的二次分析，这些健康记录通常是为了临床和计费（非科学研究）目的而收集，并通过自动化过程存储，因此这些数据库会存在许多质量控制方面的问题。




现代医疗大数据信息架构


我们对数据进行预处理的目的在于 **评估并改善数据的质量，为可靠的大数据分析打下基础。**

## 2. 数据预处理的基本流程

数据预处理大致包含以下4个基本流程：

1. 数据整合
2. 数据清洗
3. 数据转化
4. 数据简化

接下来我们将会结合具体代码和数据对4个步骤进行详细解释。

### 数据集介绍

本教程采用的数据包括结构化的电子病历记录以及时序数据类型，Python中我们使用`pandas`可以对CSV格式的数据进行快速方便的处理

In [1]:

```
# 导入Pandas
import pandas as pd
```

```
/opt/conda/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed, may indicate binar
  return f(*args, **kwds)
/opt/conda/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed, may indicate binar
  return f(*args, **kwds)
```

#### 结构化数据

我们使用的结构化数据来自一个基于MIMIC(V2.6)开展的回顾性研究，具体的数据说明请参照相关文档。

In [8]:

```
# 导入结构化数据
iac_df = pd.read_csv('/home/kesci/input/data6364/full_cohort_data.csv')
# 预览数据头十行
iac_df.head(10)
```

Out[8]:

| | aline_flg | icu_los_day | hospital_los_day | age | gender_num | weight_first | bmi | sapsi_first | sofa_first | service_unit | ... | platelet_first |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 7.63 | 13 | 72.36841 | 1.0 | 75.0 | 29.912791 | 15.0 | 9.0 | SICU | ... | 354.0 |
| **1** | 0 | 1.14 | 1 | 64.92076 | 0.0 | 55.0 | 20.121312 | NaN | 5.0 | MICU | ... | NaN |
| **2** | 0 | 2.86 | 5 | 36.50000 | 0.0 | 70.0 | 27.118272 | 16.0 | 5.0 | MICU | ... | 295.0 |
| **3** | 1 | 0.58 | 3 | 44.49191 | 0.0 | NaN | NaN | 21.0 | 7.0 | SICU | ... | 262.0 |
| **4** | 1 | 1.75 | 5 | 23.74217 | 1.0 | 95.2 | 28.464563 | 18.0 | 7.0 | SICU | ... | 22.0 |
| **5** | 0 | 1.38 | 9 | 36.54657 | 1.0 | 72.0 | 23.982402 | 14.0 | 5.0 | SICU | ... | 182.0 |
| **6** | 1 | 7.06 | 27 | 24.64717 | 1.0 | 90.0 | 25.474850 | 15.0 | 6.0 | SICU | ... | 130.0 |
| **7** | 1 | 15.30 | 33 | 78.15970 | 0.0 | 69.7 | 27.219757 | 16.0 | 8.0 | MICU | ... | 20.0 |
| **8** | 1 | 3.79 | 4 | 71.43198 | 0.0 | 52.6 | 21.910820 | 9.0 | 5.0 | MICU | ... | 238.0 |
| **9** | 1 | 7.14 | 7 | 25.41667 | 0.0 | 61.5 | 21.543184 | 13.0 | 9.0 | MICU | ... | 137.0 |

10 rows × 46 columns

In [61]:

```
iac_df.columns
```

Out[61]:

```
Index(['aline_flg', 'icu_los_day', 'hospital_los_day', 'age', 'gender_num',
       'weight_first', 'bmi', 'sapsi_first', 'sofa_first', 'service_unit',
       'service_num', 'day_icu_intime', 'day_icu_intime_num',
       'hour_icu_intime', 'hosp_exp_flg', 'icu_exp_flg', 'day_28_flg',
       'mort_day_censored', 'censor_flg', 'sepsis_flg', 'chf_flg', 'afib_flg',
       'renal_flg', 'liver_flg', 'copd_flg', 'cad_flg', 'stroke_flg',
       'mal_flg', 'resp_flg', 'map_1st', 'hr_1st', 'temp_1st', 'spo2_1st',
       'abg_count', 'wbc_first', 'hgb_first', 'platelet_first', 'sodium_first',
       'potassium_first', 'tco2_first', 'chloride_first', 'bun_first',
       'creatinine_first', 'po2_first', 'pco2_first', 'iv_day_1'],
      dtype='object')
```

In [9]:

```
# 查看数据的详细情况
#import pandas_profiling

#pandas_profiling.ProfileReport(iac_df)
```

***时序数据***

我们使用的时序数据来自Physionet 2019挑战赛，具体的细节请参照官网文档。

In [10]:

```
sepsis_df = pd.read_csv('/home/kesci/input/PhysioNet20191169/p000401.csv', index_col=0)
sepsis_df.head(10)
```

Out[10]:

| | HR | O2Sat | Temp | SBP | MAP | DBP | Resp | EtCO2 | BaseExcess | HCO3 | ... | WBC | Fibrinogen | Platelets | Age | Gender | Unit1 | Unit2 | HospAd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **1** | 65.0 | 98.0 | 35.67 | NaN | 93.0 | NaN | 11.0 | NaN | NaN | NaN | ... | NaN | NaN | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **2** | 64.0 | 99.0 | 35.72 | NaN | 95.0 | NaN | 14.5 | NaN | NaN | NaN | ... | NaN | NaN | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **3** | 65.0 | 99.5 | 35.61 | NaN | 102.0 | NaN | 15.0 | NaN | NaN | 14.0 | ... | 9.2 | 150.0 | 123.0 | 67 | 0 | NaN | NaN | -285.33 |
| **4** | 64.0 | 100.0 | 35.56 | NaN | 95.0 | NaN | 12.5 | NaN | -10.0 | 14.0 | ... | 9.2 | 150.0 | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **5** | 58.0 | 100.0 | 35.56 | 160.5 | 102.5 | 79.5 | 12.0 | NaN | -10.0 | NaN | ... | 12.8 | 214.0 | 121.0 | 67 | 0 | NaN | NaN | -285.33 |
| **6** | 55.0 | 99.0 | NaN | 154.0 | 108.0 | 79.0 | 15.0 | NaN | NaN | NaN | ... | 12.8 | 214.0 | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **7** | 57.0 | 100.0 | NaN | 154.0 | 105.0 | 76.0 | 12.0 | NaN | -9.0 | NaN | ... | NaN | NaN | NaN | 67 | 0 | NaN | NaN | -285.33 |
| **8** | 58.0 | 100.0 | NaN | 169.0 | 117.0 | 85.0 | 15.0 | NaN | -9.0 | 14.0 | ... | 11.1 | 197.0 | 104.0 | 67 | 0 | NaN | NaN | -285.33 |
| **9** | 60.0 | 100.0 | NaN | 143.0 | 100.0 | 73.0 | 17.0 | NaN | -9.5 | 14.0 | ... | 11.1 | 197.0 | NaN | 67 | 0 | NaN | NaN | -285.33 |

10 rows × 41 columns

我们可以使用`matplotlib`对时序数据进行可视化：

In [11]:

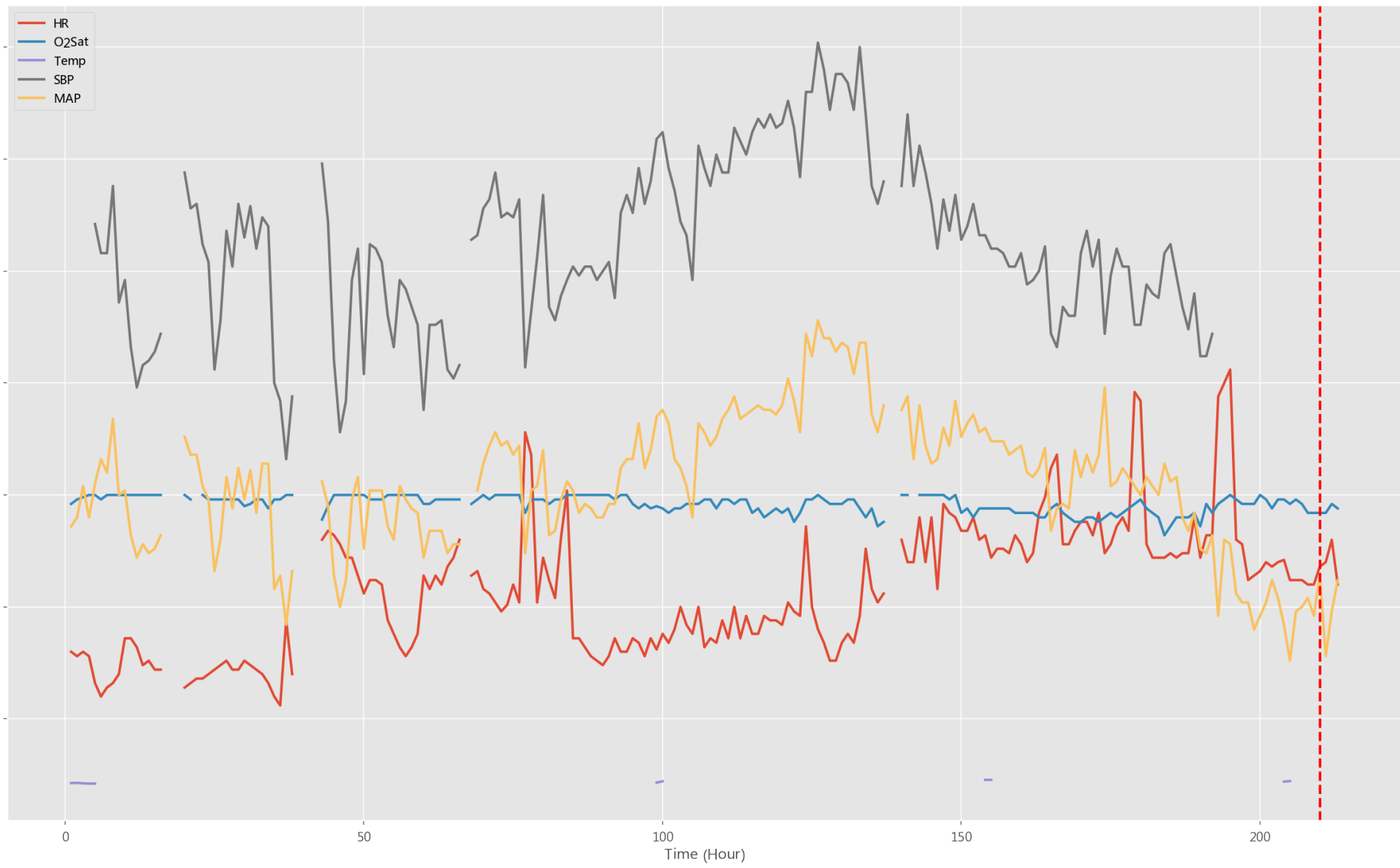
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

# 使用ggplot作为输出格式
plt.style.use('ggplot')

# 设定字符大小为14号
plt.rcParams['font.size'] = 14
```

In [12]:

```
# 对时序数据集中前5个生理指标进行可视化
cols = sepsis_df.columns.tolist()[:5]
# 获取sepsis发生时间
t_sepsis_onset = np.min(np.where(sepsis_df['SepsisLabel'] == 1)) + 6
plt.figure(figsize=(20, 12), dpi=150)
for i in range(5):
    plt.plot(sepsis_df[cols[i]], label=cols[i], linewidth=2)
plt.axvline(x=t_sepsis_onset, c="r", ls="--", lw=2)
plt.legend(loc='best')
plt.xlabel("Time (Hour)")
```

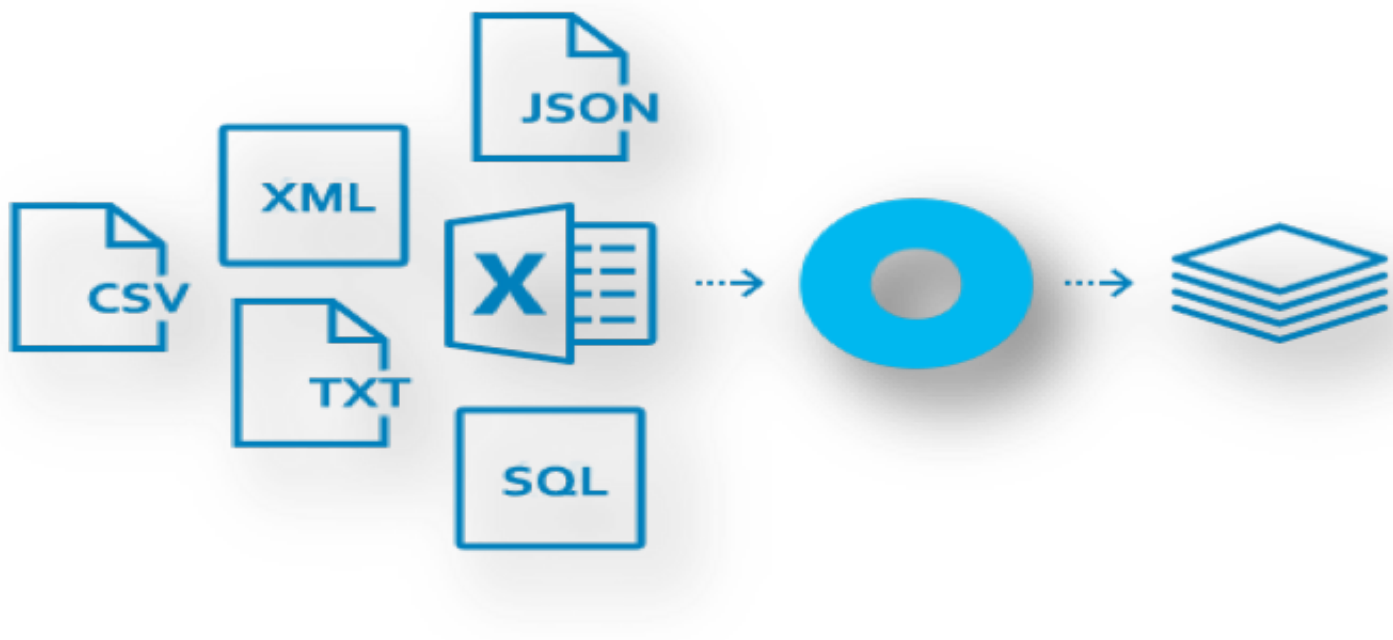
Out[12]:

```
Text(0.5, 0, 'Time (Hour)')
```




## Step 1: 数据整合（Data Integration）

数据整合指的是将来自**不同源头**的数据进行汇聚集中，例如将来自不同数据库，不同数据形式的数据集转换成统一的CSV文件。




### 关键技术点：自然语言处理（NLP）

医疗数据中很大的一部分内容是非结构化的报告形式，例如CT报告、超声报告，我们需要借助自然语言处理方法将非结构化的文本进行提取。

The patient is a frail 88-year-old caucasian male was admitted to our hospital for complaints of nausea and vomiting and suspected urinary tract infection.

He has a past medical history of hypertension, atrial fibrillation and chronic right hip pain after total hip replacement in 2012.

The patient was started on antibiotics. Urine culture confirmed an E. coli urinary tract infection sensitive to trimethoprim.

During admission an episode of possible coffee ground vomiting coupled with his non-steroidal inflammatory drug use prompted an upper GI endoscopy at which no abnormality was detected. Fecal occult blood was negative.

The patient was also provided with physiotherapy and fully remobilised.

**Clinical Findings**

| Concept ID | Preferred term | Finding context | Temporal context | Subject relationship context |
|---|---|---|---|---|
| 16932000 | Nausea and vomiting | Known present | Current or specified time | Subject of record |
| 68566005 | Urinary tract infectious disease | Suspected | Current or specified time | Subject of record |
| 38341003 | Hypertensive disorder | Known present | Current or past | Subject of record |
| 49436004 | Atrial fibrillation | Known present | Current or past | Subject of record |
| 49218002 | Hip pain | Known present | Current or past | Subject of record |
| 301011002 | Escherichia coli urinary tract infection | Known present | Current or past | Subject of record |
| 40835002 | Coffee ground vomiting | Possible | Current or specified time | Subject of record |
| 167667006 | Fecal occult blood: negative | Known present | Current or specified time | Subject of record |

**Procedures**

| Concept ID | Preferred term | Procedure context | Temporal context | Subject relationship context |
|---|---|---|---|---|
| 52734007 | Total replacement of hip | Done | Past | Subject of record |
| 117010004 | Urine culture | Done | Current or specified time | Subject of record |
| 76009000 | Esophagogastroduodenoscopy | Done | Current or specified time | Subject of record |
| 91251008 | Physical therapy procedure | Done | Current or specified time | Subject of record |

In [13]:

```
### 实践操作1：结合前一部分的“数据提取”， 请将MIMIC及eICU中的老年患者进行整合，查看他们年龄的分布
```

## Step 2: 数据清洗 （Data Clean）

从EHR中抽取的数据通常是“脏乱差”，我们需要使用不同方法对其进行“清洗”：

- 数据缺失
  - 原因：随机性的系统故障、人为失误、数据采集机制不健全等
- 噪声数据或离群值
  - 原因：数据采集器械的技术局限性、人为录入错误
- 不一致数据

### 缺失值（Missing data）

**案例：美国大型EHR数据缺失**

RECEIVED 3 September 2015
REVISED 21 January 2016
ACCEPTED 31 January 2016
PUBLISHED ONLINE FIRST 14 April 2016

# Missing clinical and behavioral health data in a large electronic health record (EHR) system

Jeanne M Madden,[1,2] Matthew D Lakoma,[2] Donna Rusinak,[2] Christine Y Lu,[2] and Stephen B Soumerai[2]

## ABSTRACT

**Objective** Recent massive investment in electronic health records (EHRs) was predicated on the assumption of improved patient safety, research capacity, and cost savings. However, most US health systems and health records are fragmented and do not share patient information. Our study compared information available in a typical EHR with more complete data from insurance claims, focusing on diagnoses, visits, and hospital care for depression and bipolar disorder.

**Methods** We included insurance plan members aged 12 and over, assigned throughout 2009 to a large multispecialty medical practice in Massachusetts, with diagnoses of depression ($N = 5140$) or bipolar disorder ($N = 462$). We extracted insurance claims and EHR data from the primary care site and compared diagnoses of interest, outpatient visits, and acute hospital events (overall and behavioral) between the 2 sources.

**Results** Patients with depression and bipolar disorder, respectively, averaged 8.4 and 14.0 days of outpatient behavioral care per year; 60% and 54% of these, respectively, were missing from the EHR because they occurred offsite. Total outpatient care days were 20.5 for those with depression and 25.0 for those with bipolar disorder, with 45% and 46% missing, respectively, from the EHR. The EHR missed 89% of acute psychiatric services. Study diagnoses were missing from the EHR's structured event data for 27.3% and 27.7% of patients.

**Conclusion** EHRs inadequately capture mental health diagnoses, visits, specialty care, hospitalizations, and medications. Missing clinical information raises concerns about medical errors and research integrity. Given the fragmentation of health care and poor EHR interoperability, information exchange, and usability, priorities for further investment in health IT will need thoughtful reconsideration.

> “Federal efforts have been undermined by the inherently fragmentary nature of the U.S. health care system and the proliferation of proprietary EHR systems that communicate poorly with each other… Going forward, our national policy for investment in EHR needs to be re-examined to **surmount the fragmentation** that currently exists in U.S. health care and to set robust technical standards for **interoperability and data quality**.”

**严重性**

每年美国急救医学研究所（Emergency Care Research Institute, ECRI）都会在其网站发布《十大医疗技术危害》，2015年，“数据缺失”榜上有名！

HEALTH IT, HOSPITALS, MEDICAL DEVICES, SYN,

# Missing EHR data rises up top 10 list of medical technology hazards to patient safety

Every year the ECRI Institute assembles a top 10 list of healthcare technology hazards that [...]

By STEPHANIE BAUM

Post a comment / Nov 25, 2014 at 5:39 PM

**缺失值产生机制**

- 完全随机缺失(Missing Completely At Random)：数据缺失和缺失值及观测值都无关
  - e.g 医生随机性地忘记记录身高、体重
- 随机缺失(Missing At Random)：数据缺失仅和观测值相关
  - e.g 老年患者可能会忘记告知医生自己是否患过肺炎，在这种情况下，肺炎这个变量的缺失情况与年龄这个变量相关。
- 非随机缺失(Missing Not At Random)：数据缺失同时和缺失值及观测值相关
  - e.g 血压正常患者可能会缺失血压测量记录

**缺失值可视化**

在正式对缺失值进行处理前，对缺失情况进行可视化是能够为接下来选择合适的处理方法打下坚实的基础。

例如，从缺失情况的分布、相关性上我们可以推测出缺失的机制，方便我们后续选择合适的处理方法。

我们所使用的可视化工具是Python上的missingno

> Bilogur, (2018). Missingno: a missing data visualization suite. Journal of Open Source Software, 3(22), 547, https://doi.org/10.21105/joss.00547

In [14]:

```
import missingno as msmo
```

***a. 缺失比例柱状图***

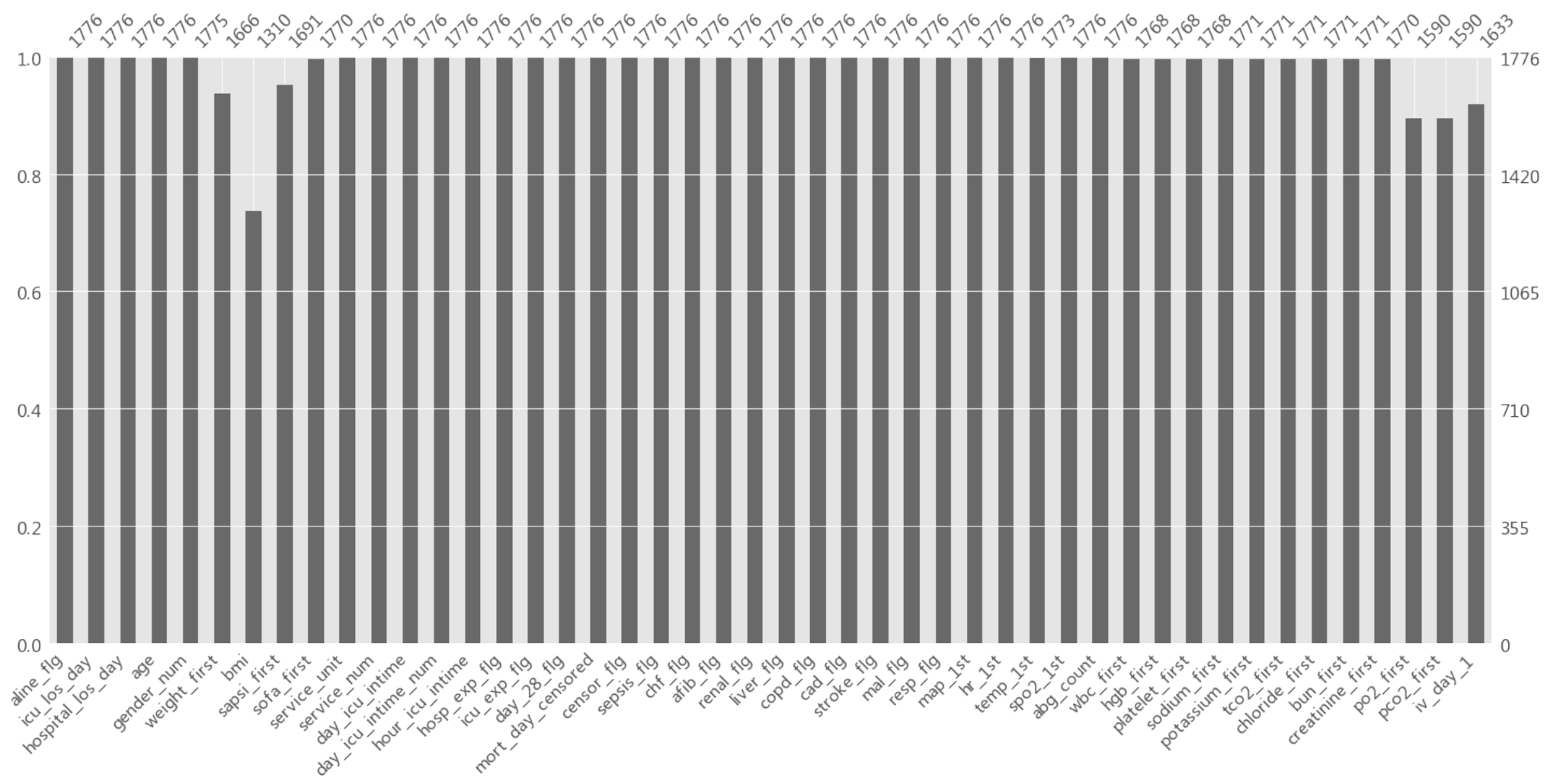
柱状图的优势是可以对缺失比例进行直观展示

In [15]:

```
msmo.bar(iac_df)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d59011ba8>
```




***b. 缺失相关性热力图***

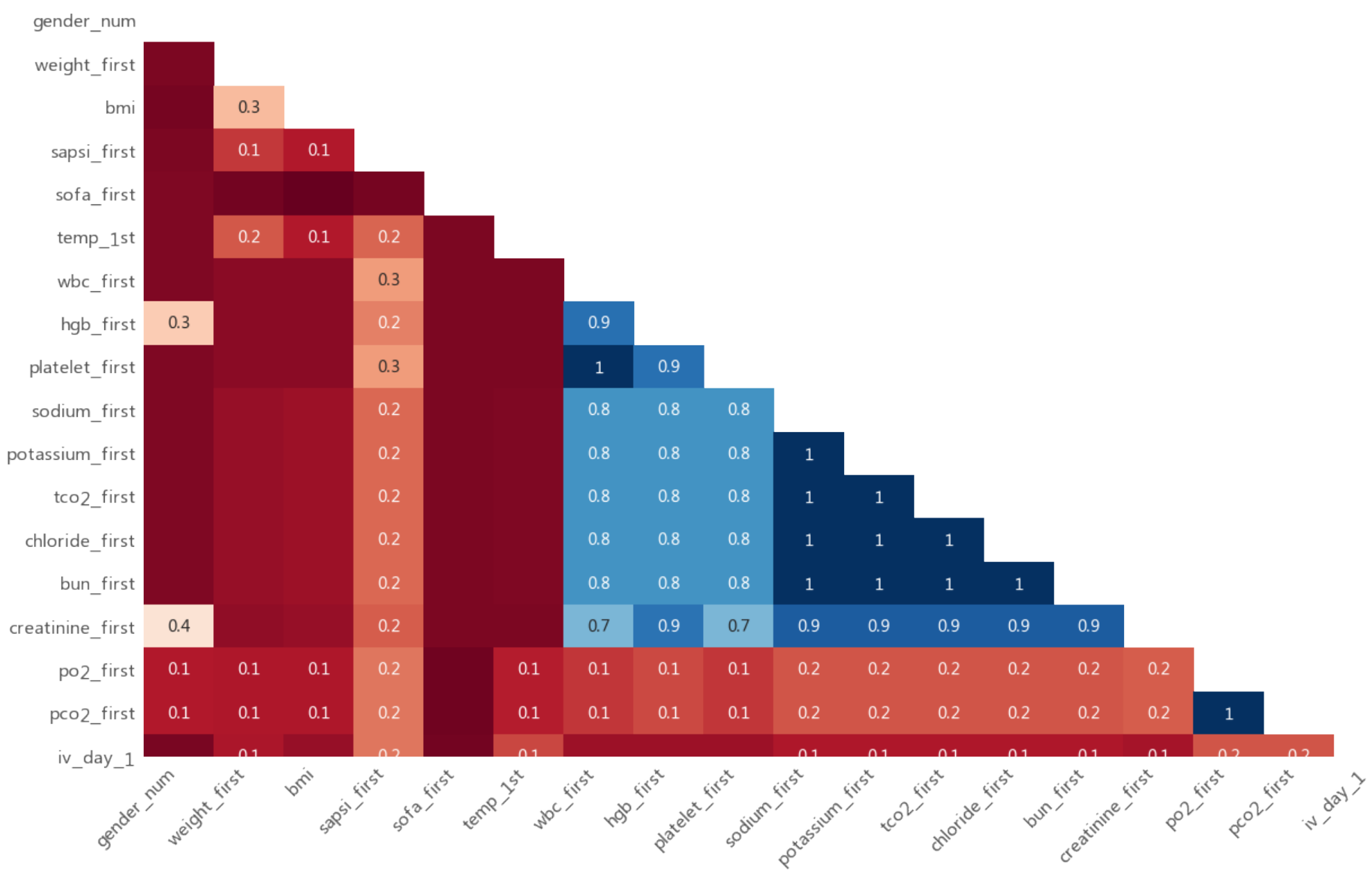
热力图的方法可以对缺失之间的相关性进行可视化

In [16]:

```
msmo.heatmap(iac_df)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d58b9f390>
```



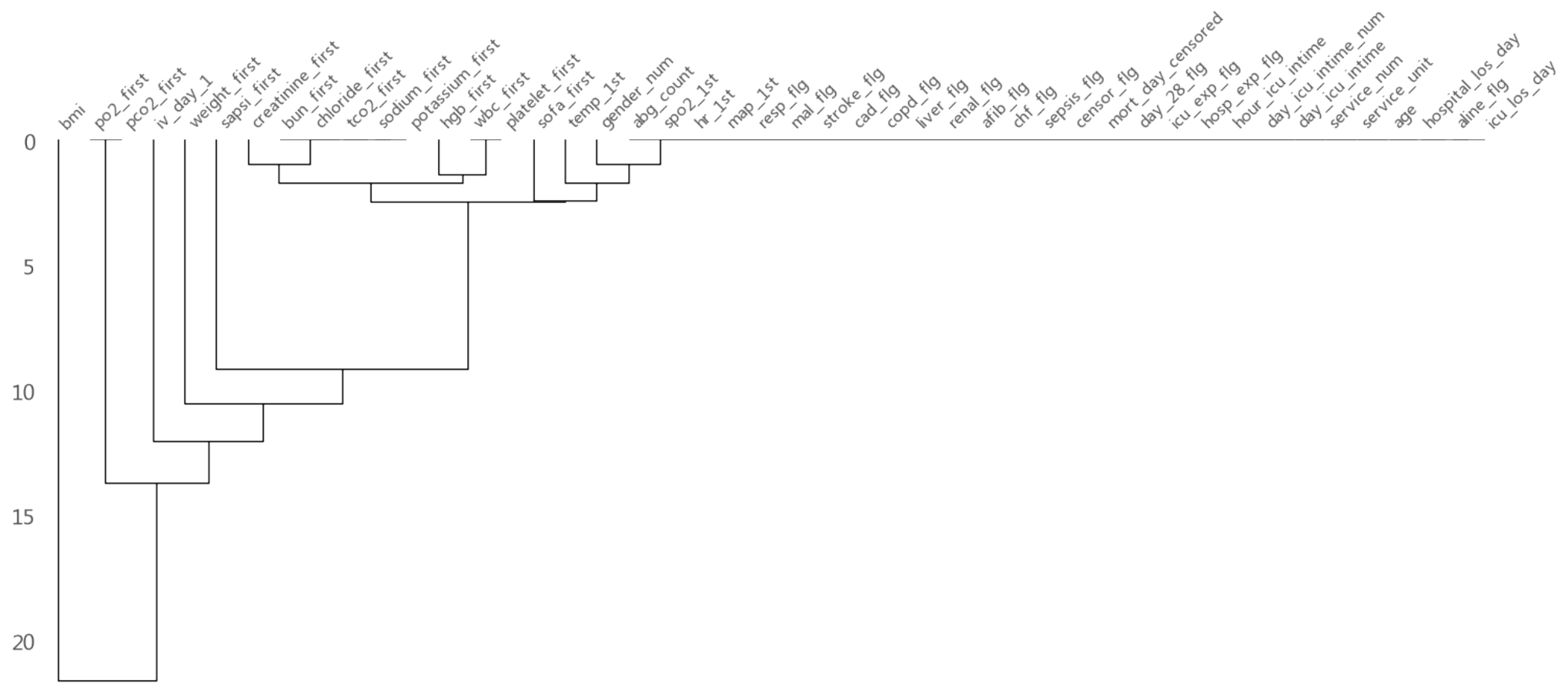

**c. 系统层级图**

In [17]:

```
msmo.dendrogram(iac_df)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d55d3e860>
```




**结构化数据处理方法**

结构化数据的处理方式包括删除法和插补法，具体的使用场景需要结合数据缺失的机制进行讨论和选择。

***删除法***

“删除法”，顾名思义，即直接删除缺失相关的记录， 根据删除的范围可分为：

- 完全案例删除（Complete Cases Analysis）
- 有效案例删除（Avaliable Cases Analysis）

`pandas`中的`dropna`提供了各种高效快捷的删除方式。

***完全案例删除***

在完全案例删除中，我们将会删除包含任何包含缺失值的记录。

这样做的前提假设是 **删除后的子集能够代表样本**。

*请注意，这种方法仅限于完全随机缺失的情况。*

In [18]:

```
complete_df = iac_df.dropna(how='any')
complete_df.head(10)
```

Out[18]:

| | aline_flg | icu_los_day | hospital_los_day | age | gender_num | weight_first | bmi | sapsi_first | sofa_first | service_unit | ... | platelet_firs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 7.63 | 13 | 72.36841 | 1.0 | 75.0 | 29.912791 | 15.0 | 9.0 | SICU | ... | 354.0 |
| **2** | 0 | 2.86 | 5 | 36.50000 | 0.0 | 70.0 | 27.118272 | 16.0 | 5.0 | MICU | ... | 295.0 |
| **4** | 1 | 1.75 | 5 | 23.74217 | 1.0 | 95.2 | 28.464563 | 18.0 | 7.0 | SICU | ... | 22.0 |
| **6** | 1 | 7.06 | 27 | 24.64717 | 1.0 | 90.0 | 25.474850 | 15.0 | 6.0 | SICU | ... | 130.0 |
| **7** | 1 | 15.30 | 33 | 78.15970 | 0.0 | 69.7 | 27.219757 | 16.0 | 8.0 | MICU | ... | 20.0 |
| **8** | 1 | 3.79 | 4 | 71.43198 | 0.0 | 52.6 | 21.910820 | 9.0 | 5.0 | MICU | ... | 238.0 |
| **9** | 1 | 7.14 | 7 | 25.41667 | 0.0 | 61.5 | 21.543184 | 13.0 | 9.0 | MICU | ... | 137.0 |
| **10** | 1 | 1.13 | 2 | 51.18725 | 1.0 | 91.1 | 32.772315 | 16.0 | 5.0 | SICU | ... | 423.0 |
| **11** | 0 | 3.71 | 6 | 75.80626 | 0.0 | 96.9 | 39.418143 | 18.0 | 7.0 | MICU | ... | 262.0 |
| **12** | 1 | 3.23 | 11 | 52.82878 | 1.0 | 74.0 | 28.002986 | 14.0 | 12.0 | SICU | ... | 91.0 |

10 rows × 46 columns

***可用案例删除***

而可用案例删除则会根据具体的分析需求，删除相关变量的缺失记录。

In [19]:

```
avaliable_df = iac_df.dropna(how='all', subset=['wbc_first'])
avaliable_df.shape
```

Out[19]:

```
(1768, 46)
```

**插补法（Imputation）**

“插补法”，将缺失值通过某些值进行替代。

In [20]:

```
def visualize_imputation(raw_data, imputed_data):
    """对插补前后的效果进行可视化
    raw_data: pandas.Series
        原始数据
    imputed_data: pandas.Series
        插补后数据
    """
    # 导入pyplot及numpy
    import matplotlib.pyplot as plt
    import numpy as np
    # 根据最大最小值确定bins的数目
    bins = np.linspace(raw_data.min(), raw_data.max(), 40)
    plt.figure(figsize=(8, 4), dpi=150)
    # histogram
    plt.hist([raw_data, imputed_data], bins, alpha=0.5, label=['raw', 'imputed'])
    plt.legend(loc='upper right')
    plt.show()
```

***中值插补***

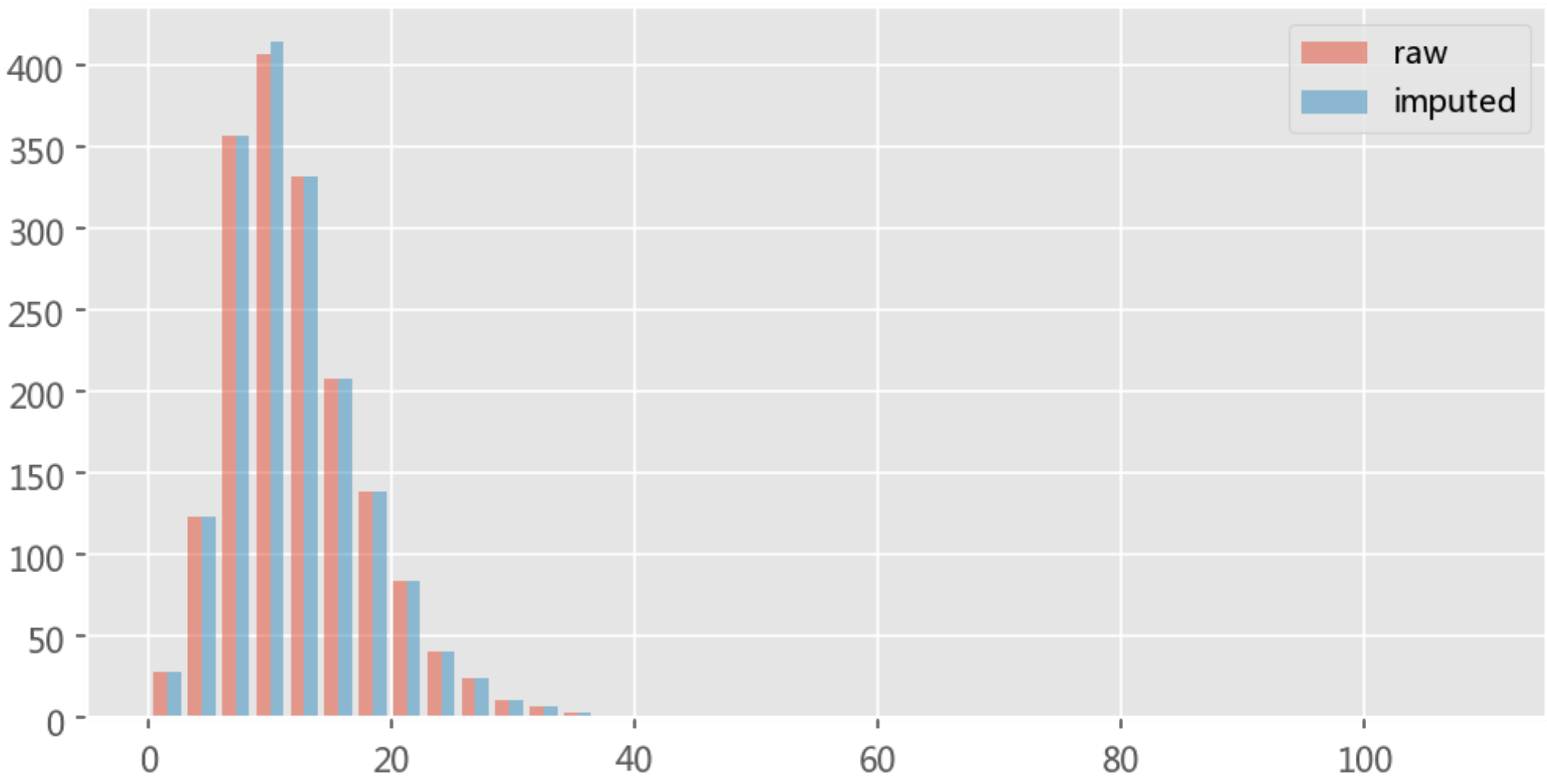
用该列的中位数替代缺失值

In [21]:

```
med_impute = iac_df.fillna(value=iac_df.median())
```

In [22]:

```
visualize_imputation(iac_df['wbc_first'], med_impute['wbc_first'])
```




***均值插补***

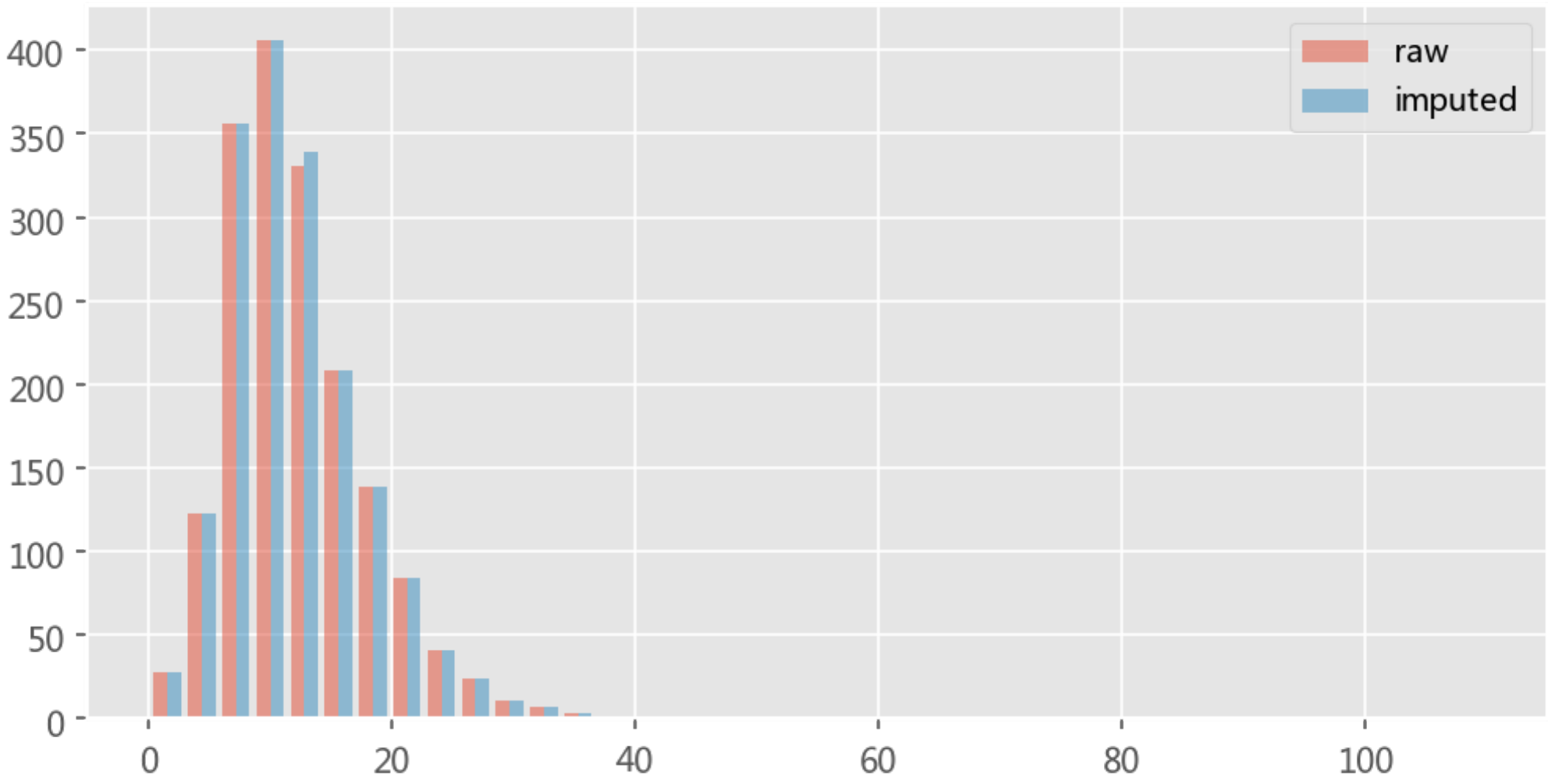
用该列的平均值替代缺失值

In [23]:

```
mean_impute = iac_df.fillna(value=iac_df.mean())
```

In [24]:

```
visualize_imputation(iac_df['wbc_first'], mean_impute['wbc_first'])
```




***kNN插补***

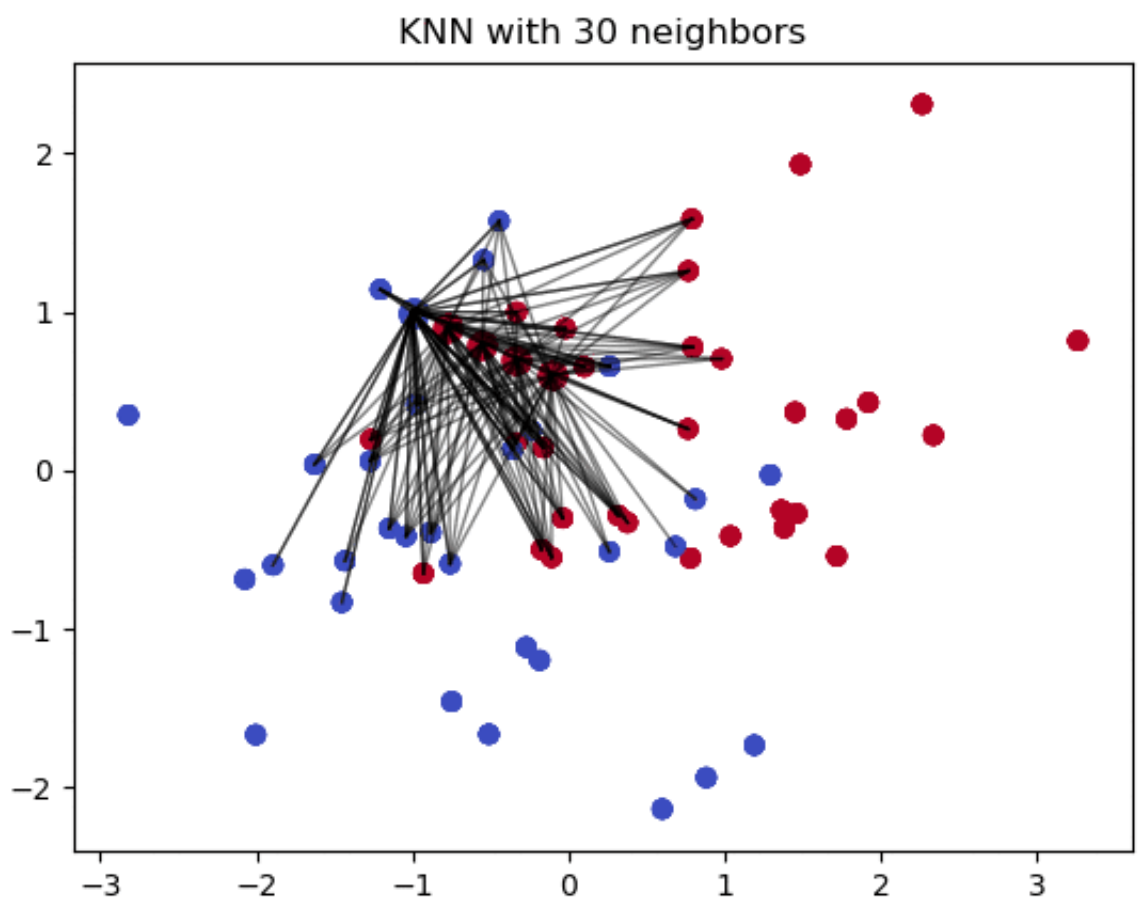
kNN即“k Nearest Neighbours”，又称“k近邻”，是机器学习领域一种基础的无监督方法。

k近邻也可用于处理缺失值。这里的缺失值将会被k个值的均值填补，这k个值是与缺失个案最相似的k个完整的观测值。把数据集归一化之后，两个个案的相似性是确定的，距离函数可以选择欧几里得距离、曼哈顿距离、马氏距离、皮尔逊距离等等。

kNN算法的主要优点是，给定足够的数据可以以合理的精度预测一个点周围的条件概率分布，从而得到比较好的估计，它能预测定性的和定量的（离散的和连续的）分布。该方法的另外一个优点是考虑了数据的相关结构。

在K近邻方法中，k值的选择是至关重要的，高k值会包括与目标观测值截然不同的分布，然而，低k值会丢失意义重大的分布。

Python中我们可以使用`impyute`来实现kNN插补



In [67]:

```
# 安装impyute
#!pip install -i https://pypi.tuna.tsinghua.edu.cn/simple impyute
```

```
Collecting impyute
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/37/28/86829f67c9affb847facaab94687761d3555539ec675f757777
Requirement already satisfied: numpy in /opt/conda/lib/python3.6/site-packages (from impyute)
Requirement already satisfied: scipy in /opt/conda/lib/python3.6/site-packages (from impyute)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.6/site-packages (from impyute)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.6/site-packages (from scikit-learn->impyute)
Installing collected packages: impyute
Successfully installed impyute-0.0.8
You are using pip version 9.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [1]:

```
# 从fancyimpute 中导入KNN
from impyute.imputation.cs import fast_knn
```

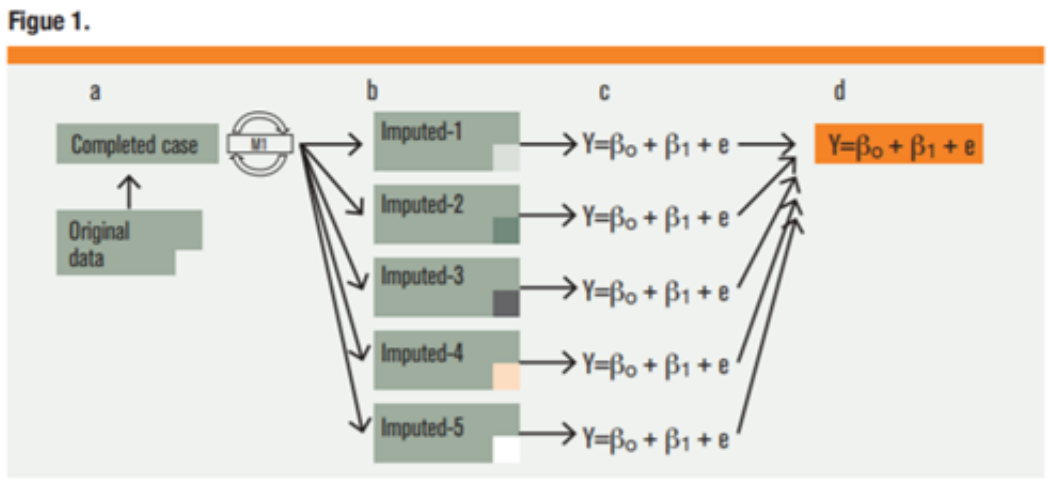
In [ ]:

```
#使用距离相近的三条记录计算插值
knn_missing = iac
#imputed_training=fast_knn(train.values, k=3)
```

**多重插补**

为了分析包含缺失值的数据集，Rubin在20世纪70年代提出了多值插值（MI）的方法，之后逐渐发展为强大的统计技术[7,16]。它是一个蒙特卡洛方法，需要3个步骤:

- 插值，可以任意选择插值方法，生成填补后的完整数据集，最后会产生M≥2个数据集（通常5-10个足以）。在这些数据集中，所有的观测值是相同的，但是填补值是不同的，这反映出插值的不确定性。
- 分析：我们对每个完整数据集进行分析（比如其中一个数据集用logistic回归预测患者的死亡率进行插补），M个数据集即产生M个结果。
- 合并：把M个分析结果整合为一个最终的结果，例如计算M个分析的均值（和95%置信区间）。




a- original data set with missing data (white cut-out portion); b- series of imputed data sets, each with a (randomly) different imputed value for the missing data; c- series of corresponding outcome regression equations (one for each imputed data set); d- final outcome regression model, which is a weighted average of equations in c.

In [2]:

```
from impyute.imputation.cs import mice
```

**时序数据**

对于时序数据的处理方式是进行插值处理。接下来我们以演示数据集中的一些包含缺失值的变量进行展示。

In [28]:

```
# 选取部分变量
sepsis_subset_df = sepsis_df[['HR', 'MAP', 'Resp']]
sepsis_subset_df.shape
```

Out[28]:

```
(214, 3)
```

In [29]:

```
# 统计缺失变量数目
sepsis_subset_df.isna().sum(axis=0)
```

Out[29]:

```
HR      11
MAP     12
Resp    13
dtype: int64
```

***a. 前向/后向插值***

In [30]:

```
# 前向插值
forwardfill_df = sepsis_subset_df.fillna(method='ffill')
# 后向插值
backwardfill_df = sepsis_subset_df.fillna(method='bfill')
```

***b. 线性插值***

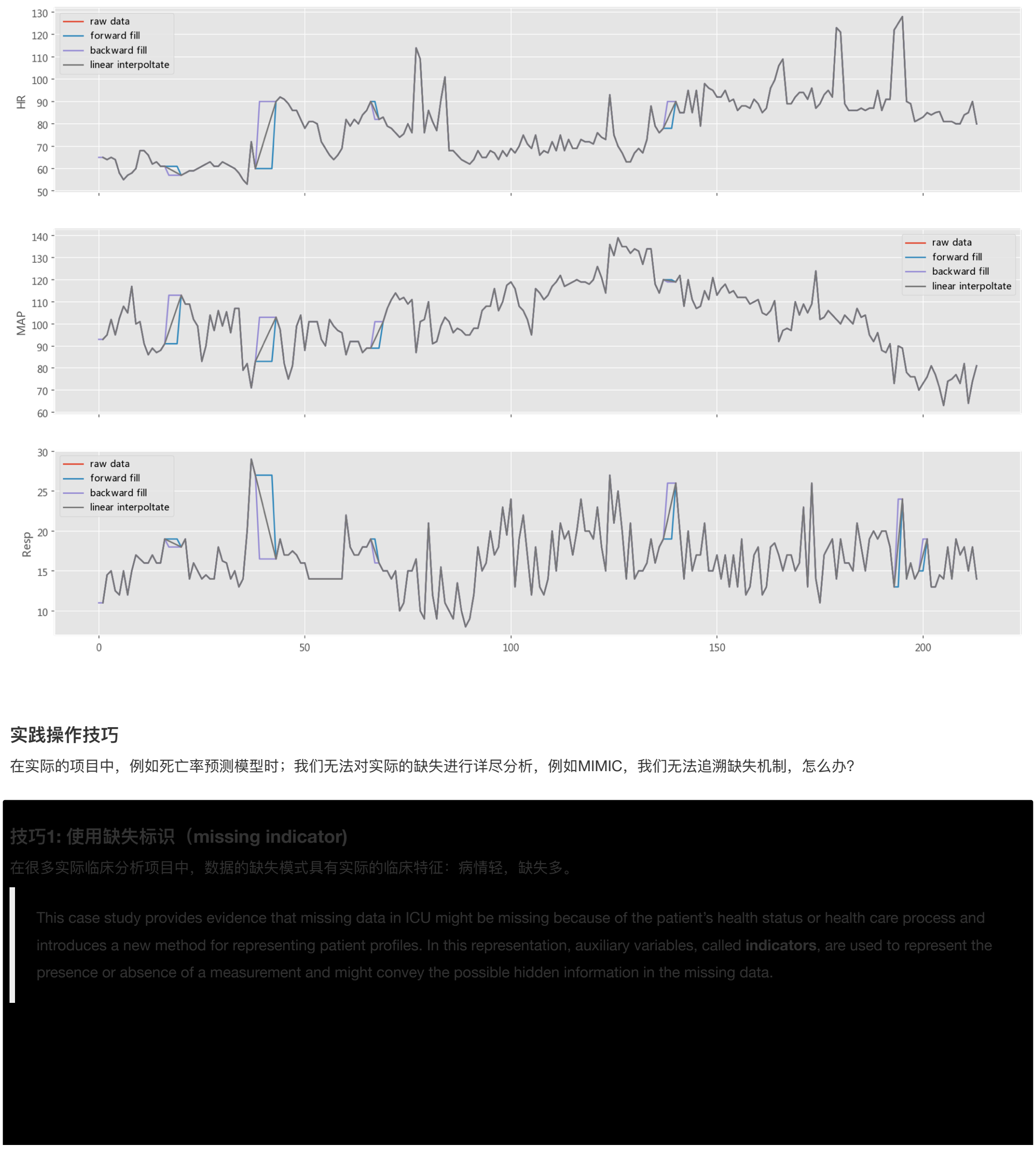
In [31]:

```
# 线性插值
interpolate_df = sepsis_subset_df.interpolate(method ='linear')
```

***c. 效果对比***

In [32]:

```
figs, axes = plt.subplots(3, 1, sharex=True, figsize=(18, 12), dpi=150)
cols = ['HR', 'MAP', 'Resp']
for i in range(3):
    axes[i].plot(sepsis_subset_df[cols[i]], label='raw data')
    axes[i].plot(forwardfill_df[cols[i]], label='forward fill')
    axes[i].plot(backwardfill_df[cols[i]], label='backward fill')
    axes[i].plot(interpolate_df[cols[i]], label='linear interpoltate')
    axes[i].legend(loc='best')
    axes[i].set_ylabel(cols[i])
```




## 实践操作技巧

在实际的项目中，例如死亡率预测模型时；我们无法对实际的缺失进行详尽分析，例如MIMIC，我们无法追溯缺失机制，怎么办?

### 技巧1: 使用缺失标识（missing indicator)

在很多实际临床分析项目中，数据的缺失模式具有实际的临床特征：病情轻，缺失多。

> This case study provides evidence that missing data in ICU might be missing because of the patient's health status or health care process and introduces a new method for representing patient profiles. In this representation, auxiliary variables, called **indicators**, are used to represent the presence or absence of a measurement and might convey the possible hidden information in the missing data.

Original Paper

# A New Insight Into Missing Data in Intensive Care Unit Patient Profiles: Observational Study

Anis Sharafoddini[1], MSc; Joel A Dubin[1,2], PhD; David M Maslove[3], MD, MS, FRCPC; Joon Lee[1,4,5], PhD

[1]Health Data Science Lab, School of Public Health and Health Systems, University of Waterloo, Waterloo, ON, Canada

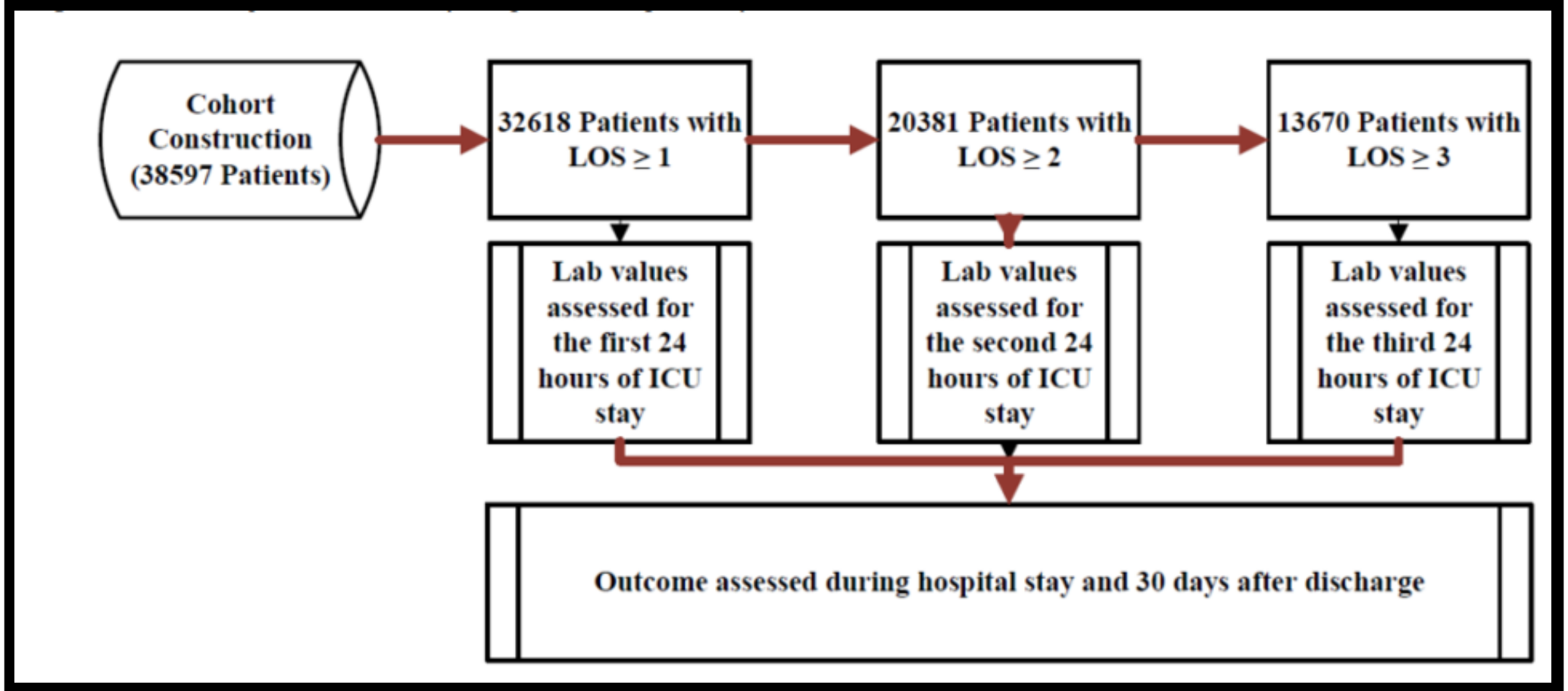
[2]Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON, Canada

[3]Department of Critical Care Medicine, Queen's University, Kingston, ON, Canada

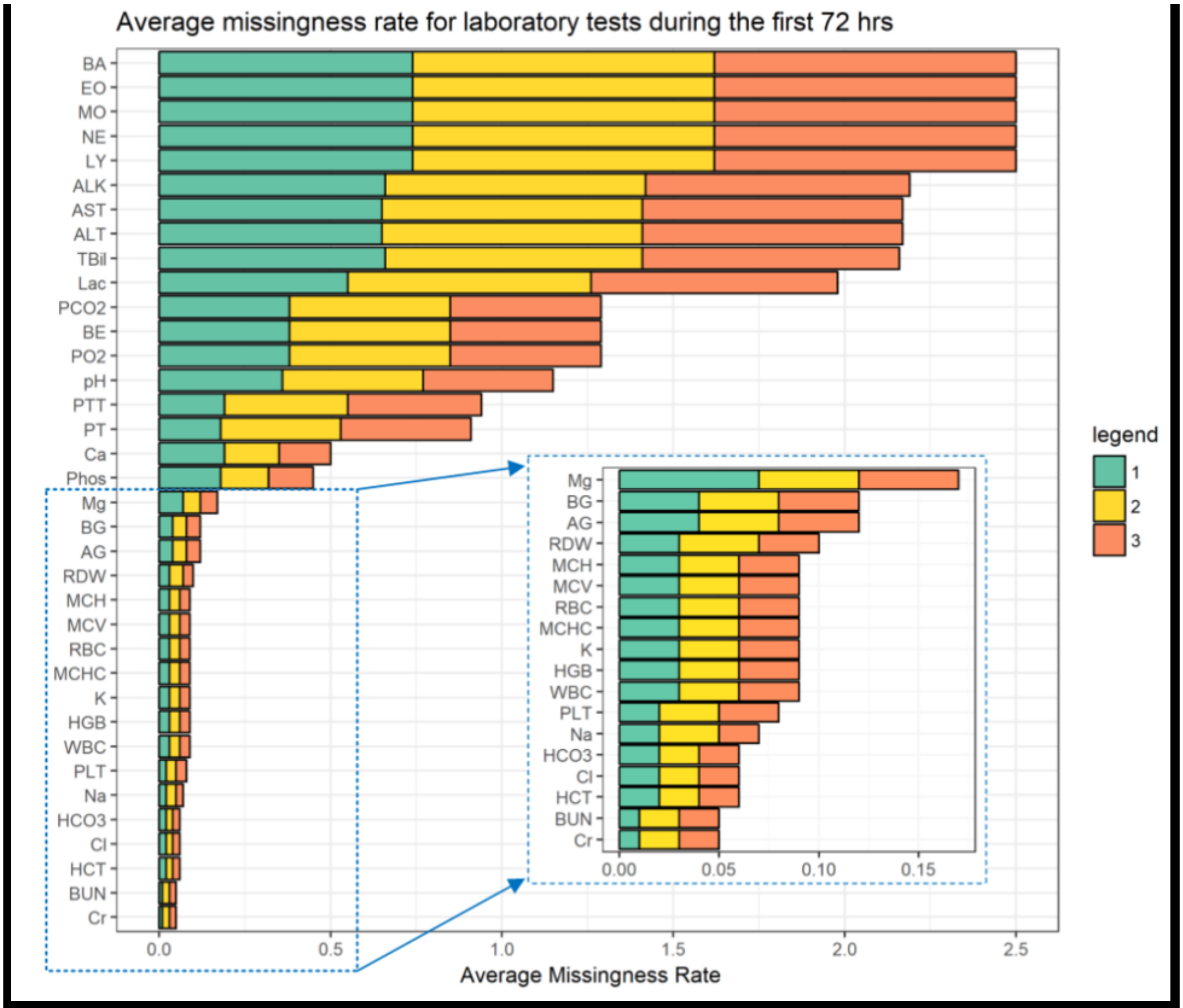
[4]Department of Community Health Sciences, Cumming School of Medicine, University of Calgary, Calgary, AB, Canada

[5]Department of Cardiac Sciences, Cumming School of Medicine, University of Calgary, Calgary, AB, Canada

分析流程




缺失情况


Average missingness rate for laboratory tests during the first 72 hrs
BA
EO
MO
NE
LY
ALK
AST
ALT
TBil
Lac
PCO2
BE
PO2
pH
PTT
PT
Ca
Phos
Mg
BG
AG
RDW
MCH
MCV
RBC
MCHC
K
HGB
WBC
PLT
Na
HCO3
Cl
HCT
BUN
Cr
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.05
0.10
0.15
Average Missingness Rate
legend
1
2
3

具体方法

Imputed data matrix

| | ALT | BG | ⋮ | ⋮ | pH | BUN |
|---|---|---|---|---|---|---|
| Patient #1 | 23 | 117 | … | … | *7.5* | *20* |
| Patient #2 | 68 | *67* | … | … | 7.3 | 17 |
| … | … | … | … | … | … | … |
| … | … | … | … | … | … | … |
| Patient #(n-1) | 56 | 100 | … | … | 7.4 | *12* |
| Patient #n | *17* | 78 | … | … | 7.2 | 25 |

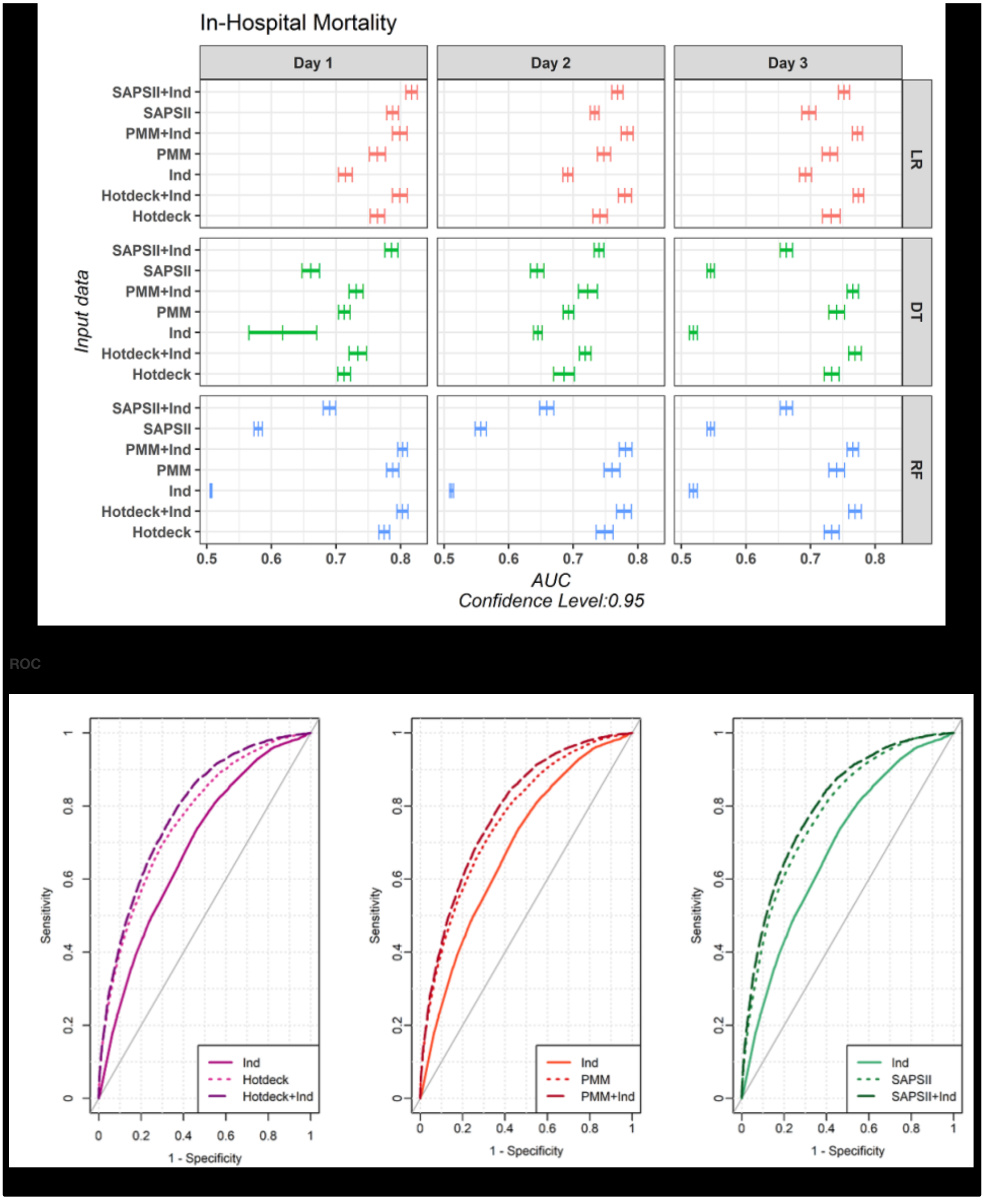
Auxiliary matrix

| | I-ALT | I-BG | ⋮ | ⋮ | I-pH | I-BUN |
|---|---|---|---|---|---|---|
| Patient #1 | 0 | 0 | … | … | 1 | 1 |
| Patient #2 | 0 | 1 | … | … | 0 | 0 |
| … | … | … | … | … | … | … |
| … | … | … | … | … | … | … |
| Patient #(n-1) | 0 | 0 | … | … | 0 | 1 |
| Patient #n | 1 | 0 | … | … | 0 | 0 |

Augmented matrix

| | ALT | I-ALT | BG | I-BG | ⋮ | ⋮ | pH | I-pH | BUN | I-BUN |
|---|---|---|---|---|---|---|---|---|---|---|
| Patient #1 | 23 | 0 | 117 | 0 | … | … | *7.5* | 1 | *20* | 1 |
| Patient #2 | 68 | 0 | *67* | 1 | … | … | 7.3 | 0 | 17 | 0 |
| … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … |
| Patient #(n-1) | 56 | 0 | 100 | 0 | … | … | 7.4 | 0 | *12* | 1 |
| Patient #n | *17* | 1 | 78 | 0 | … | … | 7.2 | 0 | 25 | 0 |

方法表现

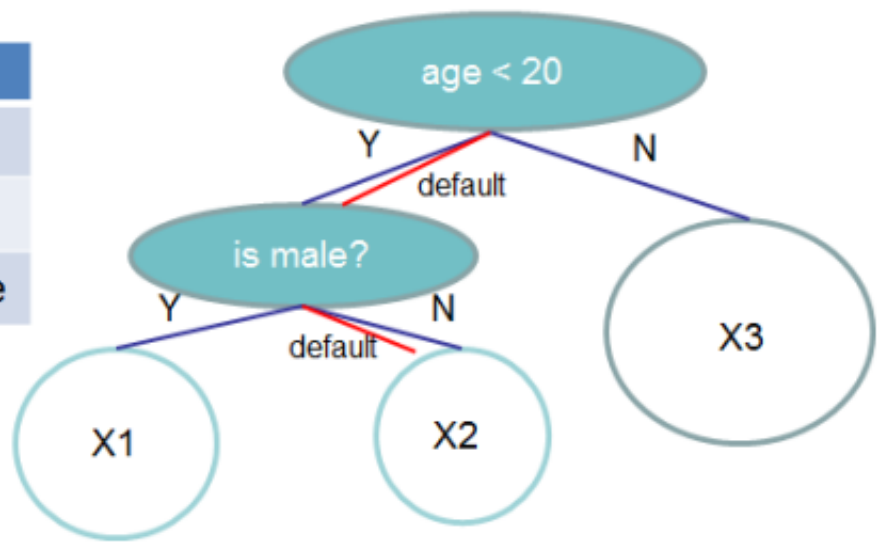



### 技巧2：使用其他预测方法

另外一个技巧是使用XGBoost或LightGBM这类对缺失值“天生免疫”的方法：

**Data**

| Example | Age | Gender |
|---|---|---|
| X1 | ? | male |
| X2 | 15 | ? |
| X3 | 25 | female |



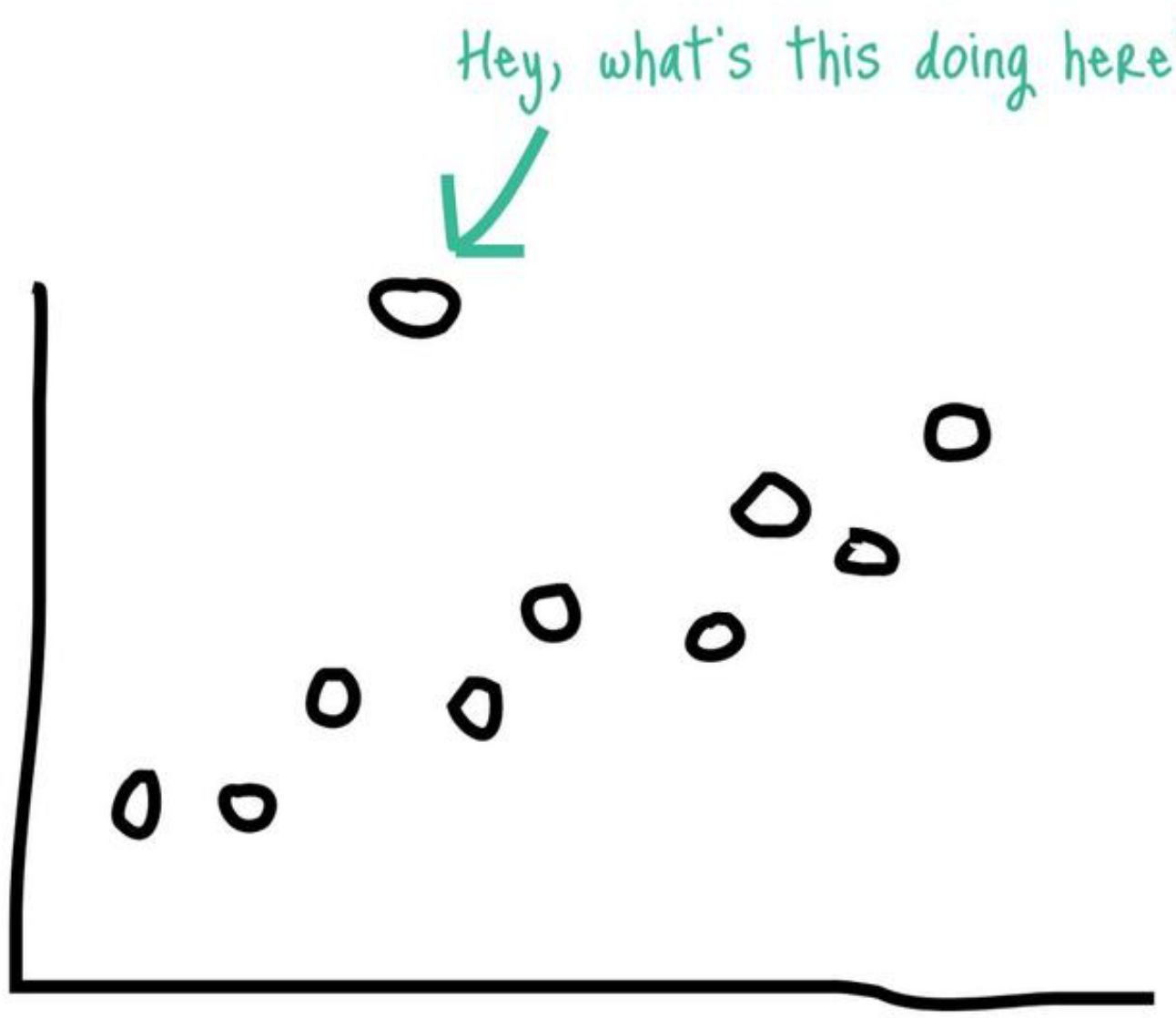
**Algorithm 3:** Sparsity-aware Split Finding

**Input**: $I$, instance set of current node
**Input**: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
**Input**: $d$, feature dimension
*Also applies to the approximate setting, only collect statistics of non-missing entries into buckets*
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
**for** $k = 1$ ***to*** $m$ **do**
  *// enumerate missing value goto right*
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  **for** $j$ *in sorted*($I_k$, *ascent order by* $\mathbf{x}_{jk}$) **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  **end**
  *// enumerate missing value goto left*
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  **for** $j$ *in sorted*($I_k$, *descent order by* $\mathbf{x}_{jk}$) **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  **end**
**Output**: Split and default directions with max gain

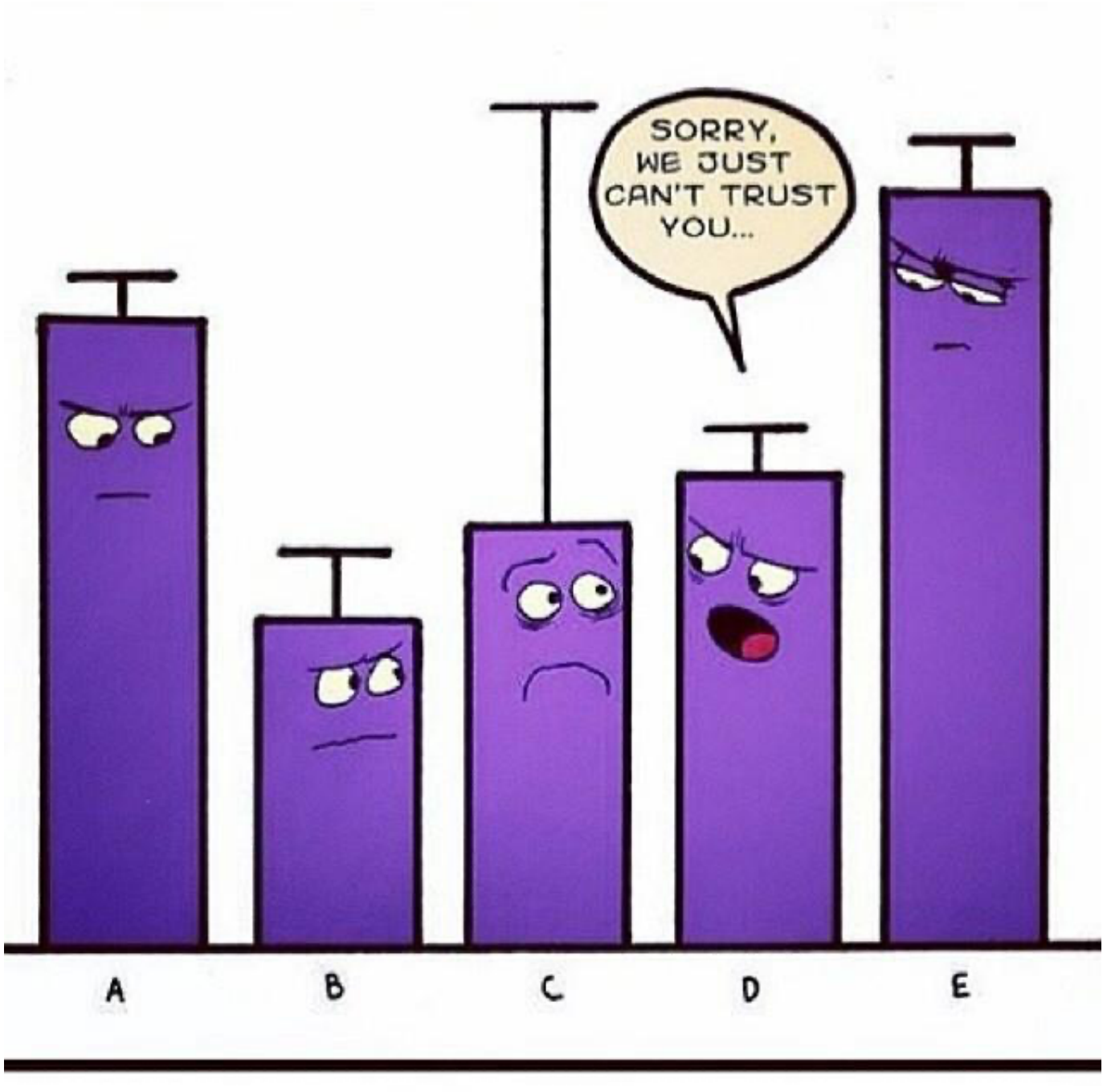
评论

**离群值（Outlier）**

离群值是指与其他数据值存在**较大差异**的值；



噪声是指被错误标识的样本（分类噪声）或者错误的属性值（属性噪声）。与噪声不同的是，离群值是一个更加广义的概念，不仅包括误差而且还包括源于观察对象或观察过程的内在变化而导致的不一致数据。




医学数据中的离群值可能来自于外界仪器故障、人为失误，也可能来自于病人自身的某些特定原因。

- 药物使用、摄入食物或饮酒
- 精神压力
- 数据采集

In [46]:

```
# 从原始数据集中提取子集用于展示
outliers_df = iac_df[[ 'map_1st', 'hr_1st', 'wbc_first', 'hgb_first', 'gender_num']]
# 剔除所有缺失值
outliers_df = outliers_df.dropna(how='any')
# 将性别值进行映射
outliers_df['gender_num'] = outliers_df['gender_num'].map({0.0: 'female', 1.0: 'male'})
print("原始数据中共包含 {} 例样本".format(outliers_df.shape[0]))
```

```
原始数据中共包含 1767 例样本
```

In [47]:

```
import seaborn as sns
```

**可视化分析**

在很多情况下，我们可以通过可视化的方法对离群值进行直观地分析。

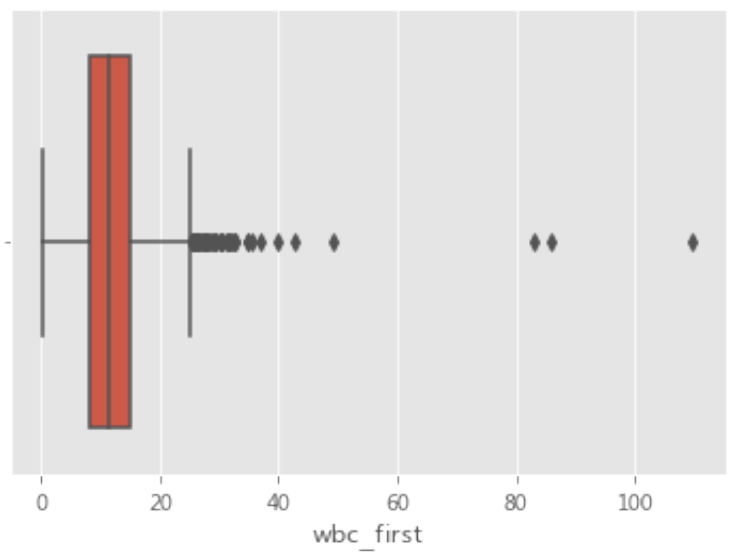
例如下图所示，分别通过箱型图（box plot）和散点图（scatter plot）对数据的分布进行可视化，展现出单变量及多变量间的数据分布。

In [48]:

```
sns.boxplot(outliers_df['wbc_first'])
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d41665828>
```




In [49]:

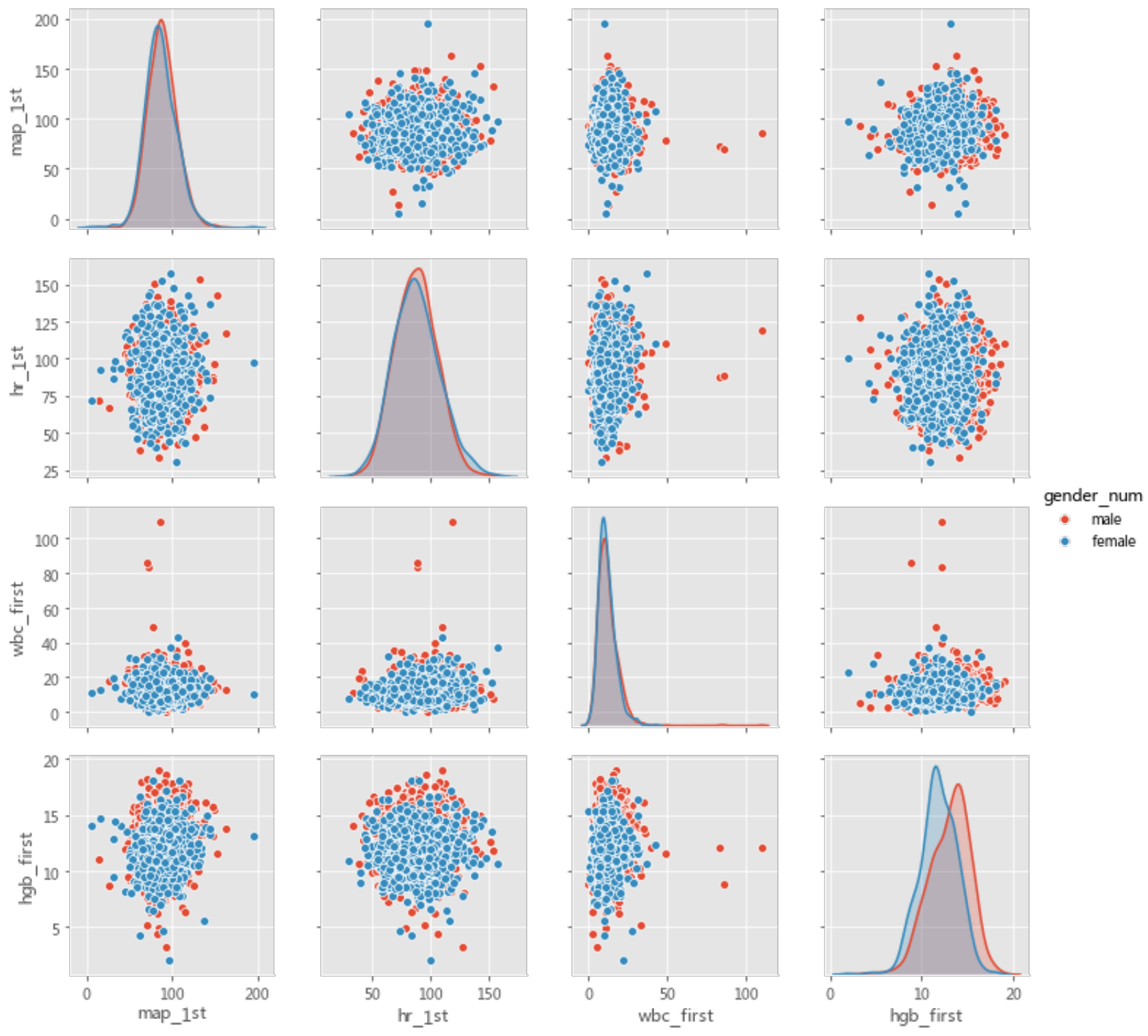
```
# 小练习：请对其他的变量通过箱型图进行可视化，并估测出异常值的范围
```

In [50]:

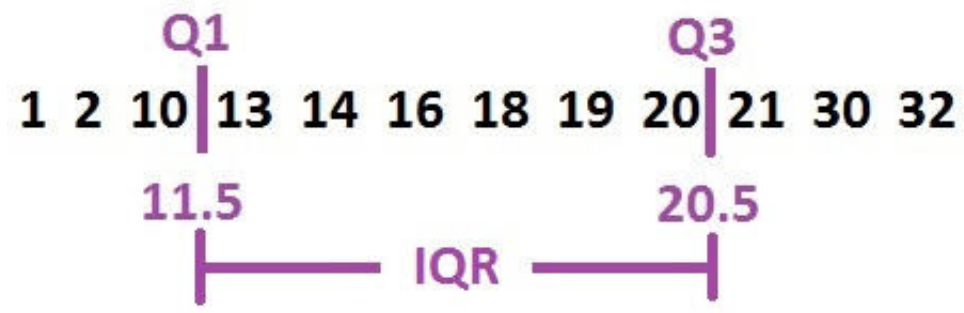
```
sns.pairplot(outliers_df, hue='gender_num')
```

Out[50]:

```
<seaborn.axisgrid.PairGrid at 0x7f6d418dc2b0>
```




**Tukey法（分位法）**




1. 假设数据集的第一分位为Q1, 第三分位为Q3, 则IQR=Q3-Q1
2. 如果数据点位于1.5倍IQR和3倍IQR之间，则**较为可能**是离群值；如果数据点大于3倍IQR，则**极为可能**是离群值。

In [51]:

```
Q1 = outliers_df.quantile(0.25)
Q3 = outliers_df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
outliers_df_iqr = outliers_df[~((outliers_df < (Q1 - 1.5 * IQR)) |(outliers_df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

```
map_1st      22.333298
hr_1st       25.000000
wbc_first     6.800000
hgb_first     3.050000
dtype: float64
```

**Z评分及改进Z评分判别**

Z分数法以标准差为单位计算数据偏移其平均值的程度。当数据为正态分布时，计算Z分数的公式为:

$$z_i = \frac{x_i - \hat{x}}{s}$$

当$|z_i| \geq 3$为离群值。

Z分数中平均值和标准差容易被极值影响，因此我们使用中位数 $\tilde{x}$以及绝对中位差$MAD = median|x_i - \tilde{x}|$重新计算

$$M_i = \frac{0.6475(x_i - \tilde{x}))}{MAD}$$

当 $M_i \geq 3.5$时为离群值。

In [52]:

```
# 从scipy中导入统计包
from scipy import stats
import numpy as np
# 计算Z评分
outliers_df = outliers_df.drop(labels=['gender_num'], axis=1)
z = np.abs(stats.zscore(outliers_df))

# 设定z评分阈值
threshold = 3

# 根据阈值滤除异常值
outliers_df_z = outliers_df[(z < 3).all(axis=1)]
```

In [53]:

```
outliers_df_z.shape
```

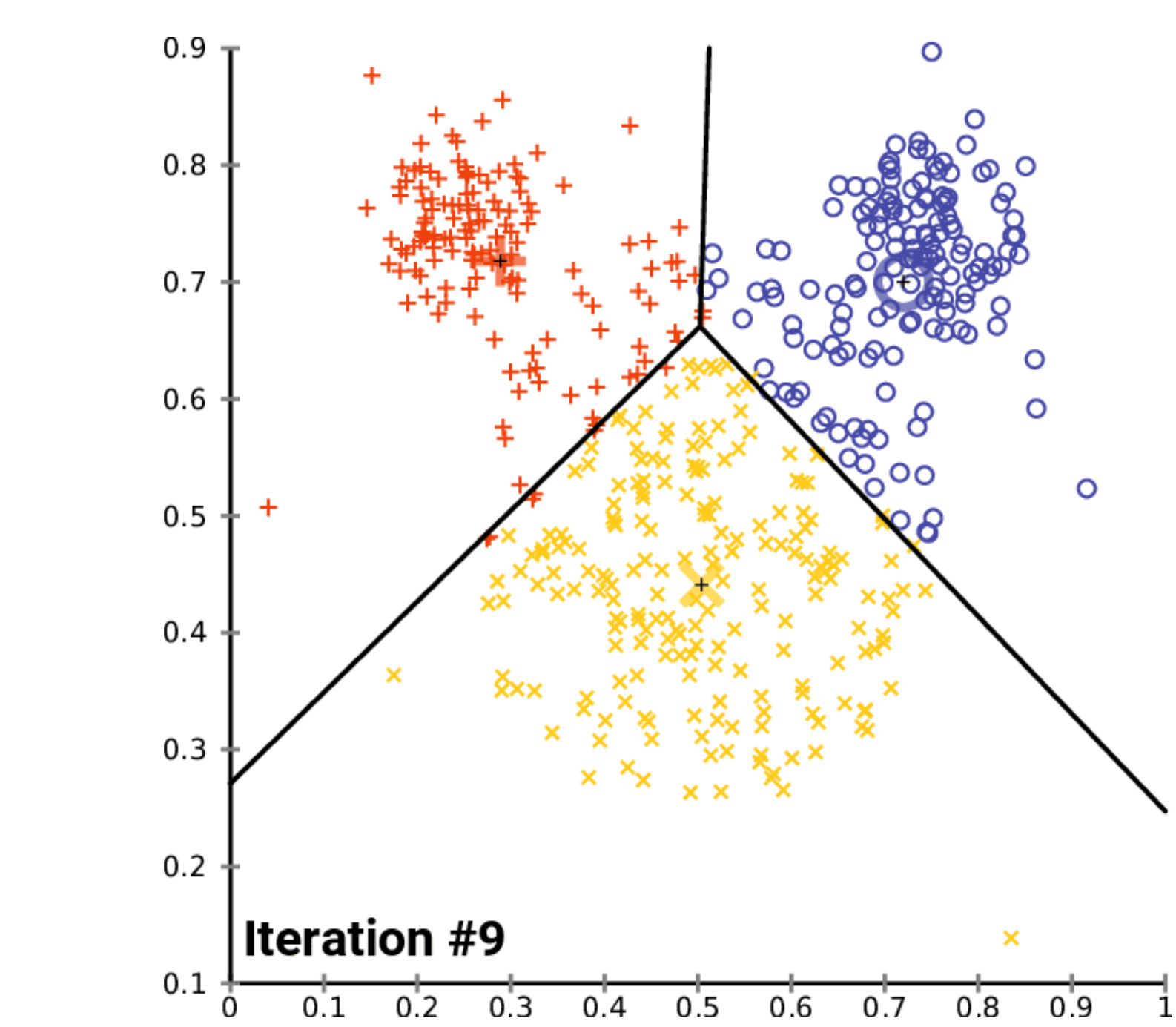
Out[53]:

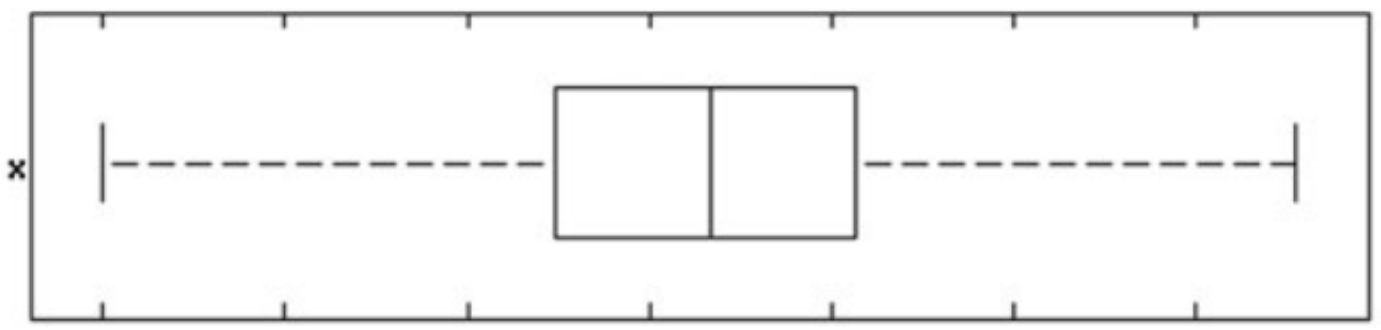
```
(1727, 4)
```

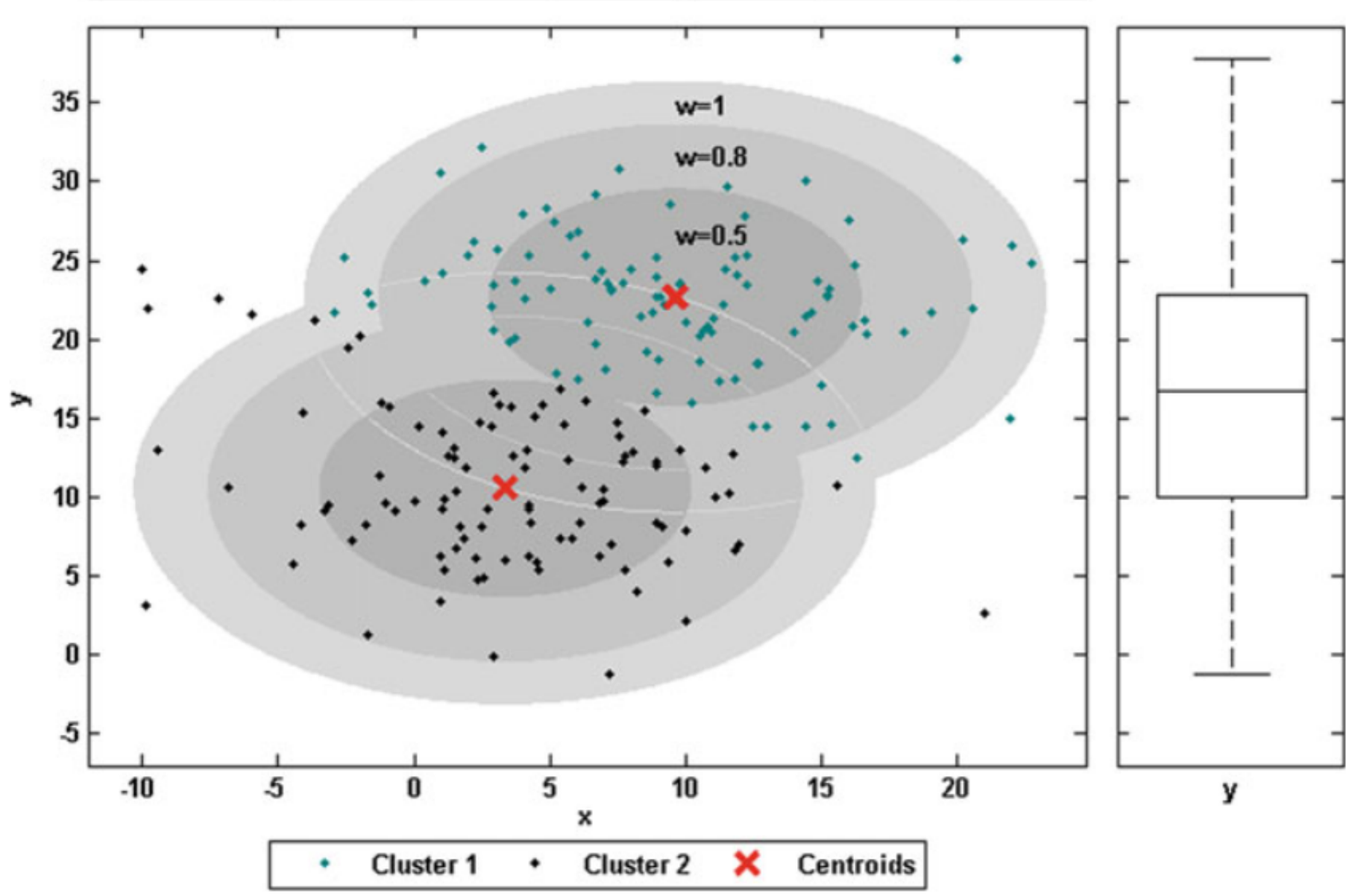
In [ ]:

```
# 请仿照上述Z评分，计算相应的"改进Z评分"
```

**聚类判别**






判别方式1：第一个标准基于每个点至聚类中心点的欧氏距离C来判断是否为离群值。

判别方式2：在这一标准中，我们计算每个数据点至聚类中心点(k-means)或者中位点(k-medoids)的距离。如果这个最近点到聚类中心的距离和这些计算后的距离之比小于某一特定阈值，则这个点被认为是离群值。

In [ ]:

```
from sklearn.cluster import k_means
```

### 专家经验的重要性

由于临床问题的复杂性和特殊性，我们需要借助临床专家的经验和专业领域知识帮助我们对离群值进行分析和处理，如下表所示。

| | Reference value | | | Analyzed data | | |
|---|---|---|---|---|---|---|
| Variable | Normal range | Critical | Impossible | IAC | Non-IAC | Units |
| Age | – | – | <17 (adults) | 15.2–99.1 | 15.2–97.5 | Years |
| SOFA | – | – | <0 and >24 | 1–17 | 0–14 | No units |
| WBC | 3 9–10.7 | ≥ 100 | <0 | 0.3–86.0 | 0 2–109.8 | $\times 10^9$ cells/L |
| Hemoglobin | Male: 13.5–17.5 | ≤ 6 and ≥ 20 | <0 | Male: 3 2–19.0 | 4.9–18.6 | g/dL |
| | Female: 12–16 | | | Female: 2.0–18.1 | 4.2–18.1 | |
| Platelets | 150–400 | ≤ 40 and ≥ 1000 | <0 | 7.0–680.0 | 9.0–988.0 | $\times 10^9$/L |
| Sodium | 136–145 | ≤ 120 and ≥ 160 | <0 | 105 0–165.0 | 111.0–154.0 | mmol/L |
| Potassium | 3.5–5 | ≤ 2.5 and ≥ 6 | <0 | 1 9–9.8 | 1.9–8.3 | mmol/L |
| $TCO_2$ | 22–28 | ≤ 10 and ≥ 40 [4] | <0 | 2.0–62.0 | 5.0–52.0 | mmol/L |
| Chloride [29] | 95–105 | ≤ 70 and ≥ 120 | <0 and ≥ 160 | 81.0–133.0 | 78.0–127.0 | mmol/L |
| BUN | 7–18 | ≥ 100 [1] | <0 | 2.0–139.0 | 2.0-126.0 | mg/dL |
| Creatinine | 0.6–1.2 | ≥ 10 | <0 | 0.2–12 5 | 0.0–18.3 | mg/dL |
| $PO_2$ | 75–105 | ≤ 40 | <0 | 25 0–594.0 | 22.0–634.0 | mmHg |
| $PCO_2$ | 33–45 | ≤ 20 and ≥ 70 | <0 | 8.0–141.0 | 14.0–158.0 | mmHg |

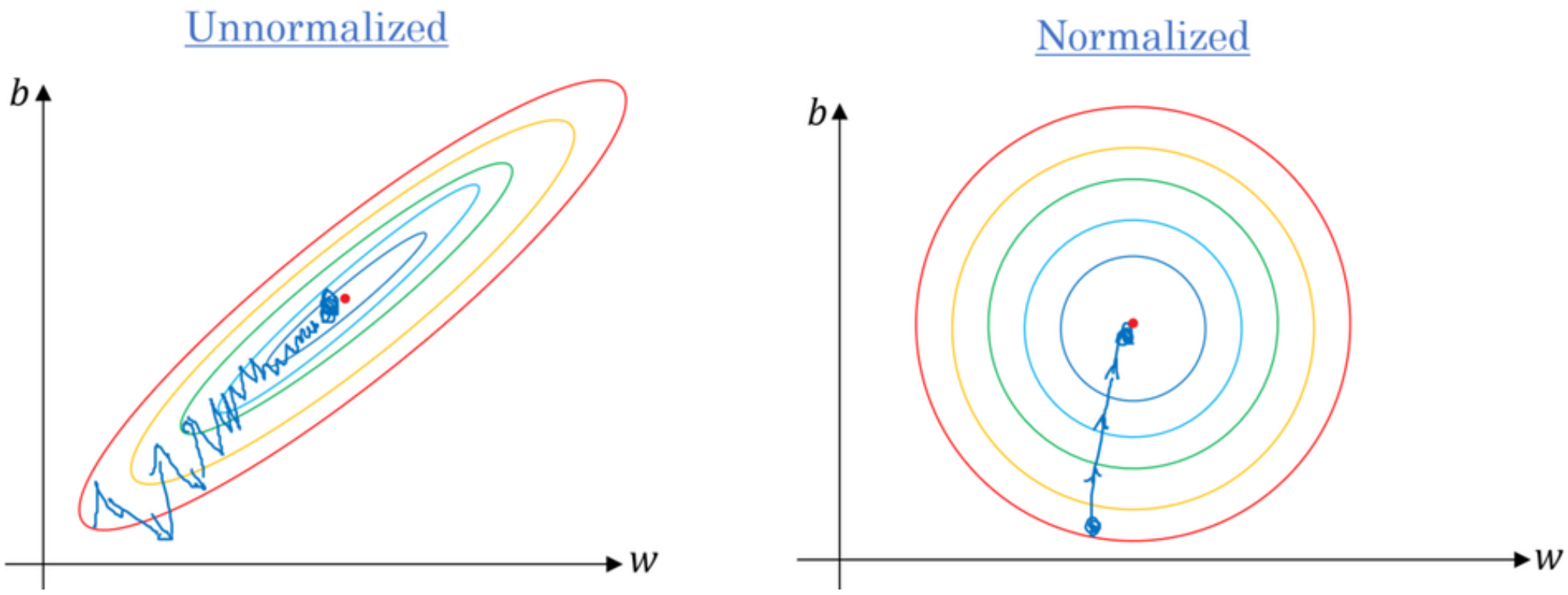
*这也是Datathon需要跨界团队配合的原因！*

## Step 3: 数据转化（Data Transformation）

数据转化的目的是将原始数据根据具体统计分析方法的需求，对原始数据值进行多种转换，将原始数据值转换为更适合分析的格式、大小或单位。

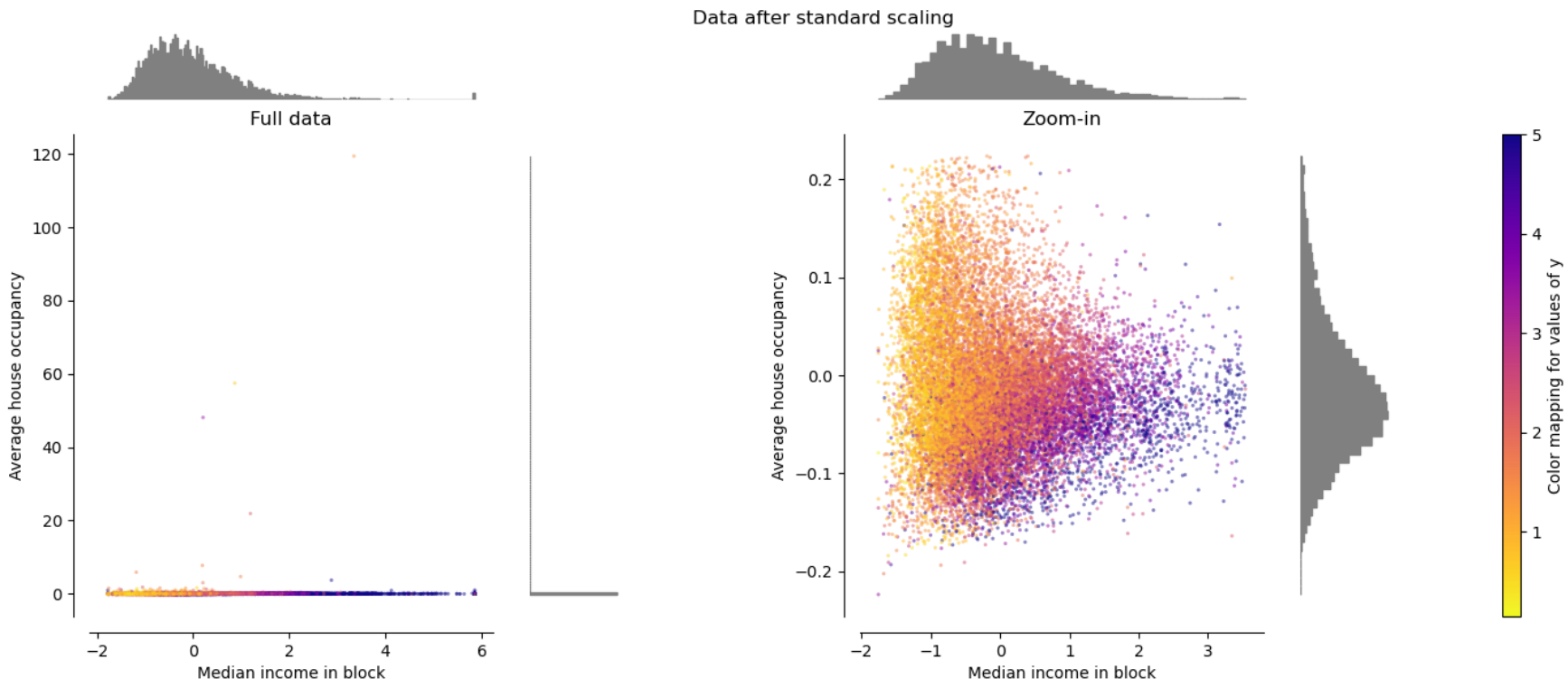
- 归一化：缩放数值变量的大小，使之符合指定的数值范围，例如将化验数值变化到0~1范围
- 聚合：将相同属性的两个或多个值聚合为一个值
- 泛化：将低级属性转化为高级属性

其中最为常见的就是归一化，对于某些模型，归一化还能够让模型加速训练速度。

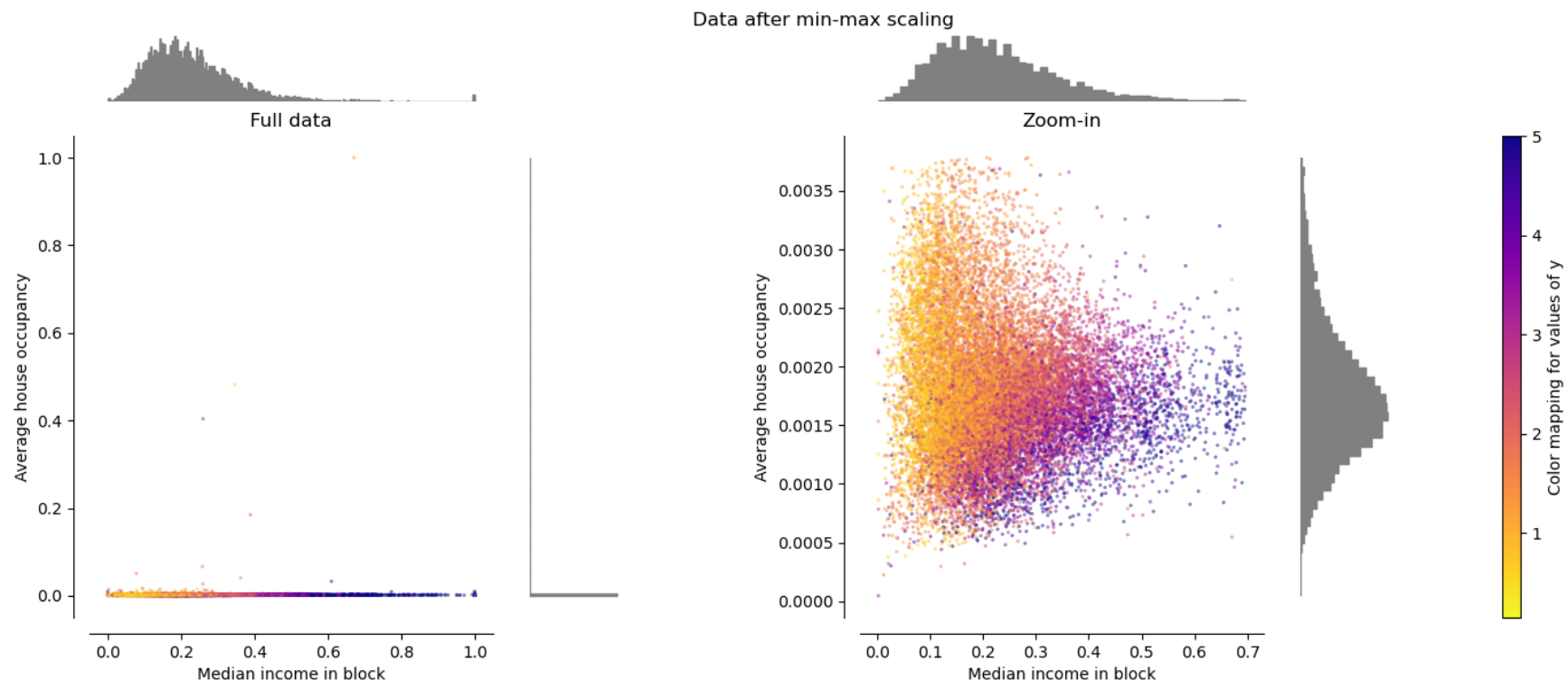



### 数值型变量（Numerical）

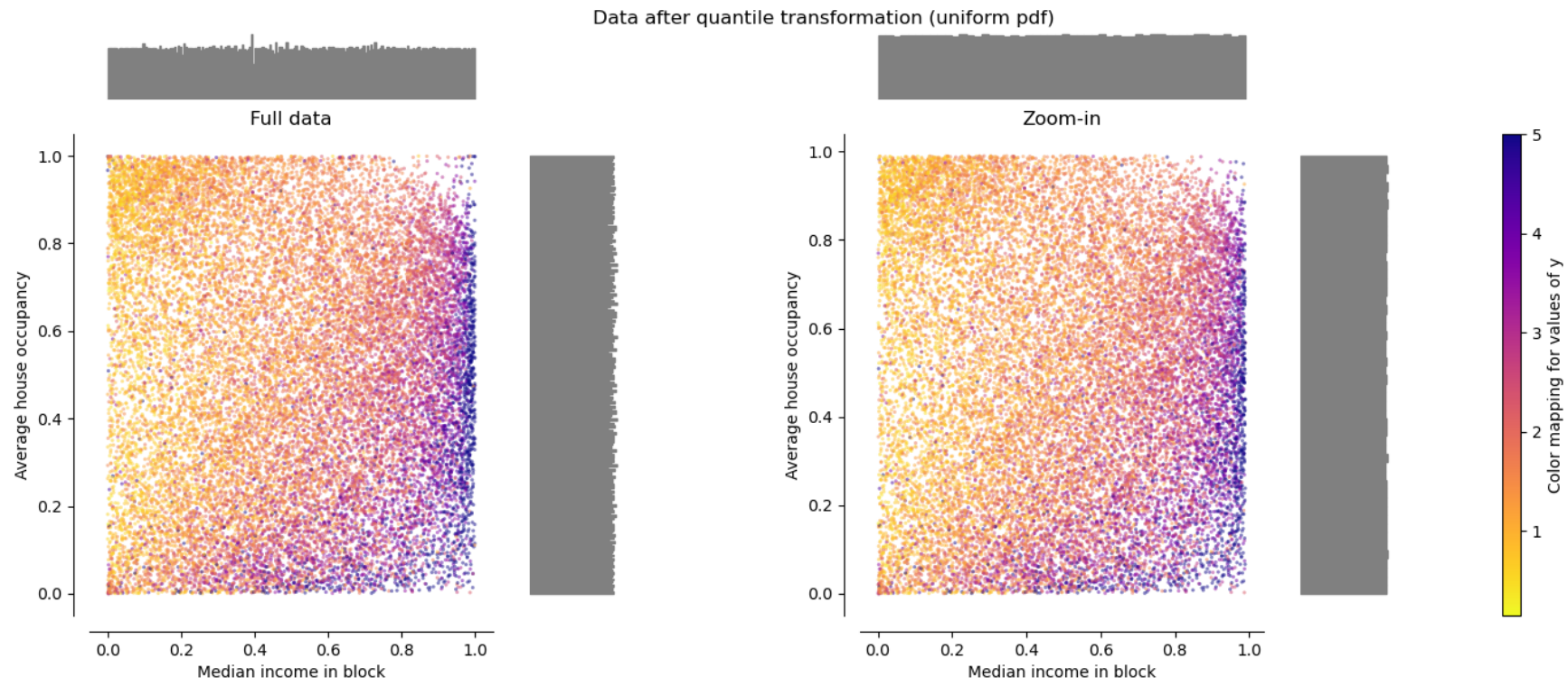
sklearn的`preprocessing`中包含了许多关于做归一化的方法，例如根据均值和标准差的归一化方法`StandardScaler`




根据最大最小值计算归一化方法的`MinMaxScaler`




还有一些非线性的变换方法，例如`PowerTransformer`和`QuantileTransformer`




例如下面的代码中，我们展示如何将`age`和`HR`使用标准方法进行转化。

In [54]:

```
from sklearn.preprocessing import StandardScaler
```

In [55]:

```
scaler = StandardScaler()
```

In [56]:

```
df_raw = iac_df[['wbc_first', 'hr_1st']]
```

In [57]:

```
df_norm = scaler.fit_transform(df_raw)
```

In [58]:

```
# 请比较正则化前后的分布变化
```

### 类别型变量（Categorical）

In [59]:

```
icu_type = iac_df['service_unit']
icu_type_dummied = pd.get_dummies(icu_type)
```

In [60]:

```
icu_type_dummied.head()
```

Out[60]:

| | FICU | MICU | SICU |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 0 | 1 | 0 |
| **2** | 0 | 1 | 0 |
| **3** | 0 | 0 | 1 |
| **4** | 0 | 0 | 1 |

### 4.3 ColumnTransformer

在一些机器学习建模场景里，我们需要对不同类型的列分别用不同的数据变换方法，例如同时进行上述两种变换。这时我们可以使用scikit-learn中的`ColumnTransformer`。

假设我们的数据集的暴露变量包括`service_unit`，`wbc_first`，我们需要分别处理。

In [ ]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
# 定义数值型的列
numerical_columns = ['wbc_first', 'hr_1st']
# 定义类别型的列
categorical_columns = ['service_unit']
```

In [ ]:

```
column_trans = ColumnTransformer(
    [('numerical', make_pipeline(SimpleImputer(), StandardScaler()), numerical_columns),
     ('categorical', OneHotEncoder(), categorical_columns)])
```

In [ ]:

```
ct_df = iac_df[['wbc_first', 'hr_1st', 'service_unit']]
```

In [ ]:

```
column_trans.fit_transform(ct_df)
```

## Step 4: 数据简化（Data Reduction）

对大型数据集进行复杂分析可能需要很长时间，甚至是不可行的。数据预处理的最后步骤是数据简化，即通过数据集的更有效表示来减少输入数据而不损害原始数据的完整性。此步骤的目的是提供一个使随后的统计分析更有效的数据集版本。数据简化可以是也可以不是无损的。也就是说，数据库的最终版本可以以更有效的格式（例如删除冗余记录）包含原始数据库的所有信息，或者可以保持数据完整性。但是当数据被变换一些信息会丢失，仅是呈现出新的形式（例如多个值表示为平均值）。

一个常见的MIMIC数据库实例是将ICD9代码合并成广泛的临床类别或感兴趣的变量并将患者按上述标准进行划分。对于感兴趣的研究领域（例如冠状动脉疾病史），通过将以文本格式存储的多个ICD9代码转变为一个二进制变量的单一记录，可以降低数据集大小。另一个例子是在分析中使用血压作为变量的情况。ICU患者通常通过动脉置管连续监测其收缩压和舒张压，或通过自动血压袖带每小时记录多次。这导致可能数千个研究患者中，每个患者都有数百个数据点。根据研究目的，有必要计算一个新的变量，例如ICU入院第一天的平均动脉压。

Pandas中的`resample`可以帮助完成简化任务。

## 参考资料


1. Introducing the ColumnTransformer: applying different transformations to different features in a scikit-learn pipeline