

# Project\_IAU

November 5, 2025

## 1 Import libraries

```
[152]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import scipy.stats as stats
from scipy.stats import shapiro, ttest_ind, mannwhitneyu, levene, pearsonr
from statsmodels.stats.power import TTestIndPower
from sklearn.model_selection import train_test_split
from pandas.api.types import CategoricalDtype
```

```
[153]: #warnings fix
warnings.filterwarnings("ignore", category=UserWarning, module="IPython")
plt.rcParams['font.family'] = 'DejaVu Sans' # Change the font globally
plt.tight_layout() # Ensure layout adjustments
```

<Figure size 640x480 with 0 Axes>

## 2 Load datasets

```
[154]: patient_df = pd.read_csv('132/patient.csv', sep = '\t')
station_df = pd.read_csv('132/station.csv', sep= '\t')
observation_df = pd.read_csv('132/observation.csv', sep = '\t')
```

```
[155]: patient_df.head()
```

```
company \
0
1 Gray, Cunningham and Morales
2
3
4 Munari s.r.l.

current_location          ssn \
0      (Decimal('32.168477'), Decimal('9.804478'))    302-73-9054
```

```

1      (Decimal('-80.289857'), Decimal('2.308813'))        499-92-6793
2      (Decimal('63.5169555'), Decimal('-48.876252'))       925-81-9055
3      (Decimal('-71.015803'), Decimal('140.978474'))       175-19-6965
4      (Decimal('-58.1525245'), Decimal('-120.099037'))    FRSMRL26C50L167Z

           name          mail  user_id residence \
0            takuma47@gmail.com     1770      NaN
1 Nicholas Campbell   curtis06@yahoo.com     946      NaN
2      Raissa Rose  junkgisbert@yahoo.de    2010      NaN
3            morimomoko@gmail.com    1100      NaN
4   Eraldo Anguillara bpergolesi@poste.it    1247      NaN

           username  birthdate \
0         snakajima  1937-03-07
1      benjamin02  1993-05-15
2        jkoehler      NaN
3      yamaguchikana  2000-08-20
4 guglielmomicheletti      NaN

           address      registration \
0          27 27  2020-08-15
1 0515 Angela Run\r\nPort Thomasberg, GU 35535  2018-11-30
2  Louise-Stey-Platz 79\r\nn88420 Bremervörde  05/09/2025, 00:00:00
3          26 5 4    985  10/21/2022, 00:00:00
4                               NaN  2024-11-26

  blood_group  station_ID
0        B-        464
1        A-        109
2        O-        462
3        B-        191
4        B+        588

```

[156]: station\_df.head()

```

[156]:      QoS code  latitude      revision  longitude  station
0      good  JP  36.00000  2020-11-25  139.55722  Okegawa
1 excellent  IN  11.93381  05/24/2021, 00:00:00  79.82979  Puducherry
2 maintenance  DE  52.21099  2022-05-10  7.02238  Gronau
3 excellent  CN  41.09822  2018-01-23  120.74792  Nanpiao
4      good  US  33.54428  08 Jun 2024 -84.23381  Stockbridge

```

[157]: observation\_df.head()

```

[157]:      SpO        HR        PI        RR        EtCO        FiO \
0  97.538229  87.194745  11.225419  14.812012  42.113735  33.852538
1  97.933271  80.787303  11.730935  14.964972  39.537692  65.326035

```

```

2 98.209983 79.733895 12.839449 14.840668 39.758706 53.925230
3 98.202790 86.156903 11.204152 14.523288 43.448577 35.227704
4 97.951933 78.966258 13.691758 14.992054 39.722280 35.005559

          PRV           BP   Skin Temperature  Motion/Activity index ... \
0 144.504405 100.455727            35.961920                  10.302567 ...
1 110.615787 102.133386            36.274352                  8.975704 ...
2 107.208040 104.036654            35.583851                  7.653790 ...
3 143.282224 105.723603            36.463180                  8.795732 ...
4 118.524021 98.996494            35.080937                  8.388887 ...

          CO  Blood Flow Index  PPG waveform features  Signal Quality Index \
0 4.022852            58.317397            49.143701            42.399816
1 4.002043            70.127865            15.557799            46.078137
2 4.001451            76.139163            53.879956            41.525607
3 4.015162            49.461570            58.701159            36.535021
4 4.001110            47.065823            52.338305            29.506008

  Respiratory effort  O extraction ratio        SNR  oximetry  latitude \
0             46.497869            0.289012  39.334620      1.0  49.183239
1             53.351208            0.290879  26.006709      0.0  33.544280
2             52.124182            0.263171  31.890829      1.0 -27.505780
3             50.342830            0.256780  30.721375      1.0  37.656390
4             39.480811            0.276094  38.214856      0.0  51.202190

  longitude
0 15.454273
1 -84.233810
2 153.102360
3 126.835000
4 7.360270

```

[5 rows x 23 columns]

lets have a look at the attributes

[158]: `observation_df.columns`

```

[158]: Index(['SpO ', 'HR', 'PI', 'RR', 'EtCO ', 'FiO ', 'PRV', 'BP',
       'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
       'CO', 'Blood Flow Index', 'PPG waveform features',
       'Signal Quality Index', 'Respiratory effort', 'O extraction ratio',
       'SNR', 'oximetry', 'latitude', 'longitude'],
      dtype='object')

```

[159]: `station_df.columns`

```
[159]: Index(['QoS', 'code', 'latitude', 'revision', 'longitude', 'station'],  
           dtype='object')
```

```
[160]: patient_df.columns
```

```
[160]: Index(['company', 'current_location', 'ssn', 'name', 'mail', 'user_id',  
             'residence', 'username', 'birthdate', 'address', 'registration',  
             'blood_group', 'station_ID'],  
            dtype='object')
```

```
[161]: for column in observation_df.columns:  
        if column in station_df.columns:  
            print ('observation_df and station_df share column : ' + str(column) )  
        if column in patient_df.columns:  
            print ('observation_df and patient_df share column : ' + str(column) )  
  
for column in patient_df.columns:  
    if column in station_df.columns:  
        print ('station_df and patient_df share column : ' + str(column) )
```

```
observation_df and station_df share column :latitude  
observation_df and station_df share column :longitude
```

we can see that observation and station dataframes might be joinable via coordinates but after manual revision we can see that patient.station\_ID will probably map to station ids

before joining the datasets, lets perform some basic EDA

### 3 Fáza 1

#### 3.1 1.1 Základný opis dát spolu s ich charakteristikami

##### 3.1.1 A) Analyza struktur dat ako subory + zaznamy

station\_df

```
[162]: station_df.head()
```

```
[162]:      QoS  code  latitude          revision  longitude  station  
0     good    JP   36.00000  2020-11-25  139.55722  Okegawa  
1  excellent   IN  11.93381  05/24/2021, 00:00:00  79.82979  Puducherry  
2  maintenance   DE  52.21099  2022-05-10    7.02238  Gronau  
3  excellent   CN  41.09822  2018-01-23  120.74792  Nanpiao  
4     good    US  33.54428       08 Jun 2024 -84.23381 Stockbridge
```

```
[163]: station_df.shape
```

```
[163]: (703, 6)
```

```
[164]: station_df.columns
```

```
[164]: Index(['QoS', 'code', 'latitude', 'revision', 'longitude', 'station'],  
           dtype='object')
```

```
[165]: station_df.dtypes
```

```
[165]: QoS          object  
       code         object  
      latitude    float64  
     revision     object  
    longitude    float64  
   station      object  
dtype: object
```

```
[166]: #2 nas, not bad  
station_df.isna().sum()
```

```
[166]: QoS      0  
       code     2  
      latitude  0  
     revision  0  
    longitude  0  
   station    0  
dtype: int64
```

```
[167]: station_df[station_df['code'].isna()]
```

```
[167]:      QoS  code  latitude      revision  longitude  station  
274  maintenance  NaN -21.98333  02/23/2016, 00:00:00  16.91667 Okahandja  
318        good  NaN -21.98333                06 Dec 2021  16.91667 Okahandja
```

```
[168]: #by there is valid code for Okahandja?  
station_df[station_df['station'] == 'Okahandja']  
#there is not :(
```

```
[168]:      QoS  code  latitude      revision  longitude  station  
274  maintenance  NaN -21.98333  02/23/2016, 00:00:00  16.91667 Okahandja  
318        good  NaN -21.98333                06 Dec 2021  16.91667 Okahandja
```

```
[169]: station_df.nunique()
```

```
[169]: QoS      4  
       code     98  
      latitude 498  
     revision  683  
    longitude  497  
   station    498  
dtype: int64
```

```
[170]: #prob useless
station_df.describe()
```

```
[170]:      latitude  longitude
count    703.000000  703.000000
mean     28.699220   16.946846
std      24.406067   70.122555
min     -44.396720  -156.474320
25%      14.354040  -14.410810
50%      36.650000   13.321270
75%      47.432685   71.552920
max     65.848110   171.253640
```

patient df

```
[171]: patient_df.head()
```

```
[171]:          company \
0
1 Gray, Cunningham and Morales
2                               Walter
3
4       Munari s.r.l.

          current_location           ssn \
0      (Decimal('32.168477'), Decimal('9.804478')) 302-73-9054
1      (Decimal('-80.289857'), Decimal('2.308813')) 499-92-6793
2      (Decimal('63.5169555'), Decimal('-48.876252')) 925-81-9055
3      (Decimal('-71.015803'), Decimal('140.978474')) 175-19-6965
4  (Decimal('-58.1525245'), Decimal('-120.099037')) FRSMRL26C50L167Z

          name            mail  user_id residence \
0           takuma47@gmail.com     1770      NaN
1 Nicholas Campbell  curtis06@yahoo.com      946      NaN
2        Raissa Rose  junkgisbert@yahoo.de     2010      NaN
3           morimomoko@gmail.com     1100      NaN
4 Eraldo Anguillara  bpergolesi@poste.it     1247      NaN

          username  birthdate \
0      snakajima  1937-03-07
1  benjamin02  1993-05-15
2      jkoehler        NaN
3  yamaguchikana  2000-08-20
4 guglielmomicheletti        NaN

          address           registration \
0          27 27  2020-08-15
1  0515 Angela Run\r\nPort Thomasberg, GU 35535  2018-11-30
```

```
2      Louise-Stey-Platz 79\r\n88420 Bremervörde 05/09/2025, 00:00:00
3                      26 5 4      985 10/21/2022, 00:00:00
4                                         NaN          2024-11-26
```

```
blood_group  station_ID
0           B-        464
1           A-        109
2           O-        462
3           B-        191
4           B+        588
```

```
[172]: patient_df.shape
```

```
[172]: (2197, 13)
```

```
[173]: patient_df.columns
```

```
[173]: Index(['company', 'current_location', 'ssn', 'name', 'mail', 'user_id',
       'residence', 'username', 'birthdate', 'address', 'registration',
       'blood_group', 'station_ID'],
       dtype='object')
```

```
[174]: patient_df.dtypes
```

```
company          object
current_location    object
ssn              object
name              object
mail              object
user_id          int64
residence         float64
username          object
birthdate         object
address            object
registration       object
blood_group        object
station_ID         int64
dtype: object
```

```
[175]: patient_df.isna().sum()
#residence is full NaN so it will be dropped, birthdate and address probably
    ↵too,
#current location needs to be processed
```

```
company          0
current_location 110
ssn              0
name              0
```

```
mail          0
user_id       0
residence    2197
username      0
birthdate     989
address       330
registration   0
blood_group   0
station_ID    0
dtype: int64
```

```
[176]: patient_df.nunique()
#many unique values
```

```
[176]: company        1989
current_location 2087
ssn              2197
name             2131
mail             2192
user_id          1407
residence        0
username         2175
birthdate        1189
address          1867
registration     1987
blood_group      8
station_ID       677
dtype: int64
```

```
[177]: patient_df.describe()
#also useless
```

```
[177]:      user_id  residence  station_ID
count  2197.000000      0.0  2197.000000
mean   1105.308603      NaN  348.892126
std    644.728055      NaN  205.367454
min    0.000000      NaN  0.000000
25%   547.000000      NaN  172.000000
50%   1118.000000      NaN  340.000000
75%   1668.000000      NaN  536.000000
max   2196.000000      NaN  702.000000
```

observation df

```
[178]: observation_df.head()
```

```
[178]:      SpO        HR        PI        RR        EtCO      FiO  \
0  97.538229  87.194745  11.225419  14.812012  42.113735  33.852538
```

```

1 97.933271 80.787303 11.730935 14.964972 39.537692 65.326035
2 98.209983 79.733895 12.839449 14.840668 39.758706 53.925230
3 98.202790 86.156903 11.204152 14.523288 43.448577 35.227704
4 97.951933 78.966258 13.691758 14.992054 39.722280 35.005559

          PRV          BP Skin Temperature Motion/Activity index ... \
0 144.504405 100.455727           35.961920           10.302567 ...
1 110.615787 102.133386           36.274352           8.975704 ...
2 107.208040 104.036654           35.583851           7.653790 ...
3 143.282224 105.723603           36.463180           8.795732 ...
4 118.524021 98.996494           35.080937           8.388887 ...

          CO Blood Flow Index PPG waveform features Signal Quality Index \
0 4.022852           58.317397           49.143701           42.399816
1 4.002043           70.127865           15.557799           46.078137
2 4.001451           76.139163           53.879956           41.525607
3 4.015162           49.461570           58.701159           36.535021
4 4.001110           47.065823           52.338305           29.506008

Respiratory effort O extraction ratio SNR oximetry latitude \
0           46.497869           0.289012 39.334620      1.0 49.183239
1           53.351208           0.290879 26.006709      0.0 33.544280
2           52.124182           0.263171 31.890829      1.0 -27.505780
3           50.342830           0.256780 30.721375      1.0 37.656390
4           39.480811           0.276094 38.214856      0.0 51.202190

longitude
0 15.454273
1 -84.233810
2 153.102360
3 126.835000
4 7.360270

```

[5 rows x 23 columns]

[179]: `observation_df.shape`

[179]: (12177, 23)

[180]: `observation_df.columns`

[180]: `Index(['SpO ', 'HR', 'PI', 'RR', 'EtCO ', 'FiO ', 'PRV', 'BP',
 'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
 'CO', 'Blood Flow Index', 'PPG waveform features',
 'Signal Quality Index', 'Respiratory effort', 'O extraction ratio',
 'SNR', 'oximetry', 'latitude', 'longitude'],
 dtype='object')`

```
[181]: observation_df.dtypes
```

```
[181]: SpO          float64  
HR           float64  
PI           float64  
RR           float64  
EtCO          float64  
FiO           float64  
PRV           float64  
BP           float64  
Skin Temperature   float64  
Motion/Activity index float64  
PVI           float64  
Hb level       float64  
SV            float64  
CO            float64  
Blood Flow Index float64  
PPG waveform features float64  
Signal Quality Index float64  
Respiratory effort float64  
O extraction ratio float64  
SNR            float64  
oximetry        float64  
latitude         float64  
longitude        float64  
dtype: object
```

```
[182]: observation_df.isna().sum()  
#great
```

```
[182]: SpO          0  
HR           0  
PI           0  
RR           0  
EtCO          0  
FiO           0  
PRV           0  
BP            0  
Skin Temperature   0  
Motion/Activity index 0  
PVI           0  
Hb level       0  
SV            0  
CO            0  
Blood Flow Index 0  
PPG waveform features 0  
Signal Quality Index 0
```

```
Respiratory effort      0
O extraction ratio     0
SNR                   0
oximetry               0
latitude               0
longitude              0
dtype: int64
```

```
[183]: observation_df.nunique()
#makes sense, only oximetry, latitude and longitude are not completely unique
```

```
[183]: Sp0                 11997
HR                  11997
PI                  11997
RR                  11997
EtCO                11997
FiO                  11997
PRV                 11997
BP                  11997
Skin Temperature     11997
Motion/Activity index 11997
PVI                 11997
Hb level              11997
SV                  11997
CO                  11997
Blood Flow Index     11997
PPG waveform features 11997
Signal Quality Index 11997
Respiratory effort     11997
O extraction ratio     11997
SNR                  11997
oximetry              2
latitude              498
longitude             497
dtype: int64
```

```
[184]: observation_df.describe()
```

```
[184]:          Sp0            HR            PI            RR            EtCO   \
count  12177.000000  12177.000000  12177.000000  12177.000000  12177.000000
mean    97.336001   83.397242   10.360642   16.158948   40.235152
std     0.657577   7.609815   2.417855   1.398210   1.715679
min     95.000000  60.000000  0.200000  12.000000  35.000000
25%    96.895657  77.573676  8.800161  15.044708  38.749846
50%    97.332851  84.323757 10.365288  15.979021  40.531056
75%    97.776356  89.763115 11.893853  17.393451  41.590048
max    100.000000 100.000000 20.000000  20.000000  45.000000
```

	FiO	PRV	BP	Skin Temperature	\
count	12177.000000	12177.000000	12177.000000	12177.000000	
mean	58.821759	117.675964	104.591413	35.426048	
std	12.119443	21.841513	4.088282	0.619283	
min	21.000000	20.000000	90.000000	33.000000	
25%	49.328258	103.101631	101.857377	35.007462	
50%	59.402365	117.695370	104.604980	35.424762	
75%	68.437184	132.088845	107.321633	35.845164	
max	100.000000	200.000000	120.000000	38.000000	
	Motion/Activity index	...	CO	Blood Flow Index	\
count	12177.000000	...	12177.000000	12177.000000	
mean	9.436818	...	4.078126	52.640590	
std	0.998907	...	0.222547	13.127091	
min	5.652322	...	4.000000	0.000000	
25%	8.766904	...	4.000762	43.996467	
50%	9.432476	...	4.007395	52.709829	
75%	10.105665	...	4.064212	61.440013	
max	13.997052	...	8.000000	100.000000	
	PPG waveform features	Signal Quality Index	Respiratory effort	\	
count	12177.000000	12177.000000	12177.000000		
mean	46.734247	47.572094	49.788653		
std	13.374194	13.487056	13.006681		
min	0.000000	0.000000	0.000000		
25%	37.650597	38.396055	40.992562		
50%	46.733166	47.810036	49.742898		
75%	55.671672	56.736043	58.630016		
max	100.000000	100.000000	100.000000		
	O extraction ratio	SNR	oximetry	latitude	\
count	12177.000000	12177.000000	12177.000000	12177.000000	
mean	0.249557	29.994576	0.597602	28.701943	
std	0.028947	5.765251	0.490401	24.402668	
min	0.200000	20.000000	0.000000	-44.396720	
25%	0.224466	24.978691	0.000000	14.082300	
50%	0.249072	30.094799	1.000000	36.650000	
75%	0.274733	34.961220	1.000000	47.484440	
max	0.300000	40.000000	1.000000	65.848110	
	longitude				
count	12177.000000				
mean	17.028640				
std	70.059129				
min	-156.474320				
25%	-13.235600				

```
50%      13.321270  
75%      71.577370  
max      171.253640
```

[8 rows x 23 columns]

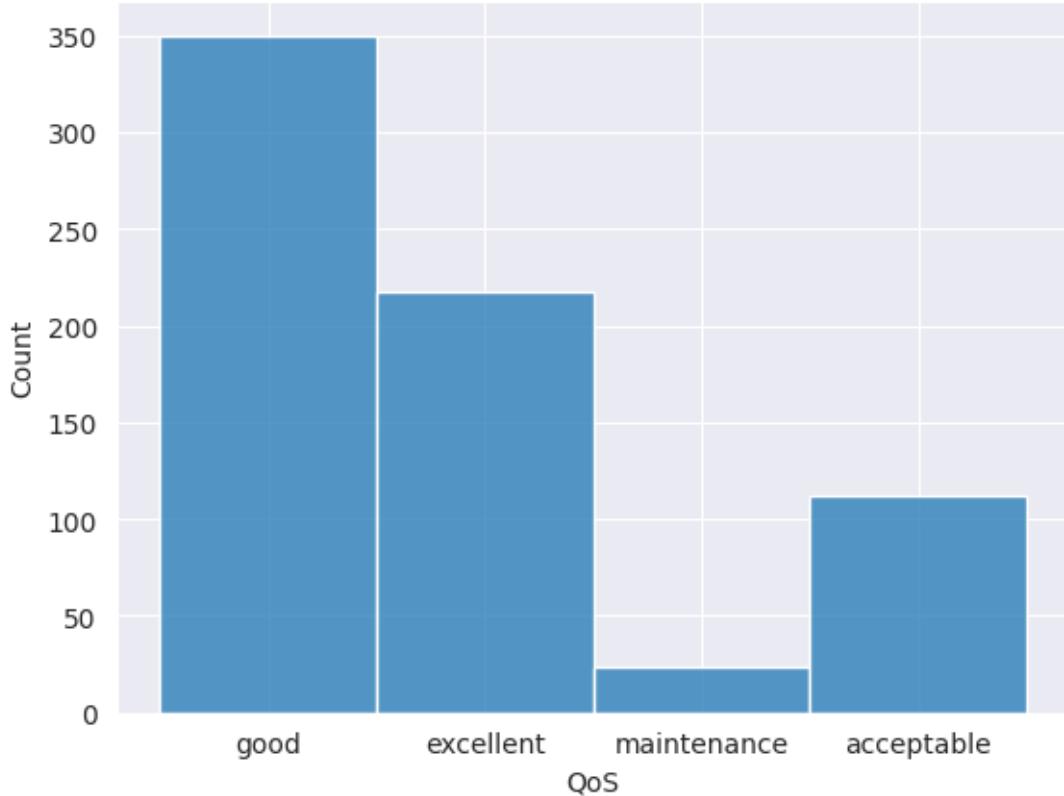
### 3.1.2 B) Analýza jednotlivých atribútov

```
[185]: #Qos(station_df)  
station_df['QoS'].unique().tolist()
```

```
[185]: ['good', 'excellent', 'maintenance', 'acceptable']
```

```
[186]: sns.histplot(station_df['QoS'])
```

```
[186]: <Axes: xlabel='QoS', ylabel='Count'>
```



```
[187]: #for exact nums  
station_df['QoS'].value_counts()
```

```
[187]: QoS
good           350
excellent     217
acceptable    112
maintenance   24
Name: count, dtype: int64
```

```
[188]: station_df['QoS'].isna().sum()
```

```
[188]: np.int64(0)
```

```
[189]: #latitude (from both station_df and observation_df)
print (station_df['latitude'].min(), observation_df['latitude'].min())
print (station_df['latitude'].max(), observation_df['latitude'].max())
#the min and the max are both realistic values, from -180 to 180

set(station_df['latitude']) == set(observation_df['latitude'])
#every station is included in observation by latitude
```

```
-44.39672 -44.39672
65.84811 65.84811
```

```
[189]: True
```

```
[190]: #longitude (from both station_df and observation_df)
print (station_df['longitude'].min(), observation_df['longitude'].min())
print (station_df['longitude'].max(), observation_df['longitude'].max())
#the min and the max are both realistic values, from -180 to 180

set(station_df['longitude']) == set(observation_df['longitude'])
#every station is included by longitude also

#this could be checked with patient -> current location but that needs
#preprocessing in further steps
```

```
-156.47432 -156.47432
171.25364 171.25364
```

```
[190]: True
```

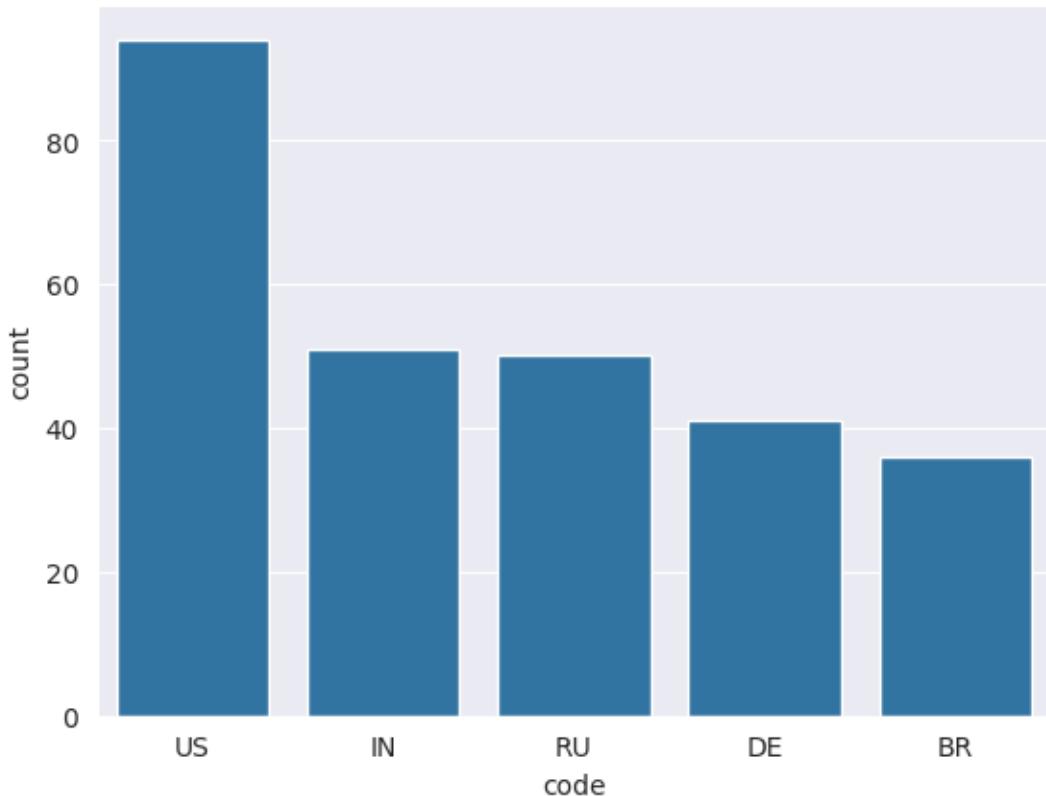
```
[191]: #Code (station_df)
station_df['code'].value_counts()
```

```
[191]: code
US      94
IN      51
RU      50
DE      41
BR      36
```

```
..  
DK      1  
CU      1  
GH      1  
AF      1  
AT      1  
Name: count, Length: 98, dtype: int64
```

```
[192]: #top 5  
top5 = station_df['code'].value_counts().head(5).reset_index()  
top5.columns = ['code', 'count']  
  
sns.barplot(x='code', y='count', data=top5)
```

```
[192]: <Axes: xlabel='code', ylabel='count'>
```

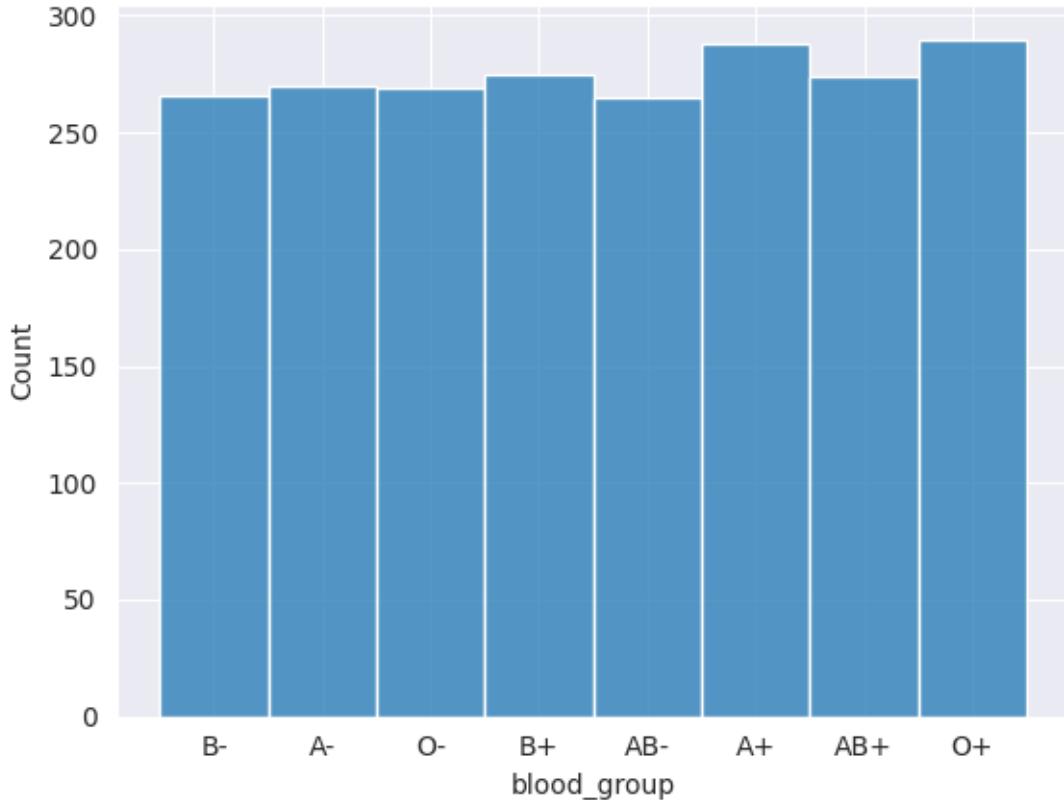


```
[193]: #Blood group (patient_df)  
patient_df['blood_group'].unique().tolist()  
#All blood groups are represented
```

```
[193]: ['B-', 'A-', 'O-', 'B+', 'AB-', 'A+', 'AB+', 'O+']
```

```
[194]: sns.histplot(patient_df['blood_group'])
#not that big of a range
```

```
[194]: <Axes: xlabel='blood_group', ylabel='Count'>
```



```
[195]: patient_df['blood_group'].value_counts()
```

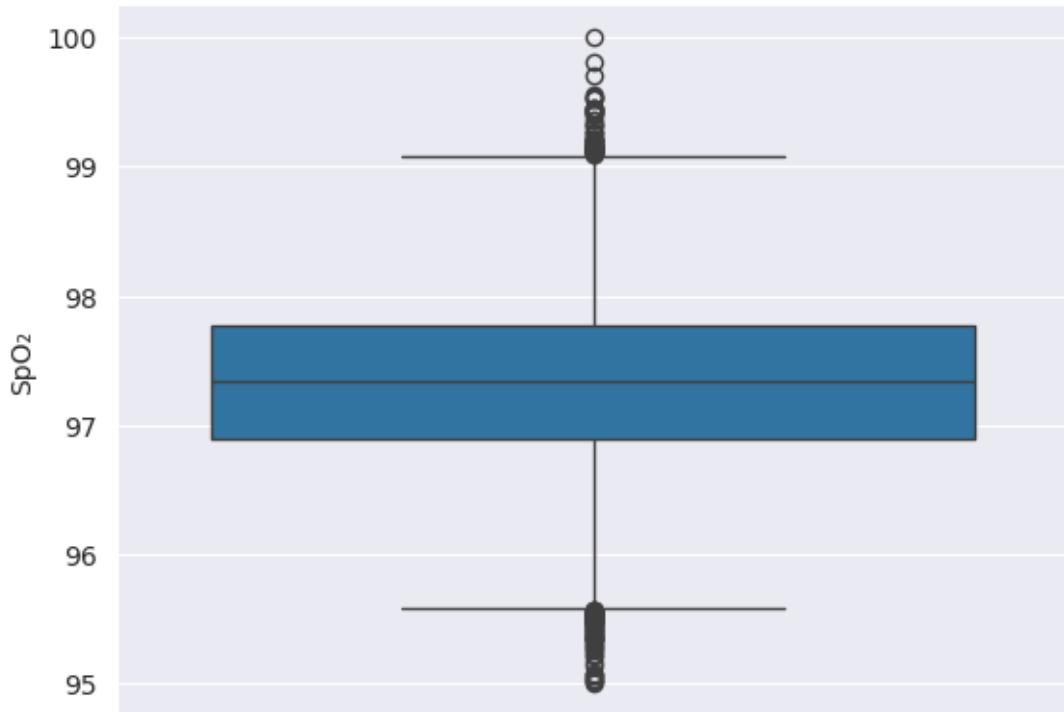
```
[195]: blood_group
0+    290
A+    288
B+    275
AB+   274
A-    270
O-    269
B-    266
AB-   265
Name: count, dtype: int64
```

```
[196]: patient_df['blood_group'].isna().sum()
#great
```

```
[196]: np.int64(0)
```

```
[197]: # SpO2  
sns.boxplot(y = observation_df['SpO'])
```

```
[197]: <Axes: ylabel='SpO'>
```

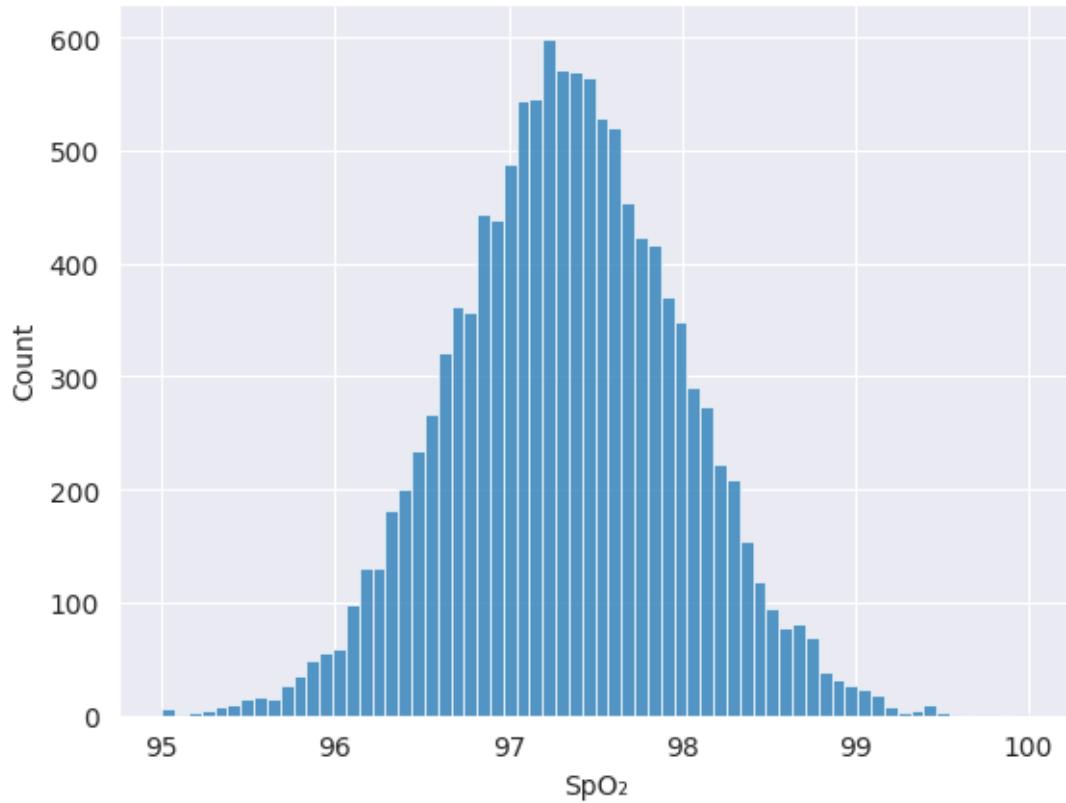


```
[198]: #just to have the exact range  
observation_df['SpO'].min(), observation_df['SpO'].max()  
#realistic values
```

```
[198]: (np.float64(95.0), np.float64(100.0))
```

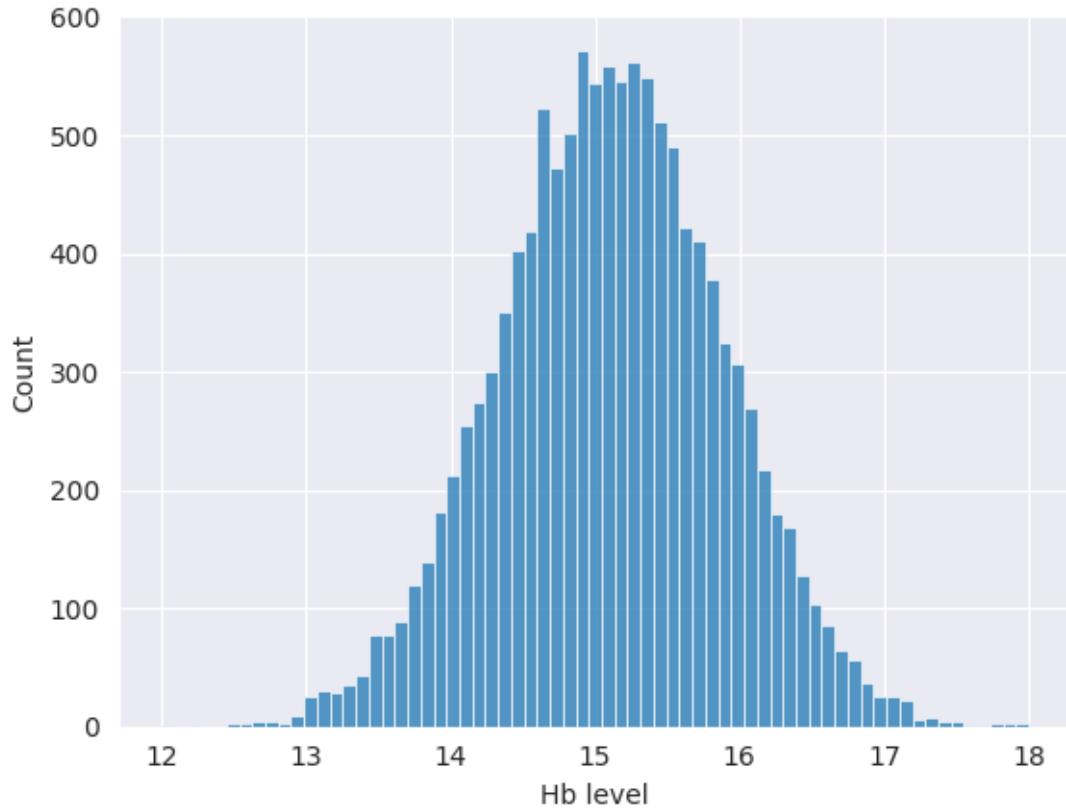
```
[199]: sns.histplot(observation_df['SpO'])  
#normal distribution
```

```
[199]: <Axes: xlabel='SpO', ylabel='Count'>
```



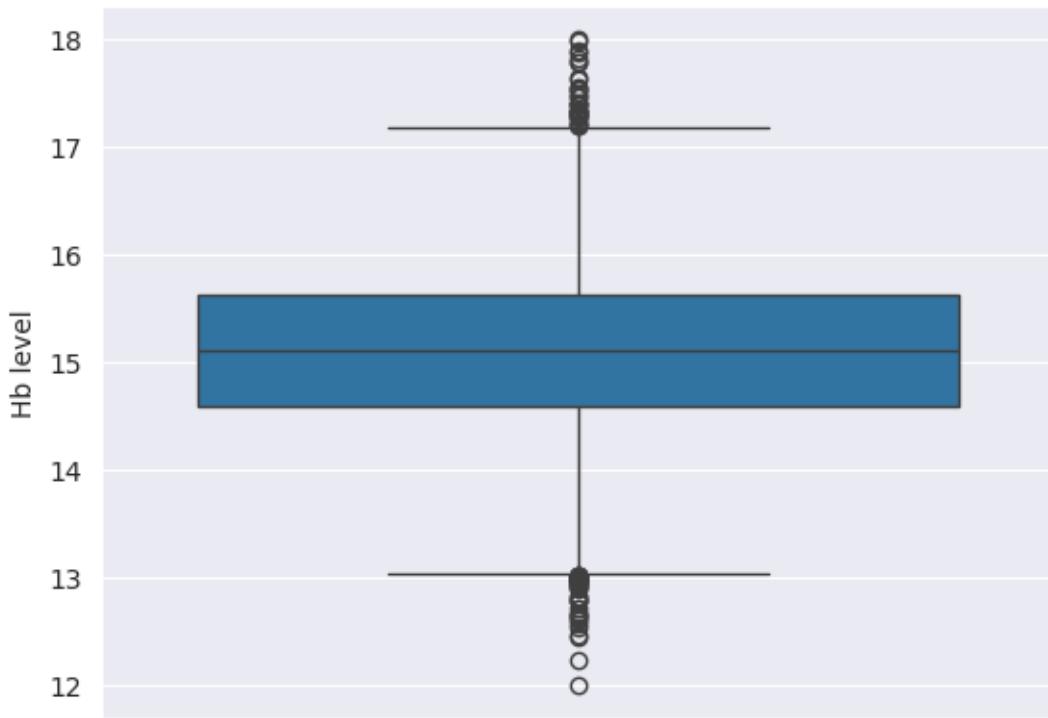
```
[200]: #Hemoglobin (Observation_df)
sns.histplot(observation_df['Hb level'])
#pretty normal, realistic values
```

```
[200]: <Axes: xlabel='Hb level', ylabel='Count'>
```



```
[201]: sns.boxplot(y = observation_df['Hb level'])
```

```
[201]: <Axes: ylabel='Hb level'>
```

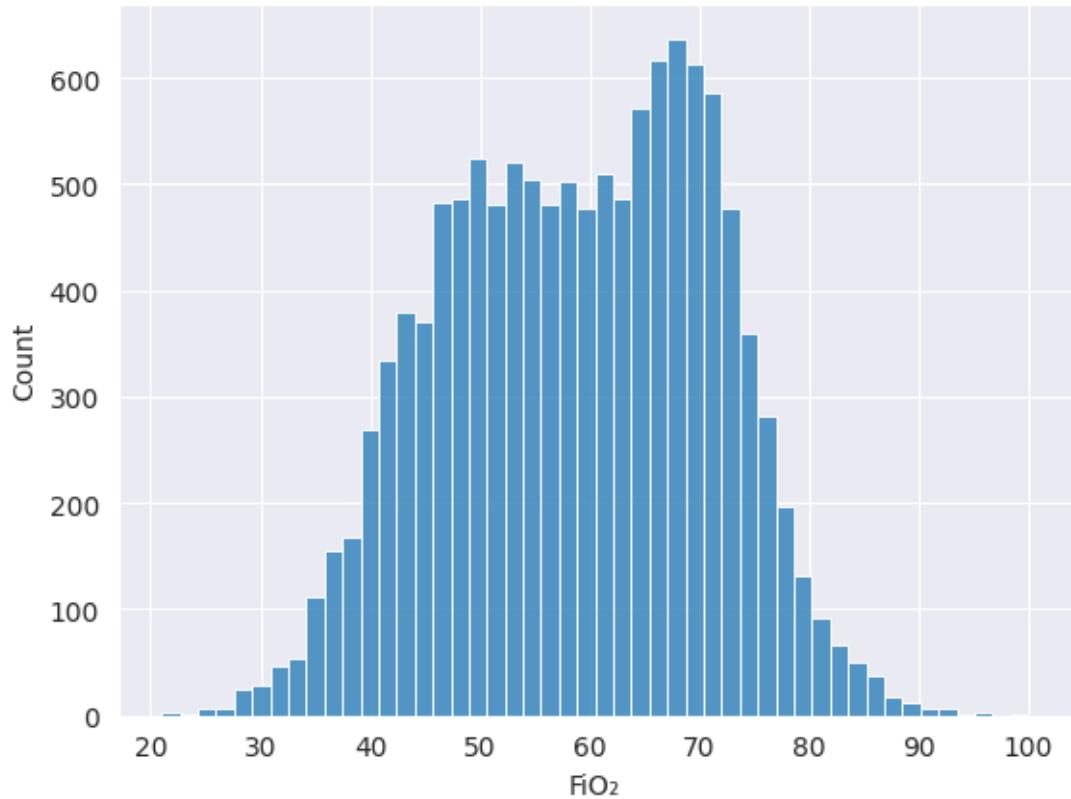


```
[202]: #exact range  
observation_df['Hb level'].min(), observation_df['Hb level'].max()
```

```
[202]: (np.float64(12.0), np.float64(18.0))
```

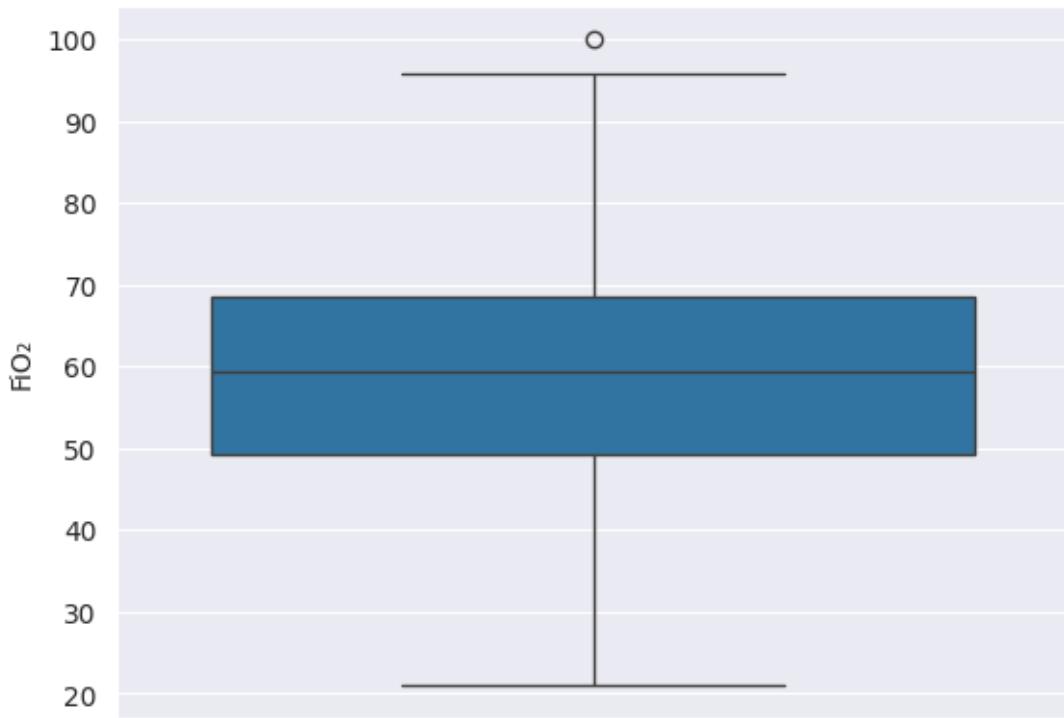
```
[203]: #Fio (Observation_df)  
sns.histplot(observation_df['FiO'])  
#not that normal, slight skew to the right at the peak  
#also values are not realistic for common people, probably on oxygen therapy or  
↳ something
```

```
[203]: <Axes: xlabel='FiO ', ylabel='Count'>
```



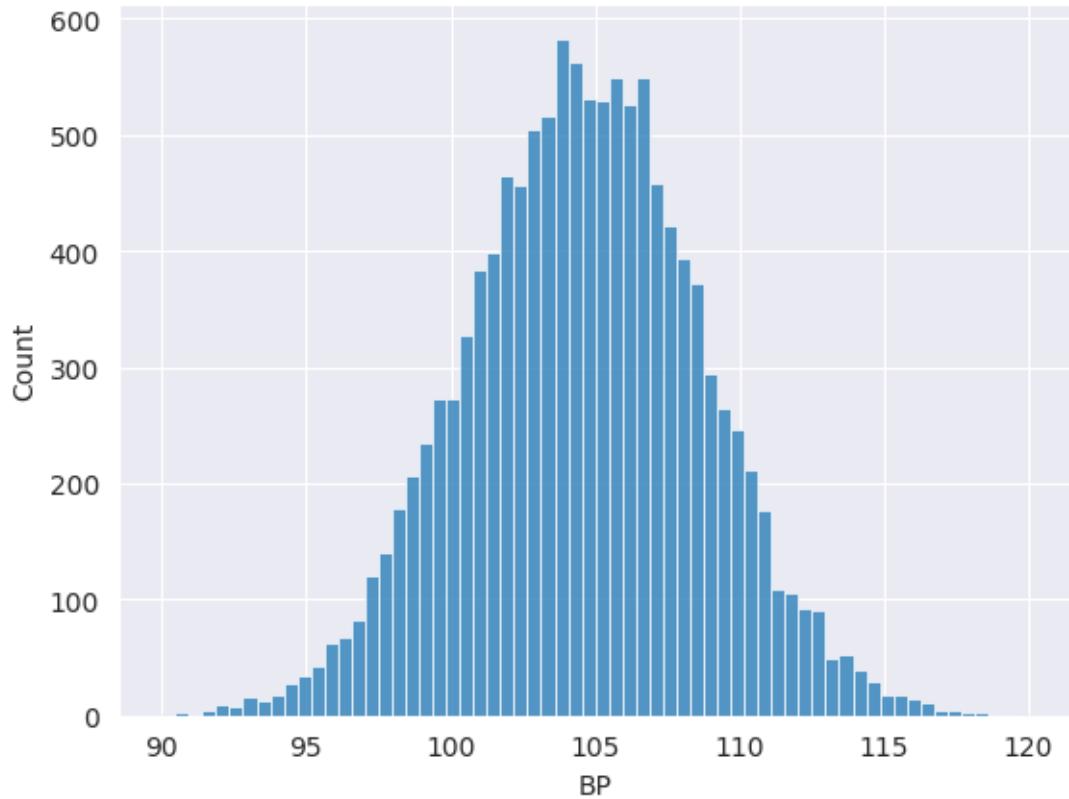
```
[204]: sns.boxplot(observation_df['FiO'])
```

```
[204]: <Axes: ylabel='FiO '>
```



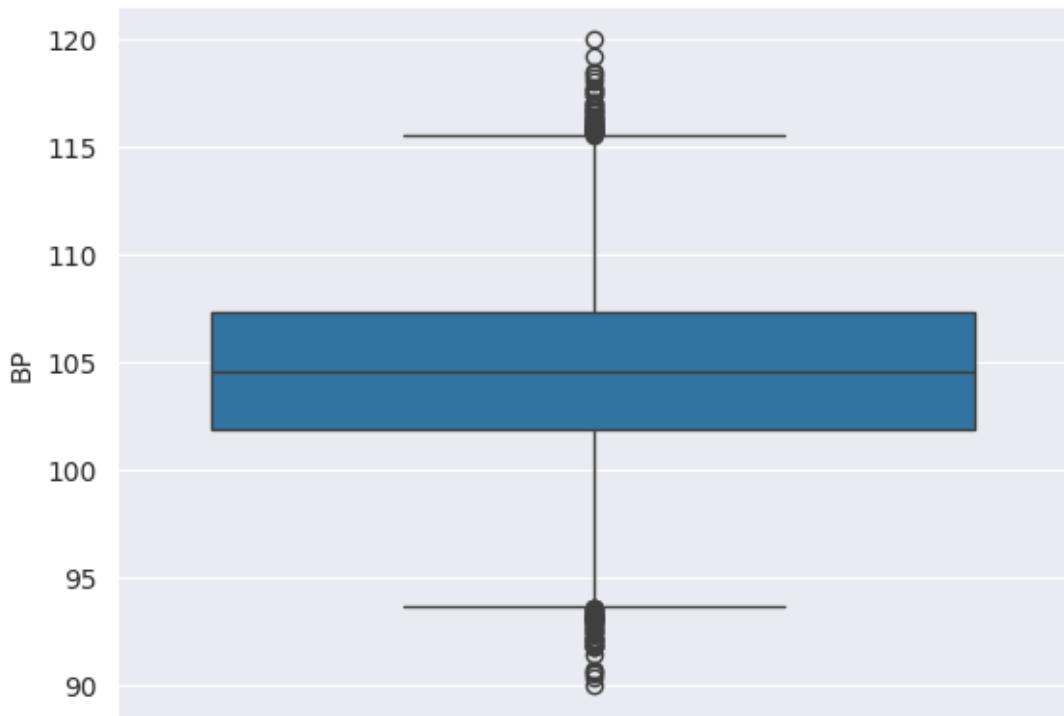
```
[205]: #blood pressure (observation_df)
sns.histplot(observation_df['BP'])
#normal distribution with usual values
```

```
[205]: <Axes: xlabel='BP', ylabel='Count'>
```



```
[206]: sns.boxplot(y = observation_df['BP'])
```

```
[206]: <Axes: ylabel='BP'>
```

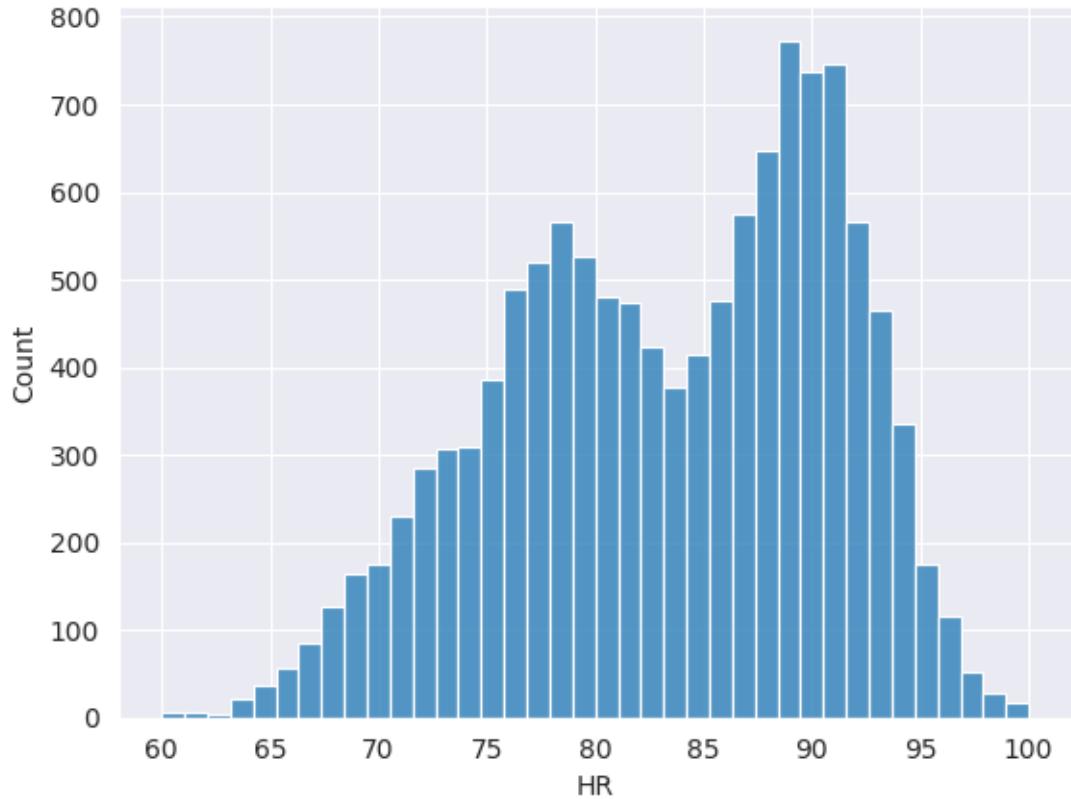


```
[207]: observation_df['BP'].min(), observation_df['BP'].max()
```

```
[207]: (np.float64(90.0), np.float64(120.0))
```

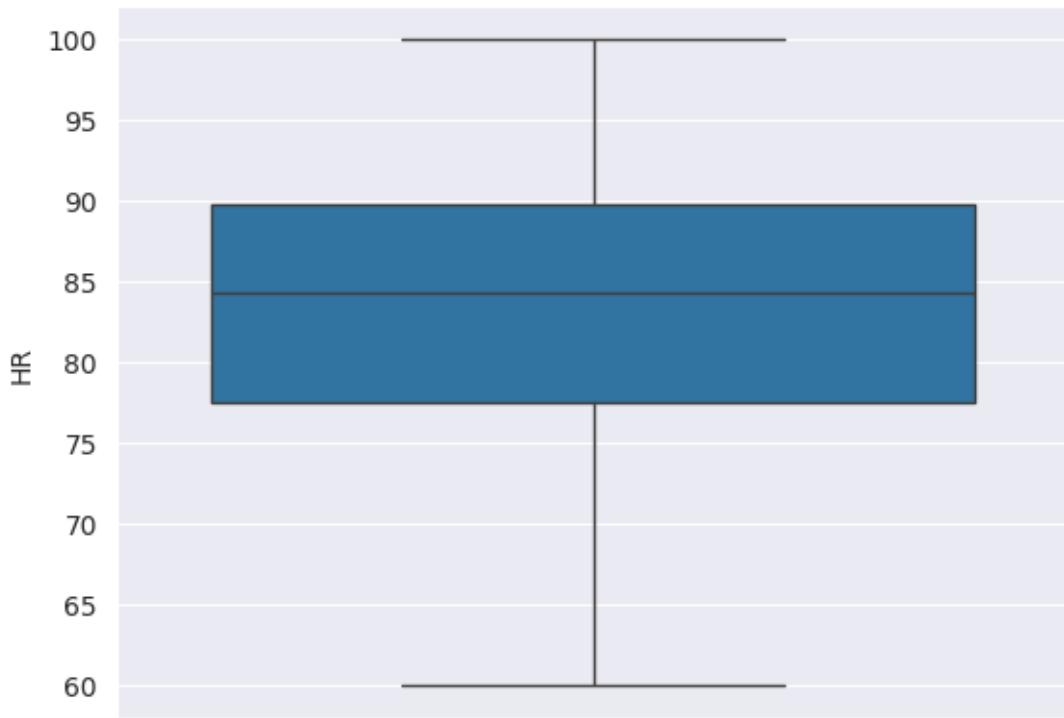
```
[208]: #Heart rate
sns.histplot(observation_df['HR'])
#pretty bimodal, but realistic values
```

```
[208]: <Axes: xlabel='HR', ylabel='Count'>
```



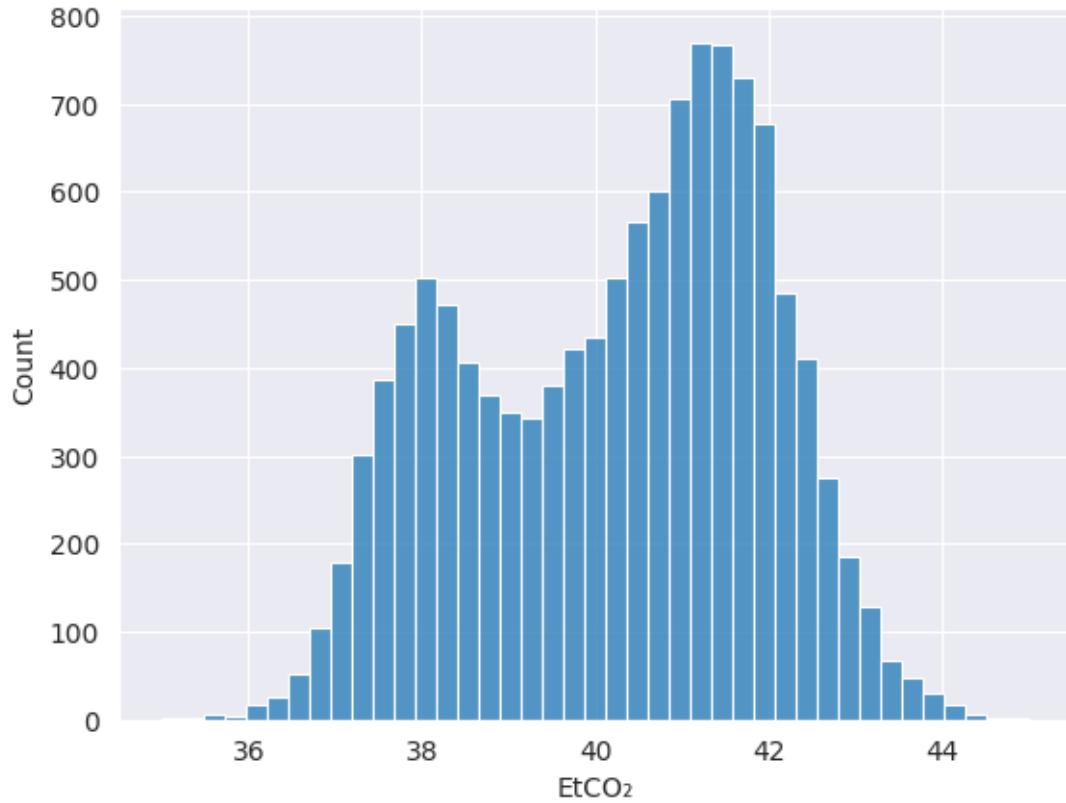
```
[209]: sns.boxplot(y = observation_df['HR'])
```

```
[209]: <Axes: ylabel='HR'>
```



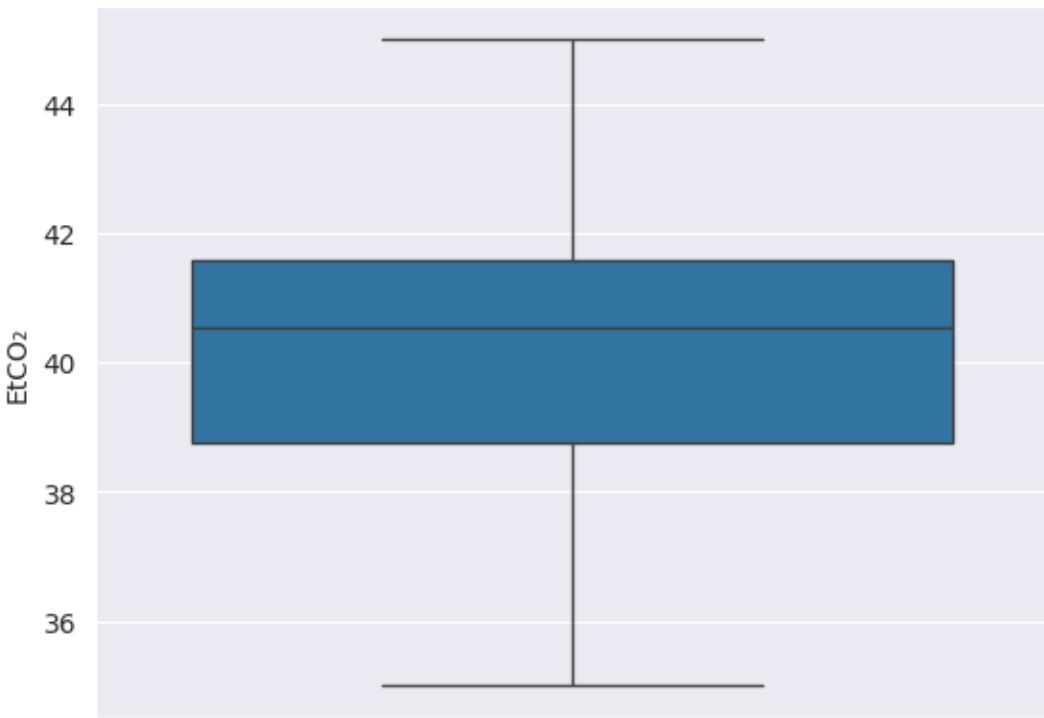
```
[210]: #ETCO2  
sns.histplot(observation_df['EtCO '])  
#also bimodal
```

```
[210]: <Axes: xlabel='EtCO ', ylabel='Count'>
```



```
[211]: sns.boxplot(observation_df['EtCO'])
```

```
[211]: <Axes: ylabel='EtCO'>
```



```
[212]: observation_df['EtCO2'].min(), observation_df['EtCO2'].max()
#also usual values
```

```
[212]: (np.float64(35.0), np.float64(45.0))
```

```
[213]: #just checking if this is relevant
patient_df['company'].nunique(), patient_df.shape
#too many unique vals, irrelevant
```

```
[213]: (1989, (2197, 13))
```

### 3.1.3 C) Párová analýza dát: Identifikujte vzťahy a závislostí medzi dvojicami atribútov.

But first lets create a dataset with merged tables with only the attributes we consider as necessary

#### Creation of the joined df

```
[214]: #This is a list of attributes that will make up the new merged dataset
#QoS might be of significance with Fio and other attrs,
#some codes may also have a touch (for example US station are healthier... idk)
#latitude and longitude will be kept for potential current location ↴
#(patient_df) relation
```

```

station_attributes = ['QoS', 'code', 'latitude', 'longitude', ]

#current location has been mentioned in station attrs, may be dropped -> later
#comment : it was dropped
#blood group may be relevant, may be not we will see
#user id just so we have some unique user identifier since we are dropping name
#and everything
patient_attributes = ['user_id', 'blood_group', 'station_ID']

#everything apart from longitude and latitude since that is already kept from
#station_attributes and we mentioned before that all stations are mentioned
#in obs_df
observation_attributes = ['SpO ', 'HR', 'PI', 'RR', 'EtCO ', 'FiO ', 'PRV', 'BP',
    'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
    'CO', 'Blood Flow Index', 'PPG waveform features',
    'Signal Quality Index', 'Respiratory effort', 'O extraction ratio',
    'SNR', 'oximetry']

#for check, lets count the attrs
len(station_attributes) + len(patient_attributes) + len(observation_attributes)

```

[214]: 28

[215]: station\_df2 = station\_df[station\_attributes].copy()  
 patient\_df2 = patient\_df[patient\_attributes].copy()  
 observation\_df2 = observation\_df[observation\_attributes + ['latitude',  
 #longitude']].copy()  
 #this is needed to convert the ID into a normal attribute  
 station\_df2 = station\_df2.reset\_index().rename(columns={'index': 'station\_ID'})

[216]: observation\_df.shape

[216]: (12177, 23)

[217]: df\_obs\_stat = observation\_df2.merge(  
 station\_df2,  
 left\_on=['latitude', 'longitude'], # from merged patients  
 right\_on=['latitude', 'longitude'], # from observation\_df  
 how='inner'  
 )

[218]: #the shape should be the same as observation\_df.shape  
 df\_obs\_stat.shape  
 #it is not because in the following cell we can see that some stations have the  
 #same coordinates

[218]: (21385, 26)

```
[219]: station_df2[['latitude', 'longitude']].duplicated().sum()
```

```
[219]: np.int64(205)
```

```
[220]: df_obs_stat.head(5)
```

```
[220]:          SpO          HR          PI          RR          EtCO         FiO \
0  97.538229  87.194745  11.225419  14.812012  42.113735  33.852538
1  97.933271  80.787303  11.730935  14.964972  39.537692  65.326035
2  97.933271  80.787303  11.730935  14.964972  39.537692  65.326035
3  98.209983  79.733895  12.839449  14.840668  39.758706  53.925230
4  98.202790  86.156903  11.204152  14.523288  43.448577  35.227704
```

```
          PRV          BP Skin Temperature Motion/Activity index ...
0  144.504405  100.455727           35.961920           10.302567 ...
1  110.615787  102.133386           36.274352           8.975704 ...
2  110.615787  102.133386           36.274352           8.975704 ...
3  107.208040  104.036654           35.583851           7.653790 ...
4  143.282224  105.723603           36.463180           8.795732 ...
```

```
      Signal Quality Index Respiratory effort O extraction ratio        SNR \
0            42.399816           46.497869           0.289012  39.334620
1            46.078137           53.351208           0.290879  26.006709
2            46.078137           53.351208           0.290879  26.006709
3            41.525607           52.124182           0.263171  31.890829
4            36.535021           50.342830           0.256780  30.721375
```

```
      oximetry   latitude  longitude station_ID        QoS code
0       1.0    49.183239  15.454273       403     good  CZ
1       0.0    33.544280 -84.233810        4     good  US
2       0.0    33.544280 -84.233810       426  excellent  US
3       1.0   -27.505780  153.102360       219  excellent  AU
4       1.0    37.656390  126.835000       319  excellent  KR
```

[5 rows x 26 columns]

```
[221]: #now we need to join the df_obs_stat with patient_df
df = df_obs_stat.merge(
    patient_df2,
    left_on=['station_ID'],
    right_on=['station_ID'],
    how='inner'
)
```

```
[222]: df.shape
```

#now we have many more entries since there are many patients sharing the same station as we can see in the next cell

```
[222]: (66973, 28)
```

```
[223]: patient_df2[['station_ID']].duplicated().sum()
```

```
[223]: np.int64(1520)
```

```
[224]: df.columns
```

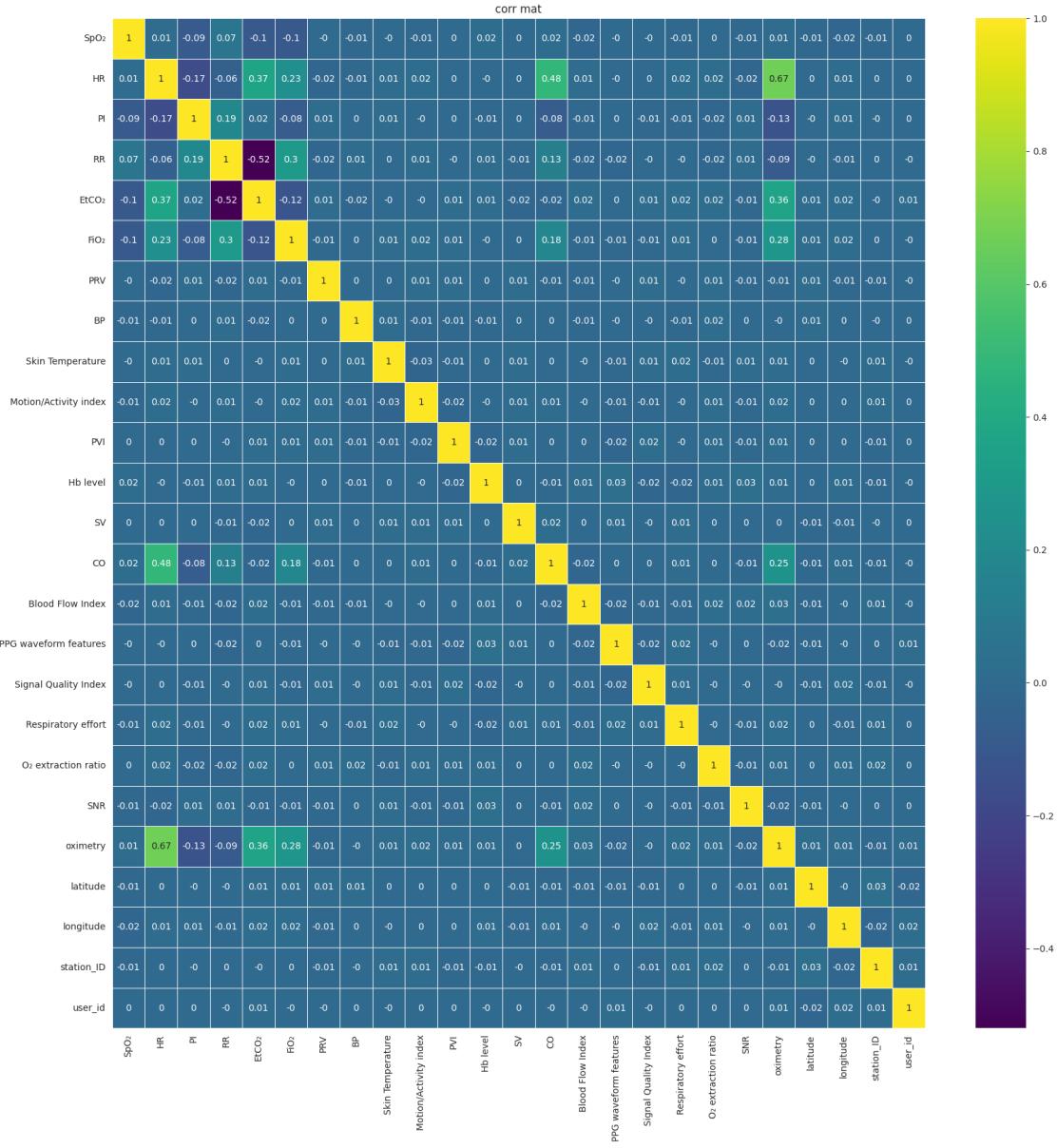
```
[224]: Index(['SpO ', 'HR', 'PI', 'RR', 'EtCO ', 'FiO ', 'PRV', 'BP',
       'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
       'CO', 'Blood Flow Index', 'PPG waveform features',
       'Signal Quality Index', 'Respiratory effort', 'O extraction ratio',
       'SNR', 'oximetry', 'latitude', 'longitude', 'station_ID', 'QoS', 'code',
       'user_id', 'blood_group'],
      dtype='object')
```

correlation matrix with heatmap

```
[225]: numeric_df = df.select_dtypes(include=['float64', 'int64'])
```

```
corr_matrix = numeric_df.corr().round(2)
```

```
[226]: plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix, cmap='viridis', annot=True, linewidths=0.5)
plt.title("corr mat")
plt.show()
```



```
[227]: #based on this heatmap lets write out all the correlations so we can take a look at them later
#I only included minimum 0,2 corr
```

```
#GROUP 1 (WITHOUT OXIMETRY): -- used in this step
```

```
#HR - EtCO2(weak), FiO2(weak), CO(mid)
#RR - Etco2(mid), FiO2(weak),
```

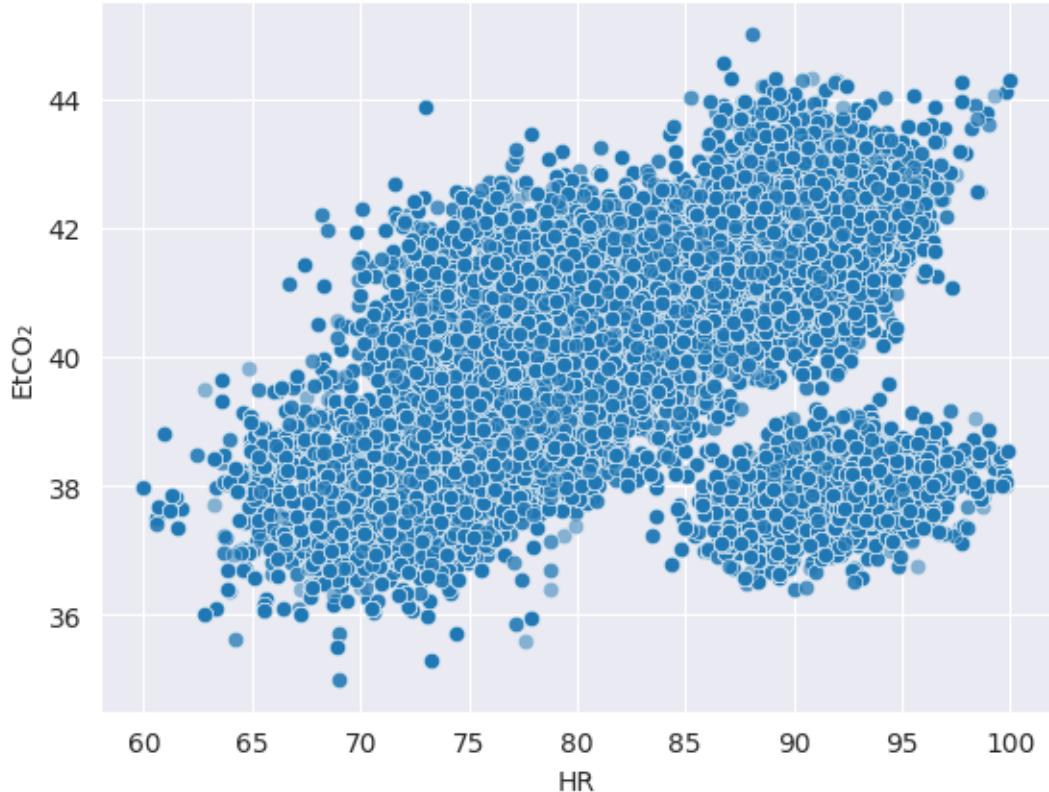
```
#Group 2(OXIMETRY):
```

```
#OX - HR(strong), EtCO2(weak) , FiO2(weak), CO(weak) -- used in 1D
```

**solution for 1.1C)** here are the comparisons between the attributes we got from the correlation heatmap

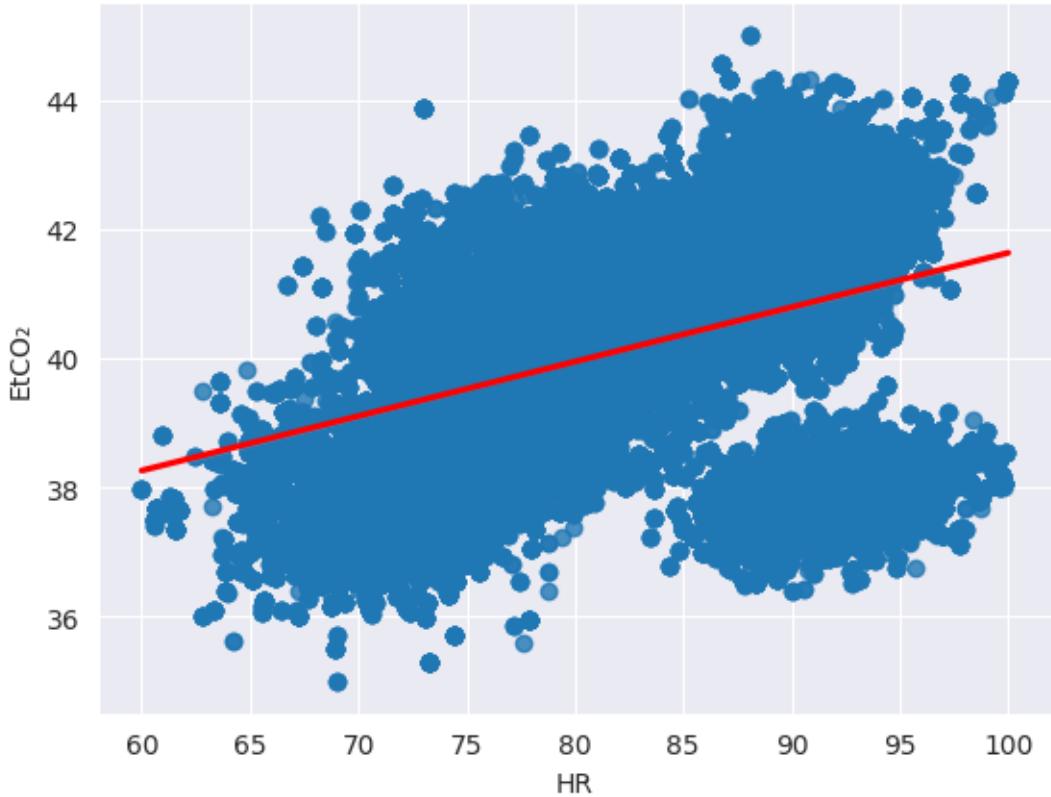
```
[228]: #HR-EtCO2  
sns.scatterplot(data=df, x='HR', y='EtCO2', alpha=0.5)
```

```
[228]: <Axes: xlabel='HR', ylabel='EtCO2'>
```



```
[229]: sns.regplot(data=df, x='HR', y='EtCO2', line_kws={'color':'red'})
```

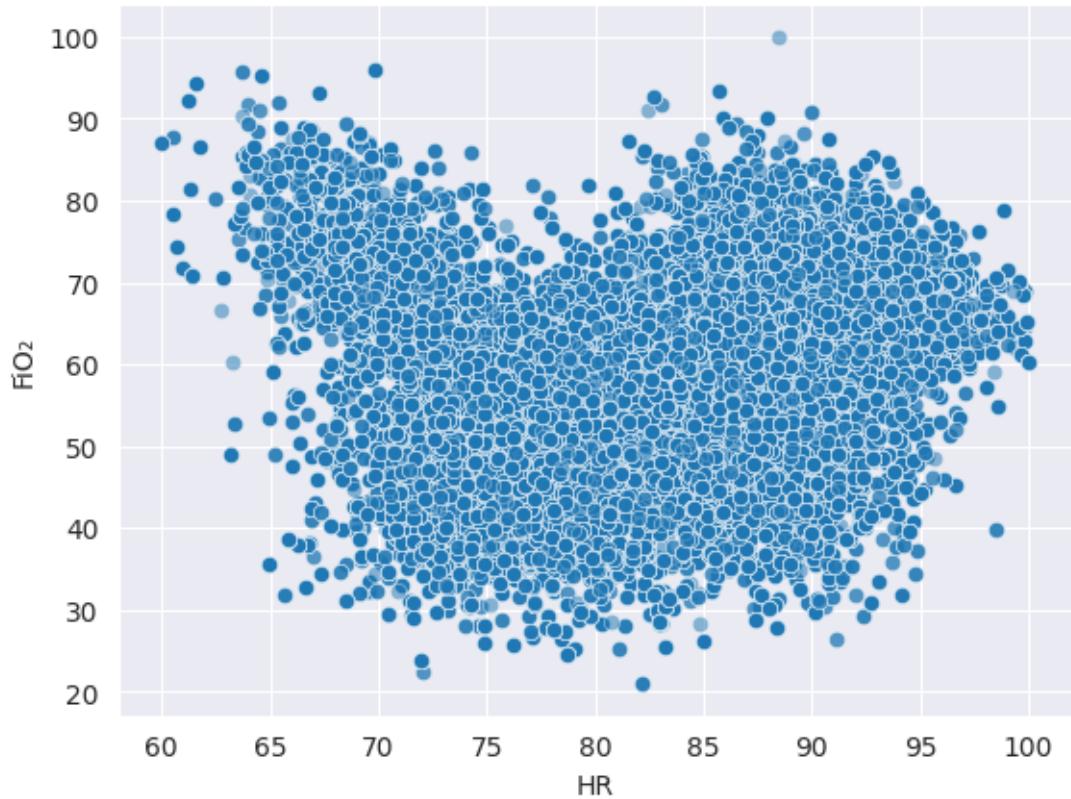
```
[229]: <Axes: xlabel='HR', ylabel='EtCO2'>
```



```
[230]: #this is not enough to determine that these two variables correlate, maybe without the cluster around [92.5 , 38]
```

```
[231]: #HR-FiO2  
sns.scatterplot(data=df, x='HR', y='FiO ', alpha=0.5)
```

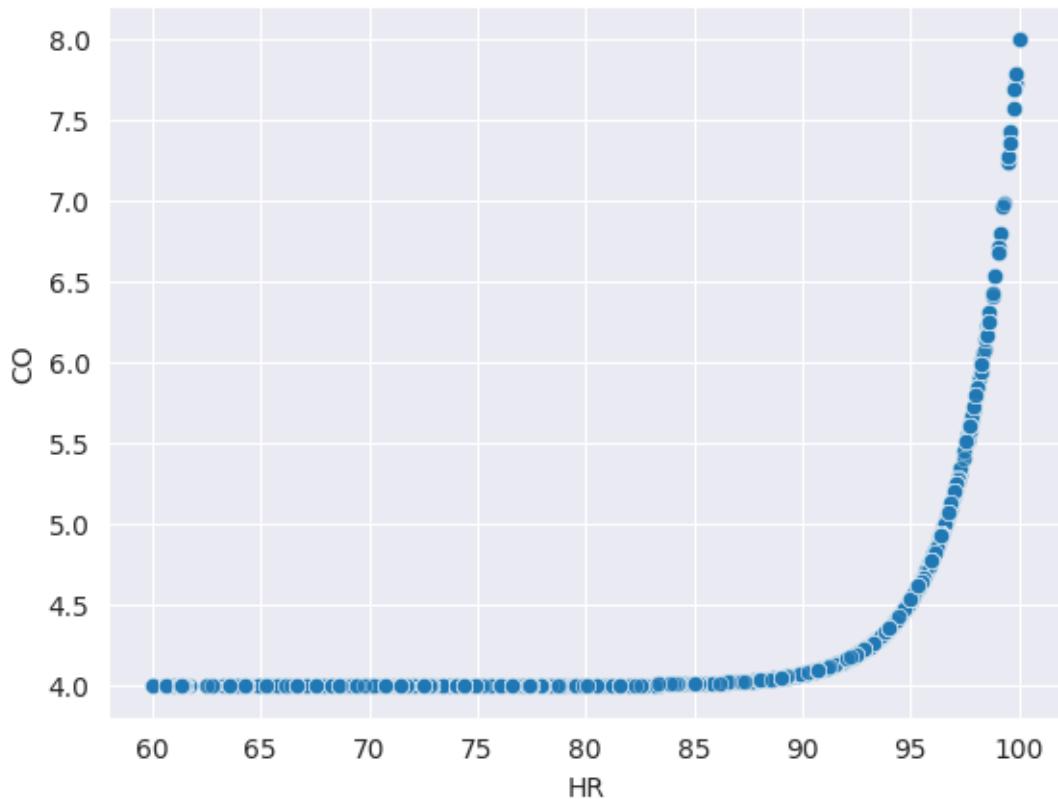
```
[231]: <Axes: xlabel='HR', ylabel='FiO '>
```



[232]: *#way too spread out, insignificant*

[233]: `#HR-CO  
sns.scatterplot(data=df, x='HR', y='CO', alpha=0.5)`

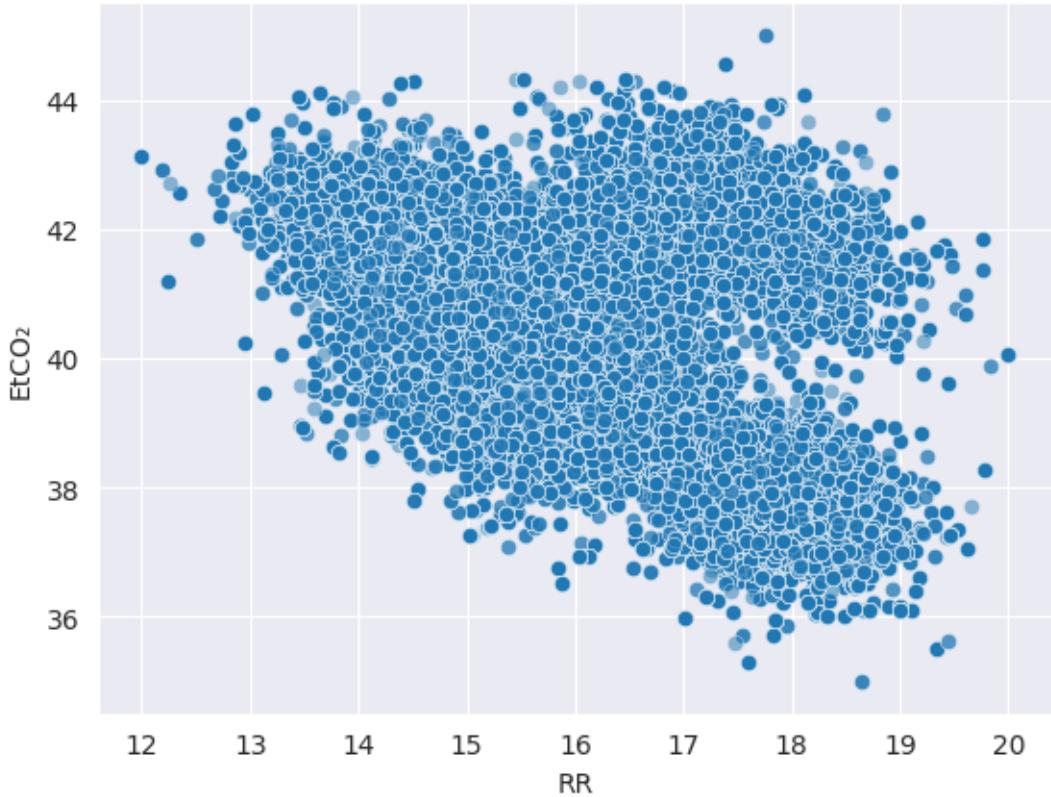
[233]: <Axes: xlabel='HR', ylabel='CO'>



```
[234]: #This is definitely a significant correlation, but not linear since CO values  
↪are cut off at 4.0
```

```
[235]: #RR - Etco2  
sns.scatterplot(data=df, x='RR', y='EtCO ', alpha=0.5)
```

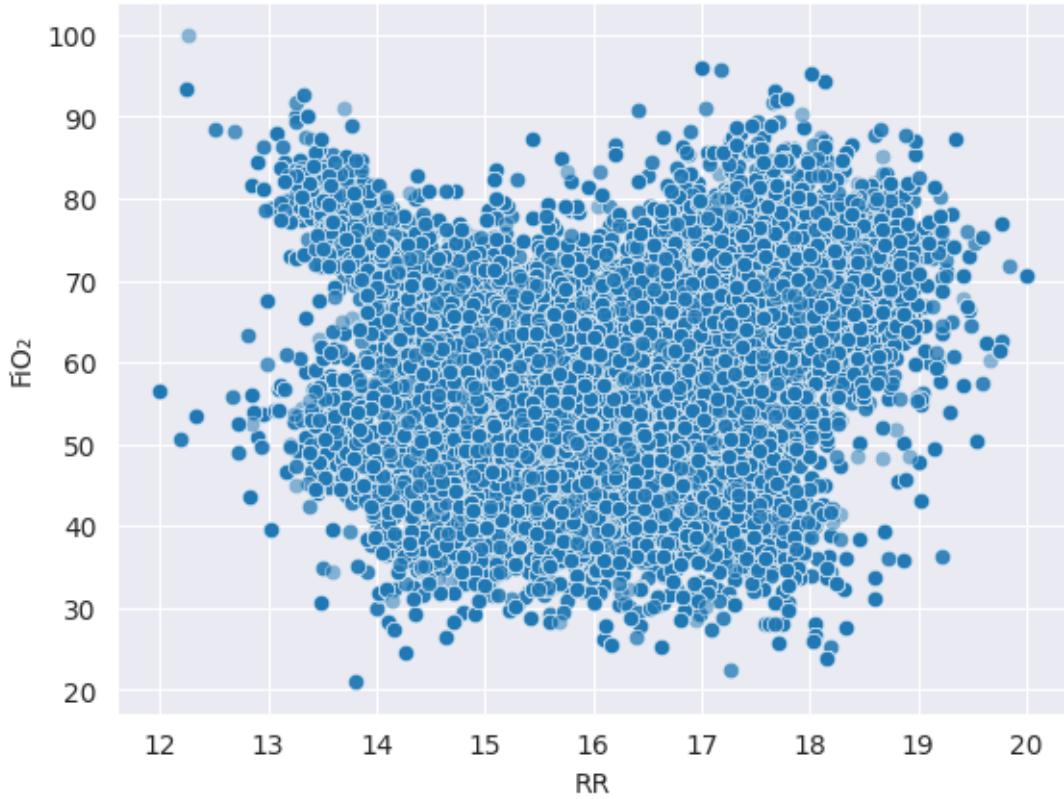
```
[235]: <Axes: xlabel='RR', ylabel='EtCO '>
```



[236]: *#slight show of negative correlation, but ruined by the upper right cluster*

[237]: *#RR FiO2*  
sns.scatterplot(data=df, x='RR', y='FiO ', alpha=0.5)

[237]: <Axes: xlabel='RR', ylabel='FiO '>



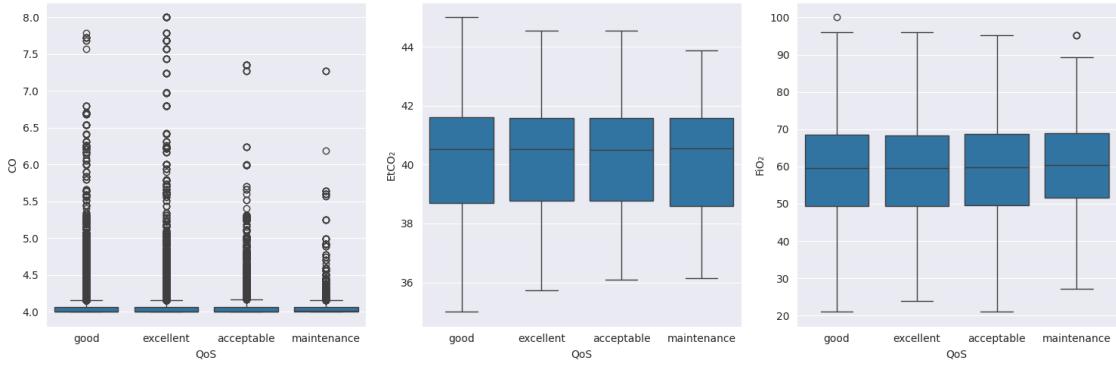
[238]: `#no correlation`

these were only the numeric attributes we took from the heatmap, now lets take a look at non numeric correlations aswell

[239]: `non_numeric = df.select_dtypes(exclude=['float64', 'int64']).columns`  
`print(non_numeric)`  
*#these will be tested for correlation with CO,FiO2 and EtCO2 since these seem ↴ to be the most important attributes*

`Index(['QoS', 'code', 'blood_group'], dtype='object')`

[240]: `#QoS`  
`fig, axes = plt.subplots(1, 3, figsize=(15, 5))`  
  
`sns.boxplot(data=df, x='QoS', y='CO', ax=axes[0])`  
`sns.boxplot(data=df, x='QoS', y='EtCO', ax=axes[1])`  
`sns.boxplot(data=df, x='QoS', y='FiO', ax=axes[2])`  
  
`plt.tight_layout()`  
`plt.show()`  
*#no real significance*

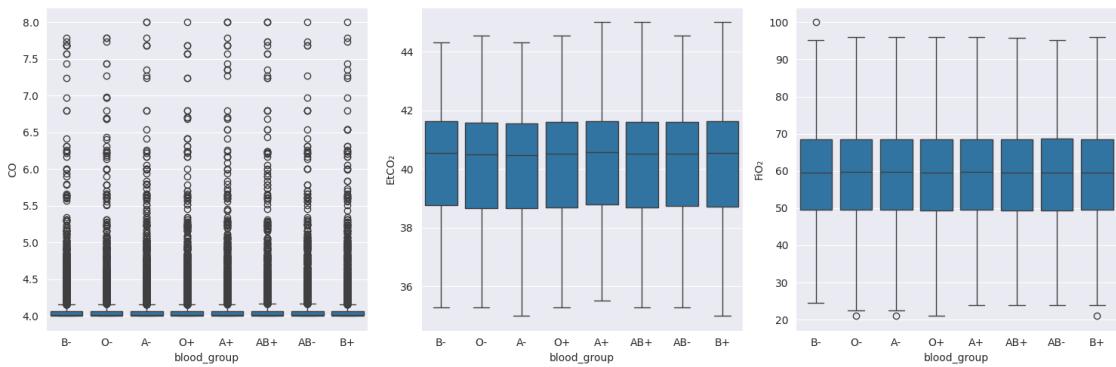


[241]: #QoS

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

sns.boxplot(data=df, x='blood_group', y='CO', ax=axes[0])
sns.boxplot(data=df, x='blood_group', y='EtCO', ax=axes[1])
sns.boxplot(data=df, x='blood_group', y='FiO', ax=axes[2])

plt.tight_layout()
plt.show()
```



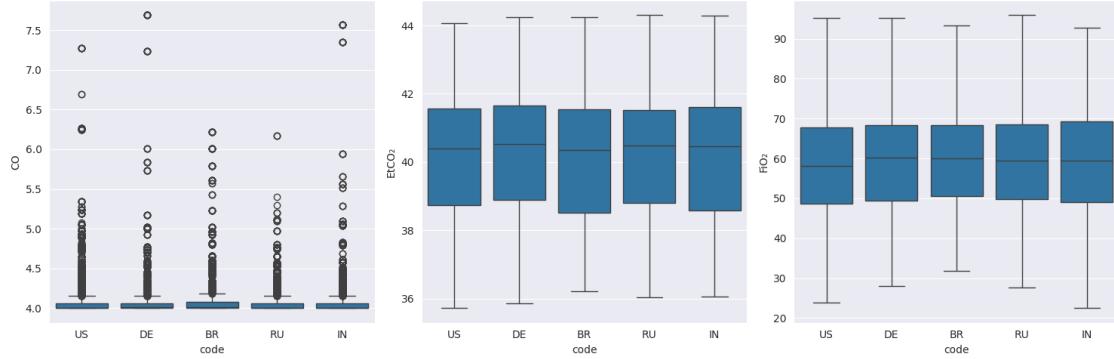
[242]: #code - top 5

```
top_codes = df['code'].value_counts().head(5).index
filtered_df = df[df['code'].isin(top_codes)]

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

sns.boxplot(data=filtered_df, x='code', y='CO', ax=axes[0])
sns.boxplot(data=filtered_df, x='code', y='EtCO', ax=axes[1])
sns.boxplot(data=filtered_df, x='code', y='FiO', ax=axes[2])
```

```
plt.tight_layout()
plt.show()
```



### 3.1.4 D) Párová analýza dát: Identifikujte závislosti medzi predikovanou premennou a ostatnými premennými (potenciálnymi prediktormi)

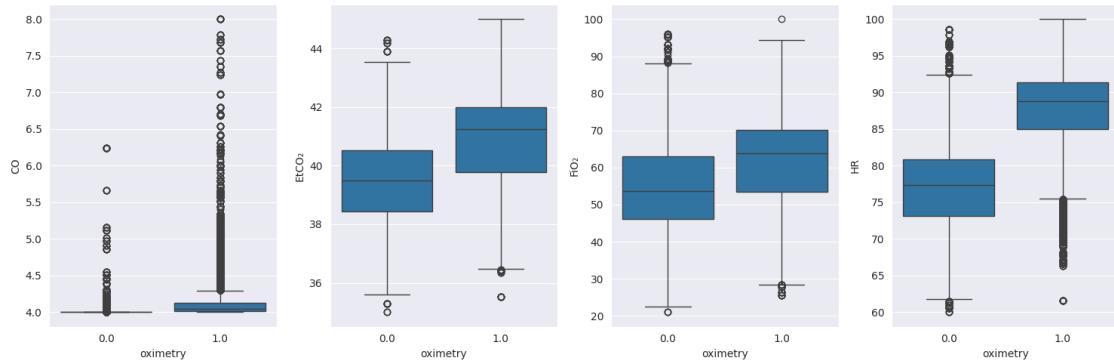
from the correlation heatmap we have these attributes that seem to have some correlation with the predicted attribute

HR(strong),EtCO2(weak) ,FiO2(weak), CO(weak)

```
[243]: fig, axes = plt.subplots(1, 4, figsize=(15, 5))
```

```
sns.boxplot(data=df, x='oximetry', y='CO', ax=axes[0])
sns.boxplot(data=df, x='oximetry', y='EtCO', ax=axes[1])
sns.boxplot(data=df, x='oximetry', y='FiO', ax=axes[2])
sns.boxplot(data=df, x='oximetry', y='HR', ax=axes[3])

plt.tight_layout()
plt.show()
```



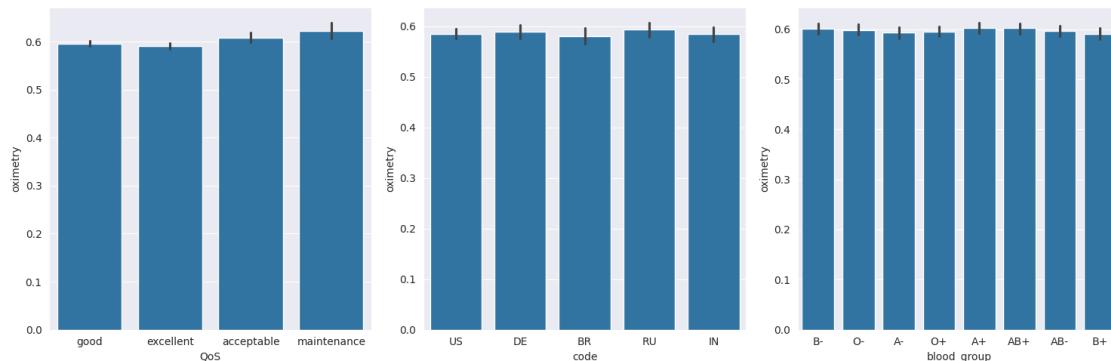
[244]: #CO is hard to determine, the IQRs are close but the outliers may mean a lot  
 ↳(will probably see more after normalization), the EtCO2 and FiO2 seem to be  
 ↳higher when oximetry is set to 1, HR seems to have a really significant  
 ↳correlation with oximetry, as suggested by the correlation heatmap

[245]: #correlation with non numerical values

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# QoS
sns.barplot(ax=axes[0], x=df['QoS'], y=df['oximetry'])
# code (top 5)
top5 = df['code'].value_counts().head(5).index
sns.barplot(ax=axes[1], x=df[df['code'].isin(top5)]['code'], y=df[df['code'].isin(top5)]['oximetry'])
# blood_group
sns.barplot(ax=axes[2], x=df['blood_group'], y=df['oximetry'])

plt.tight_layout()
plt.show()
```



[246]: #this graphs show the oximetry 1/0 ratio per each QoS, Code and blood group. So  
 ↳rather than correlation, this shows that the data are evenly distributed

### 3.1.5 E) Findings, thought ect.

all of our thoughts and findings are thoroughly documented throughout the operations, but here is the summary:

Firstly we analysed each table and gave a quick look for every attribute, for dataset station, we used the coordinations to merge it with observation\_df, we ended up not using the station name as it is insignificant as well as revision. The only missing values were 2 codes which will probably be filled later with a newly created code. The patient dataframe is full of insignificant attributes, we end up only using the blood group and station ID for merge with station df. From observation we keep every value, although some seem to be more significant than others, for example FiO2, CO, RR, EtCO2. We take some time to look at individual distributions of attributes in B), but we dont

really find any abnormalities that would pose a threat to our models precision, most distributions are either normal or bimodal or uniform. In C) we needed to take a look at correlation between the attributes, so we needed to create a correlation heatmap which was created from a joint dataframe consisting of all three datasets, but only the attributes that we deemed as important. We discarded the useless features and we will not work with them from that point onward. However we find out minor correlation between HR and EtCO2 as well as a negative one between HR and FiO2 and a RR and EtCO2. HR and CO seem to have a strong positive correlation but since CO is capped at 4.0, it is not shown that nicely. In D) we see that the predicted value oximetry strongly correlates with HR, and FiO2 and EtCO2 also show medium signs of correlation. Correlation with CO is harder to determine without normalising the data first.

### 3.2 1.2 Identifikácia problémov, integrácia a čistenie dát

#### 3.2.1 A) nevhodná štruktúra, nejednotne formaty ,duplikáty, chýbajúce hodnoty, vychýlene hodnoty, abnormalne hodnoty, nelogické vzťahy

[247]: df.columns

```
[247]: Index(['SpO ', 'HR', 'PI', 'RR', 'EtCO ', 'FiO ', 'PRV', 'BP',
       'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
       'CO', 'Blood Flow Index', 'PPG waveform features',
       'Signal Quality Index', 'Respiratory effort', 'O extraction ratio',
       'SNR', 'oximetry', 'latitude', 'longitude', 'station_ID', 'QoS', 'code',
       'user_id', 'blood_group'],
      dtype='object')
```

[248]: # I see many column name that are correctly name but it is so annoying to copy ↵ the same `` symbol over and over again so that's why im gonna rename it just ↵ for now

```
rename_map = {
    'SpO ': 'SpO2',
    'EtCO ': 'EtCO2',
    'FiO ': 'FiO2',
    'O extraction ratio': 'O2 extraction ratio',
}
df = df.rename(columns={k:v for k,v in rename_map.items() if k in df.columns})
```

[249]: for c in ['QoS', 'code', 'blood\_group']:
 if c in df.columns:
 df[c] = df[c].astype('category')

[250]: #checking if every value in oximetry col is type int so it does not make any ↵ issues in the future but most of this is already done in previous cells in 1. ↵ 1

```
if 'oximetry' in df.columns:
    df['oximetry'] = df['oximetry'].astype(int)
```

```
[251]: df.dtypes.value_counts()
```

```
[251]: float64      22
       int64        3
       category     1
       category     1
       category     1
Name: count, dtype: int64
```

```
[252]: df.duplicated().sum()
```

```
[252]: np.int64(0)
```

```
[253]: df.duplicated(subset=['HR', 'RR', 'BP']).sum()
```

```
[253]: np.int64(55244)
```

```
[254]: duplicates_output = df[df.duplicated(subset=[col for col in df.columns if col != 'user_id'], keep=False)]
duplicates_output
```

	Sp02	HR	PI	RR	EtCO2	Fi02	\
6	97.933271	80.787303	11.730935	14.964972	39.537692	65.326035	
7	97.933271	80.787303	11.730935	14.964972	39.537692	65.326035	
9	97.933271	80.787303	11.730935	14.964972	39.537692	65.326035	
30	96.851933	91.008225	11.323571	15.148349	42.056062	58.615353	
32	96.851933	91.008225	11.323571	15.148349	42.056062	58.615353	
...	...	...	...	...	...	...	
66965	96.698981	71.499309	13.427977	15.751513	41.201634	55.177636	
66966	96.698981	71.499309	13.427977	15.751513	41.201634	55.177636	
66967	96.698981	71.499309	13.427977	15.751513	41.201634	55.177636	
66968	96.698981	71.499309	13.427977	15.751513	41.201634	55.177636	
66970	96.698981	71.499309	13.427977	15.751513	41.201634	55.177636	
	PRV	BP	Skin Temperature	Motion/Activity index	...	\	
6	110.615787	102.133386	36.274352	8.975704	...		
7	110.615787	102.133386	36.274352	8.975704	...		
9	110.615787	102.133386	36.274352	8.975704	...		
30	77.660061	102.695264	35.163246	11.823341	...		
32	77.660061	102.695264	35.163246	11.823341	...		
...	...	...	...	...	...		
66965	103.386006	102.693239	34.895596	11.429244	...		
66966	103.386006	102.693239	34.895596	11.429244	...		
66967	103.386006	102.693239	34.895596	11.429244	...		
66968	103.386006	102.693239	34.895596	11.429244	...		
66970	103.386006	102.693239	34.895596	11.429244	...		
	O2 extraction ratio	SNR	oximetry	latitude	longitude	\	

```

6          0.290879  26.006709          0  33.54428 -84.23381
7          0.290879  26.006709          0  33.54428 -84.23381
9          0.290879  26.006709          0  33.54428 -84.23381
30         0.285955  32.768112          1  10.29085 105.75635
32         0.285955  32.768112          1  10.29085 105.75635
...
...        ...      ...
66965     0.283565  28.887543          1  51.04962 12.13690
66966     0.283565  28.887543          1  51.04962 12.13690
66967     0.283565  28.887543          1  51.04962 12.13690
66968     0.283565  28.887543          1  51.04962 12.13690
66970     0.283565  28.887543          1  51.04962 12.13690

      station_ID      QoS   code user_id blood_group
6           426 excellent   US    398      O+
7           426 excellent   US    988      O+
9           426 excellent   US   2033      O+
30          663 excellent   VN   1225      A-
32          663 excellent   VN   1454      A-
...
...        ...      ...
66965     223 excellent   DE    575      B-
66966     223 excellent   DE    468      O+
66967     223 excellent   DE   1212      B-
66968     223 excellent   DE    928      B-
66970     223 excellent   DE     51      B-

```

[21334 rows x 28 columns]

```
[255]: duplicates_output['user_id'].nunique(), len(duplicates_output)
```

```
[255]: (599, 21334)
```

```
[256]: duplicates_output['station_ID'].value_counts().head()
```

```
[256]: station_ID
186    468
8      414
350    352
208    318
223    318
Name: count, dtype: int64
```

After checking for duplicates, we found that some patients shared identical measurements. It is normal but not if we found out that 21 336 row that are similar. This was caused by merging patient\_df and observation\_df using station\_ID.

```
[257]: na_count = df.isna().sum().sort_values(ascending=False)
na_count
```

```
[257]: code           170
SpO2            0
PI              0
HR              0
EtCO2           0
FiO2            0
PRV             0
BP              0
Skin Temperature 0
Motion/Activity index 0
PVI             0
RR              0
Hb level        0
SV              0
Blood Flow Index 0
CO              0
Signal Quality Index 0
Respiratory effort 0
O2 extraction ratio 0
PPG waveform features 0
SNR             0
oximetry         0
longitude        0
latitude         0
station_ID       0
QoS              0
user_id          0
blood_group      0
dtype: int64
```

### 3.2.2 B) Kontrola správnosť v dátach

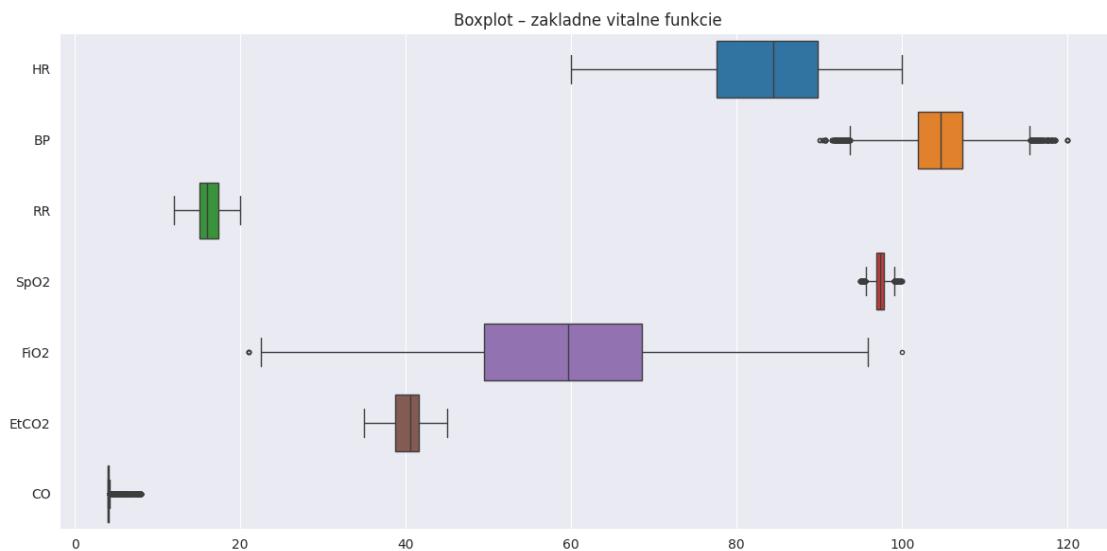
Most of the analyzing the attributes was done in part 1.1, individual findings were documented along the way.

Some of the most important findings that were captured along the way about the abnormality of the data could be divided into 3 groups. 1) Value-range-wise, most of the attributes have real values, for example the latitude and longitude do range from -180 to 180, SPO2 and blood groups are also in check and so are many more attributes. The only attribute where this was not the case would be FIO2, where the patients must have been altered from normal people, for example on oxygen therapy. 2) Distribution-wise, no attributes are abnormally different. Some distributions were normal, e.g. SPO2, Hb level. Some were uniform, for example blood group. Some were bimodal, for example HR or EtCO2 which does show quite a skew to abnormality. FIO2 is slightly abnormal, but in most values it would pass as a normal distribution. 3) Shape-wise, patients current location is mapped as an object, but we did not waste time with asserting this issue as we dropped the column altogether. The station's revision attribute is also an anomaly, since the shape is inconsistent, sometimes having only the date, sometimes also the time... The residence attribute is empty, having no entries

Illogical relationships would also be the division of the data into three tables. What struck me the most was the inability to find a patients observation, due to the fact that after merging patients and stations, we find out that multiple patient are linked to a singular station, and no further link to observation is provided apart than a 1:many relationship between a station and an observation with coordinates.

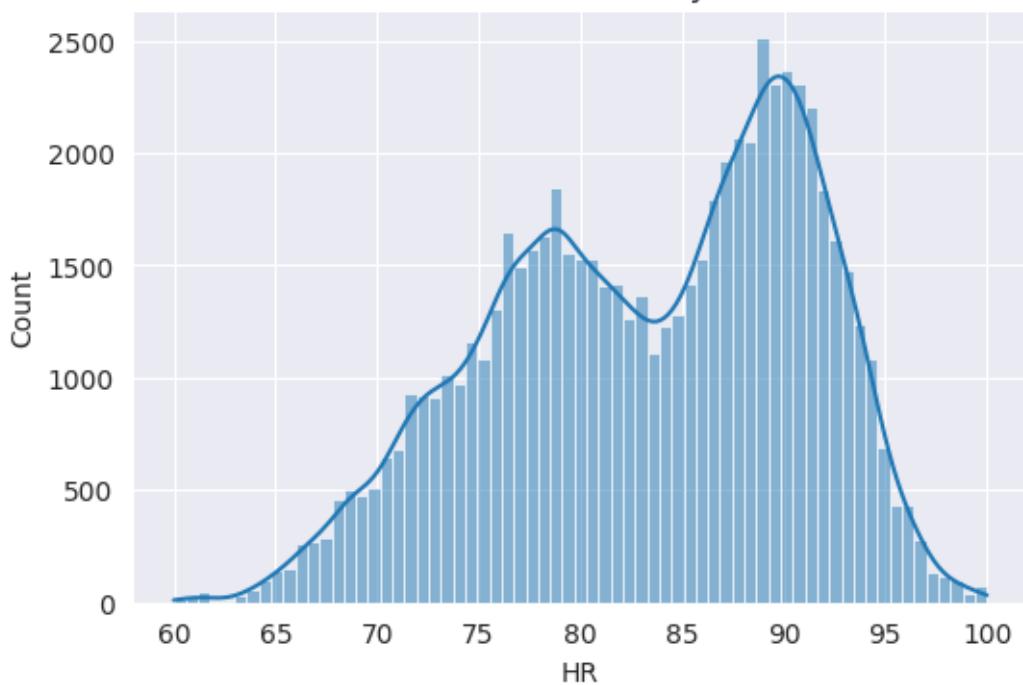
### 3.2.3 C) Outlier detection

```
[258]: cols1 = ['HR', 'BP', 'RR', 'SpO2', 'FiO2', 'EtCO2', "CO"]
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[cols1], orient='h', fliersize=3)
plt.title('Boxplot - zakladne vitalne funkcie')
plt.tight_layout()
plt.show()
```

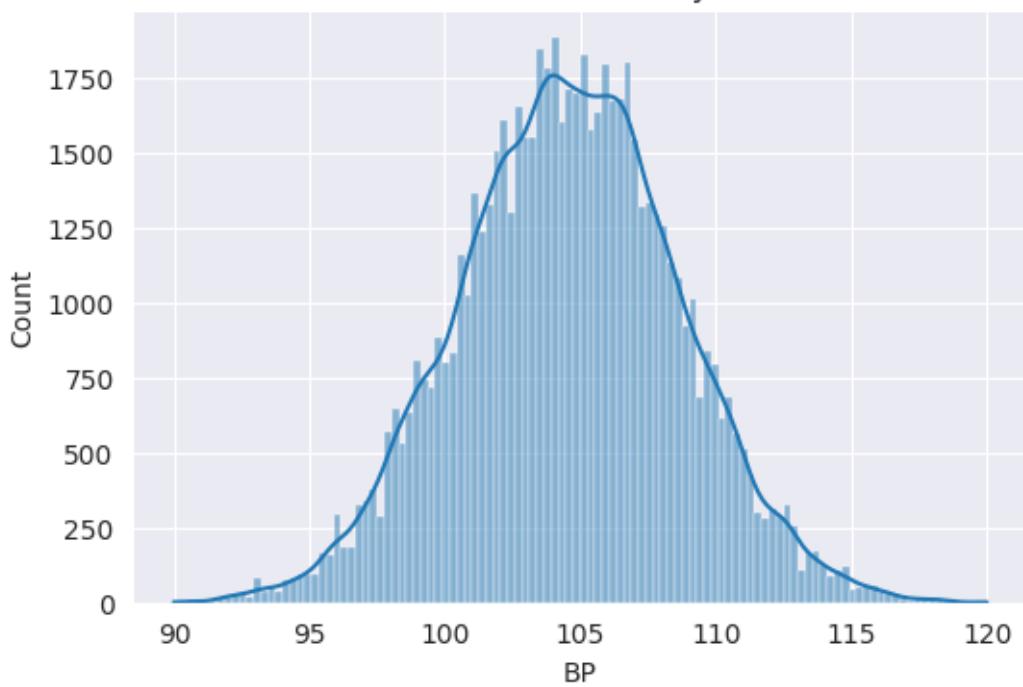


```
[259]: for col in cols1:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribucia hodnoty {col}')
    plt.show()
```

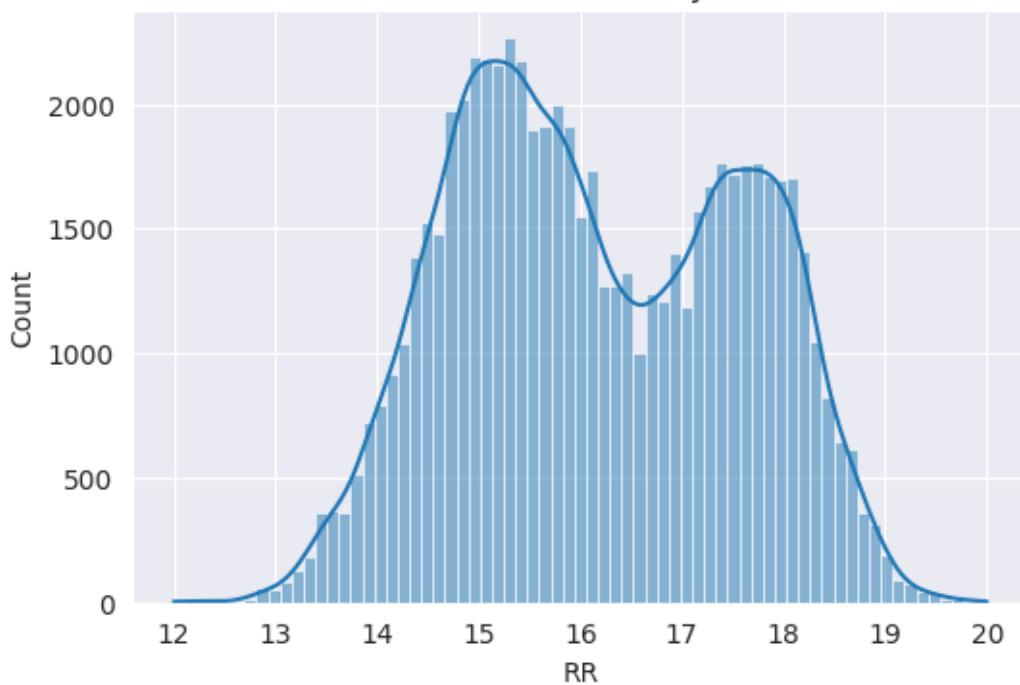
Distribucia hodnoty HR



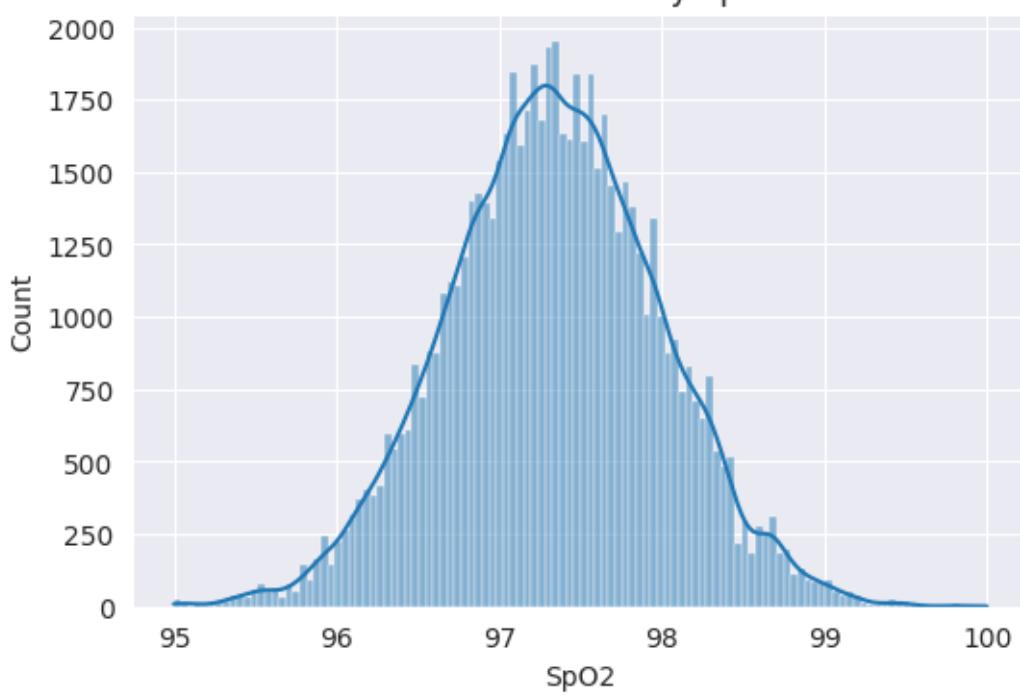
Distribucia hodnoty BP



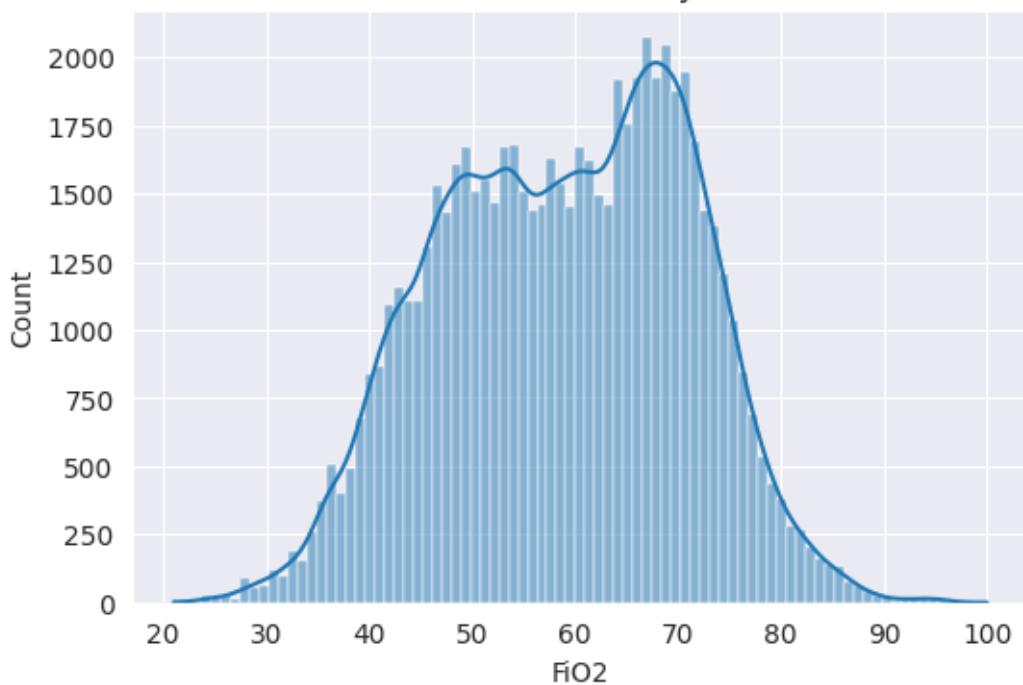
Distribucia hodnoty RR



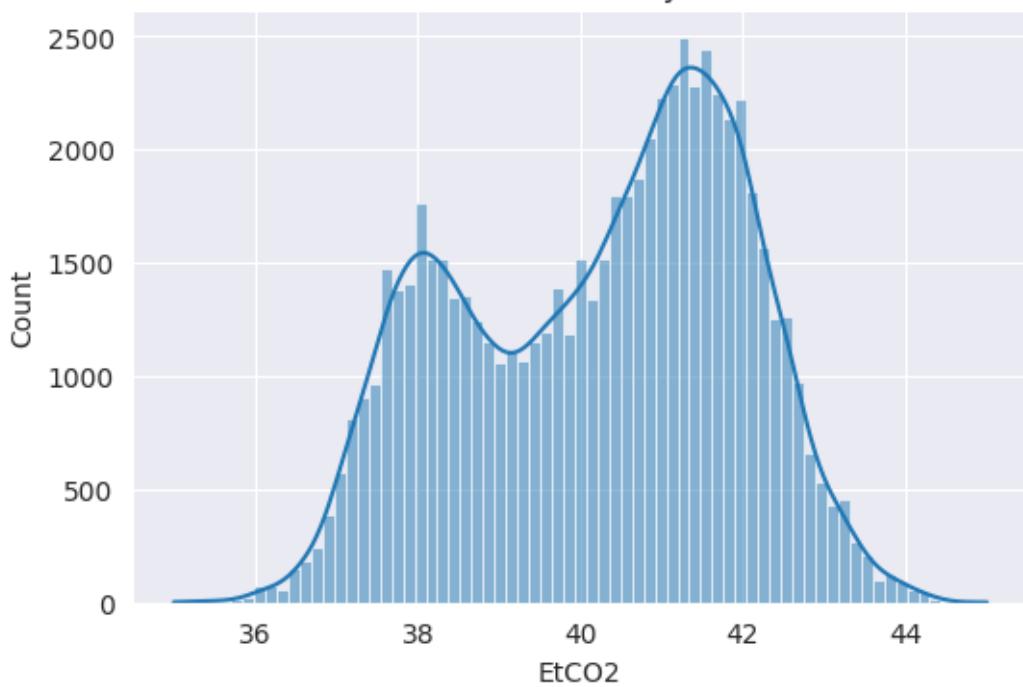
Distribucia hodnoty SpO<sub>2</sub>

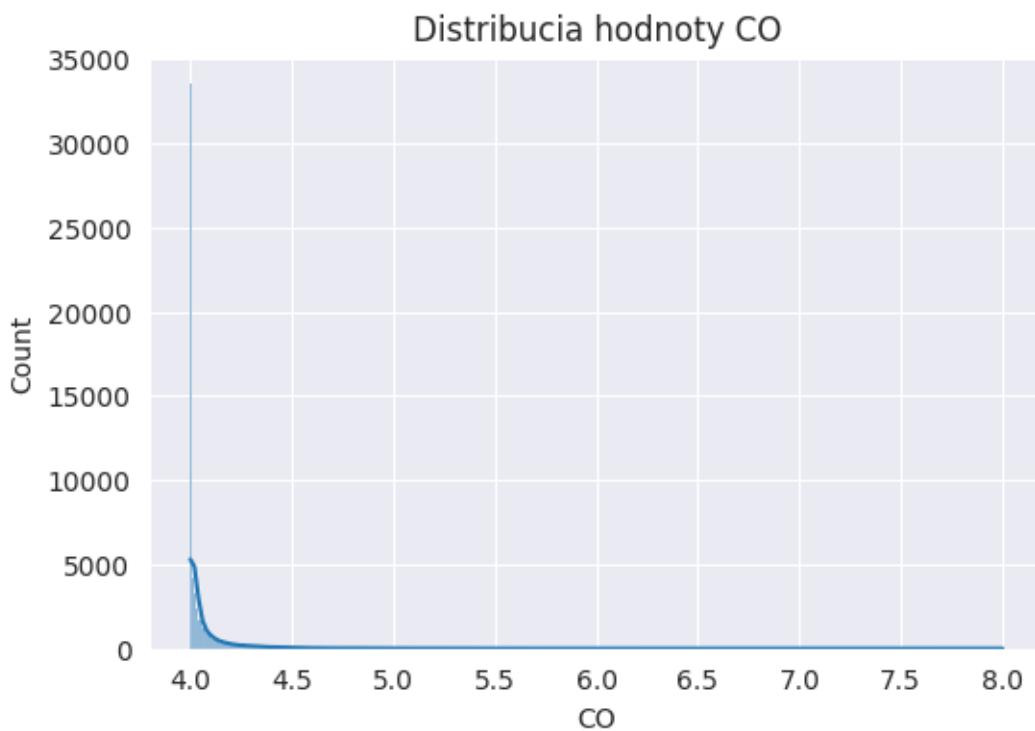


Distribucia hodnoty FiO<sub>2</sub>



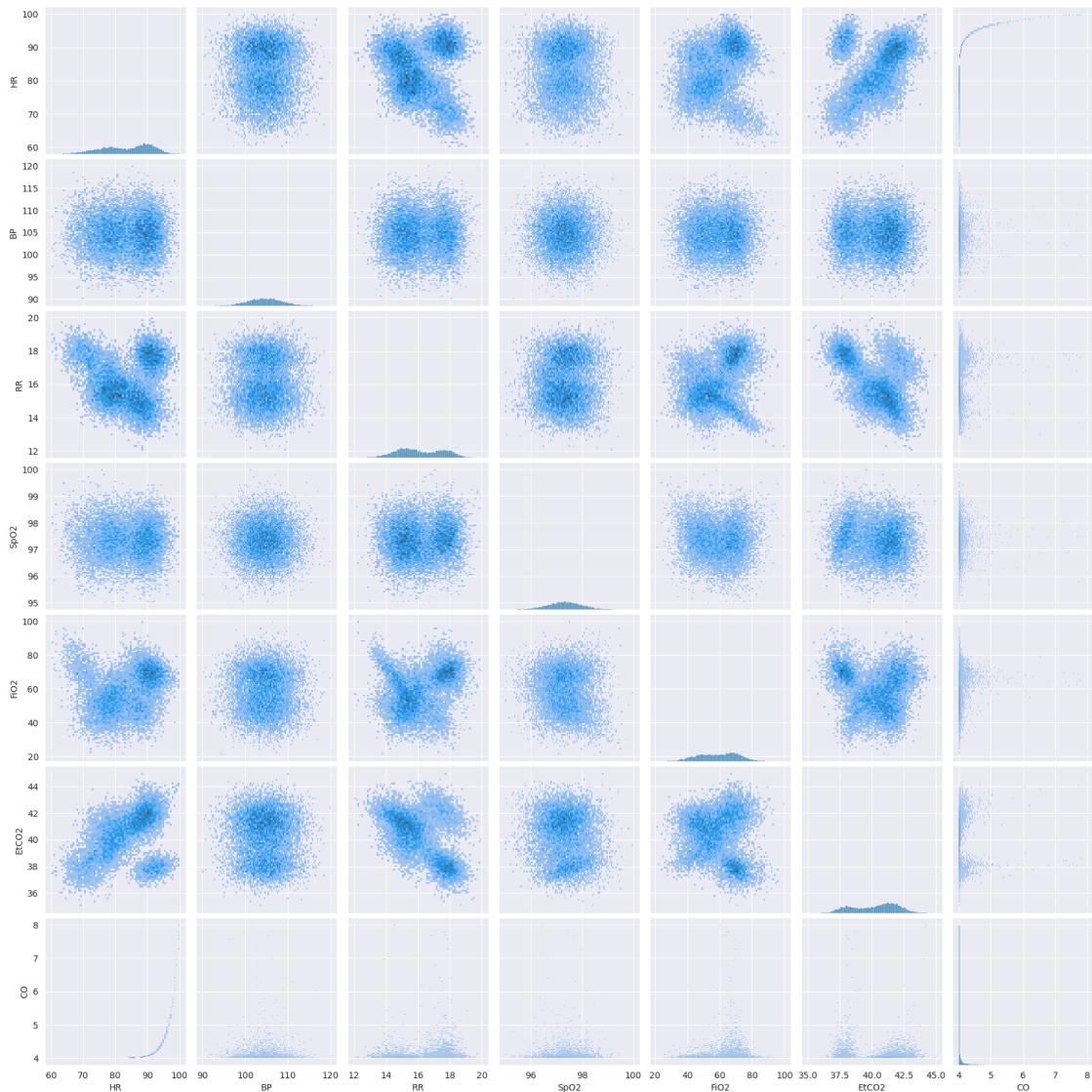
Distribucia hodnoty EtCO<sub>2</sub>





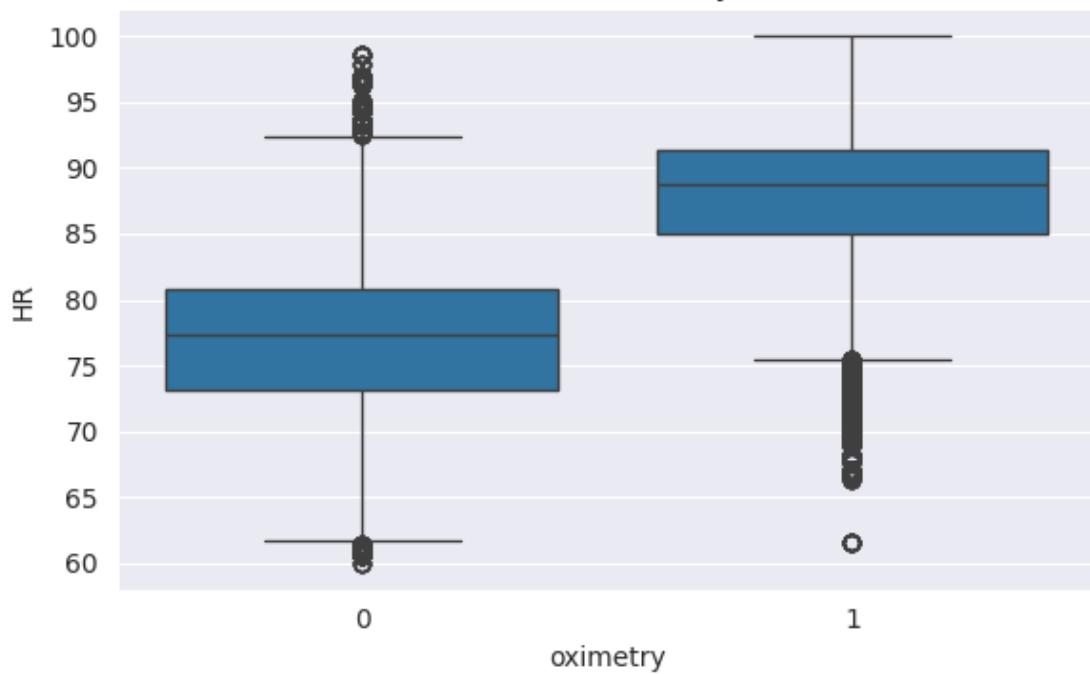
```
[260]: sns.pairplot(data=df[cols1], kind='hist')
```

```
[260]: <seaborn.axisgrid.PairGrid at 0x2faf57d7890>
```

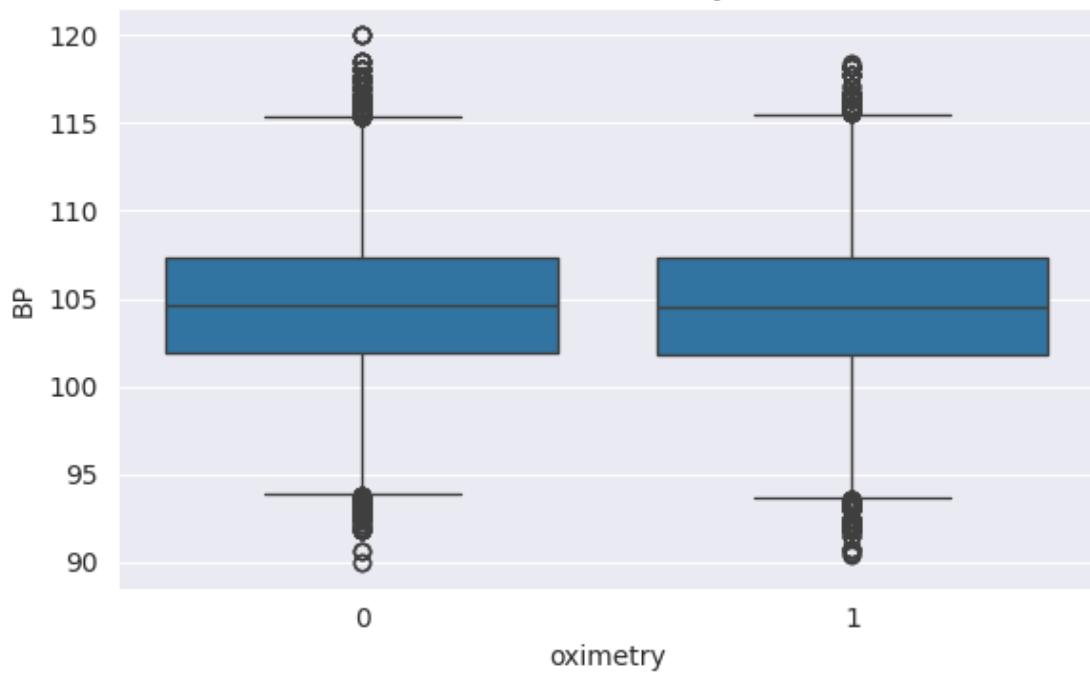


```
[261]: for col in cols1:
    plt.figure(figsize=(6,4))
    sns.boxplot(data=df, x='oximetry', y=col)
    plt.title(f'{col} vs. oximetry')
    plt.tight_layout()
    plt.show()
```

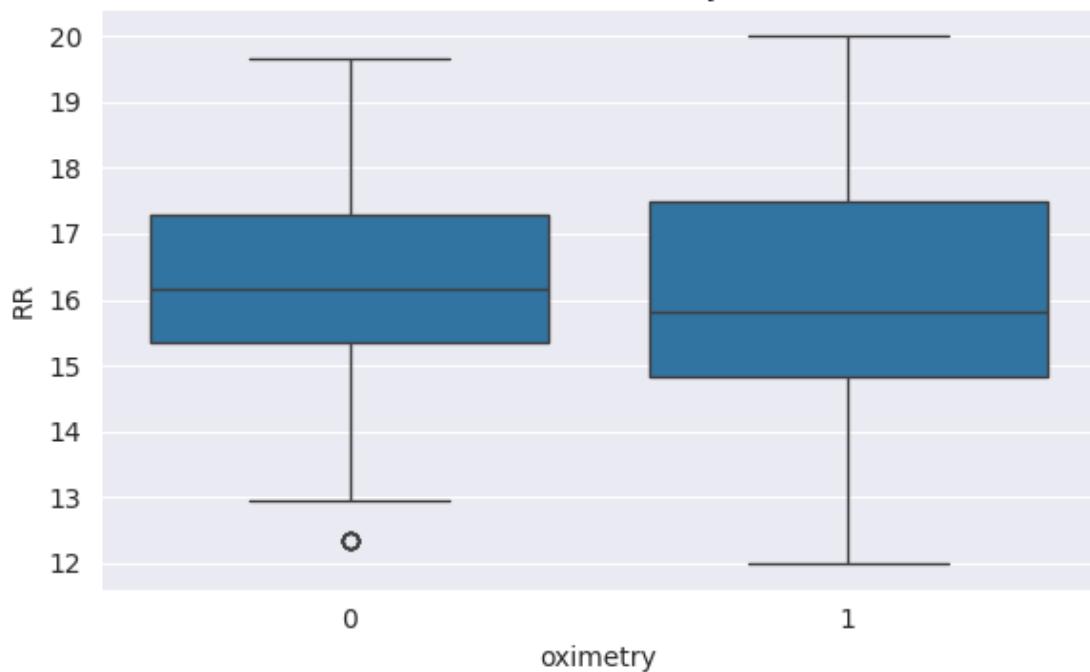
HR vs. oximetry



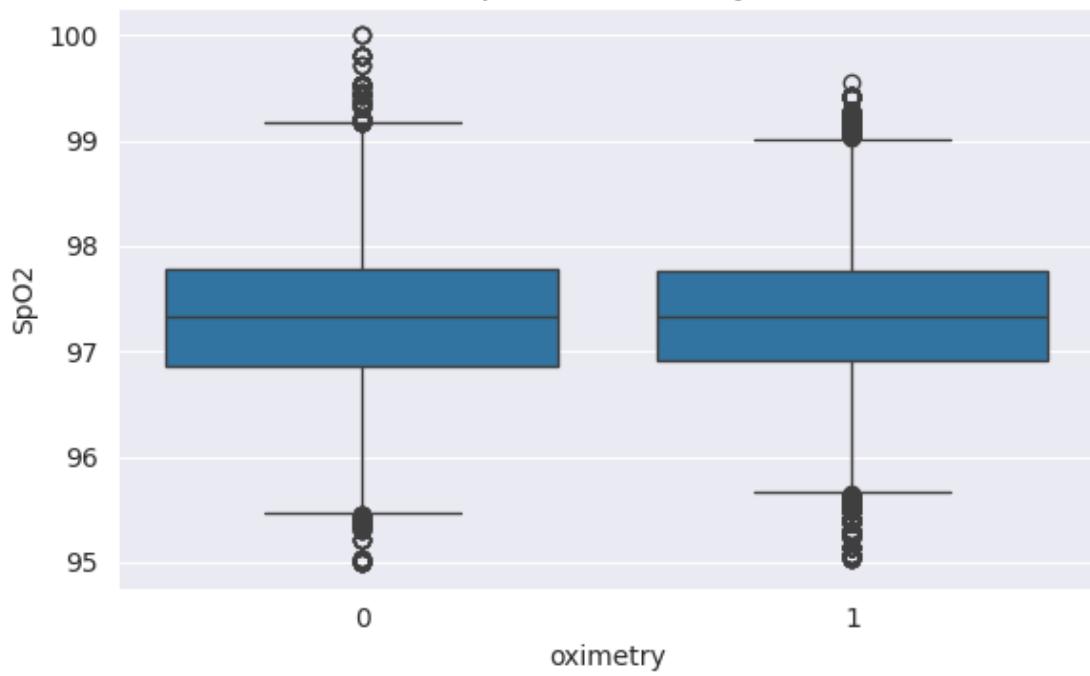
BP vs. oximetry



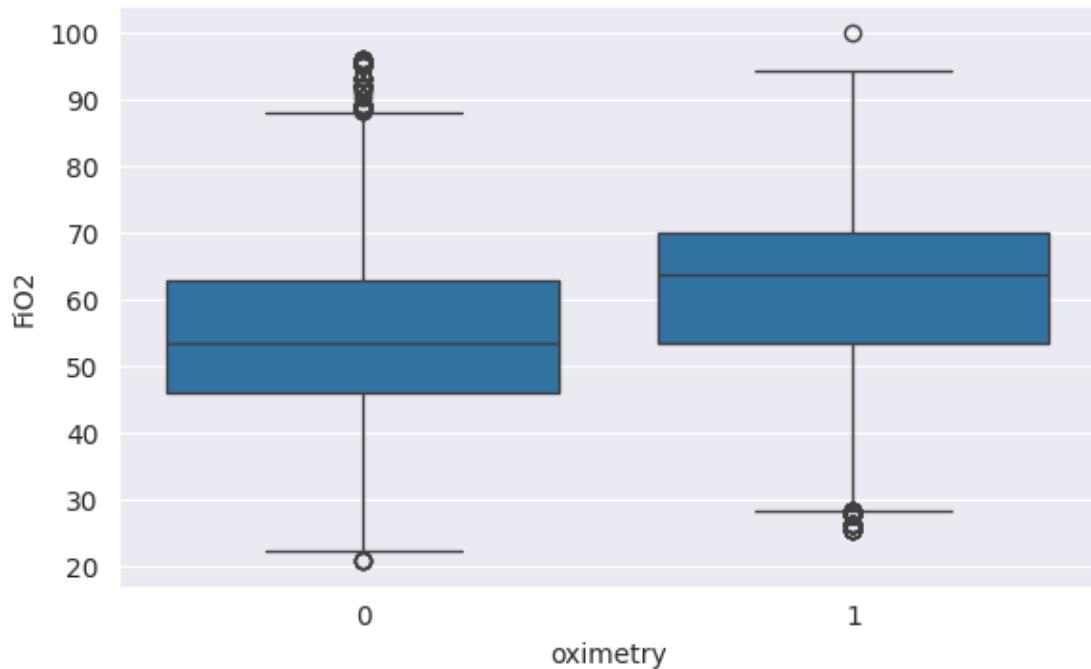
RR vs. oximetry



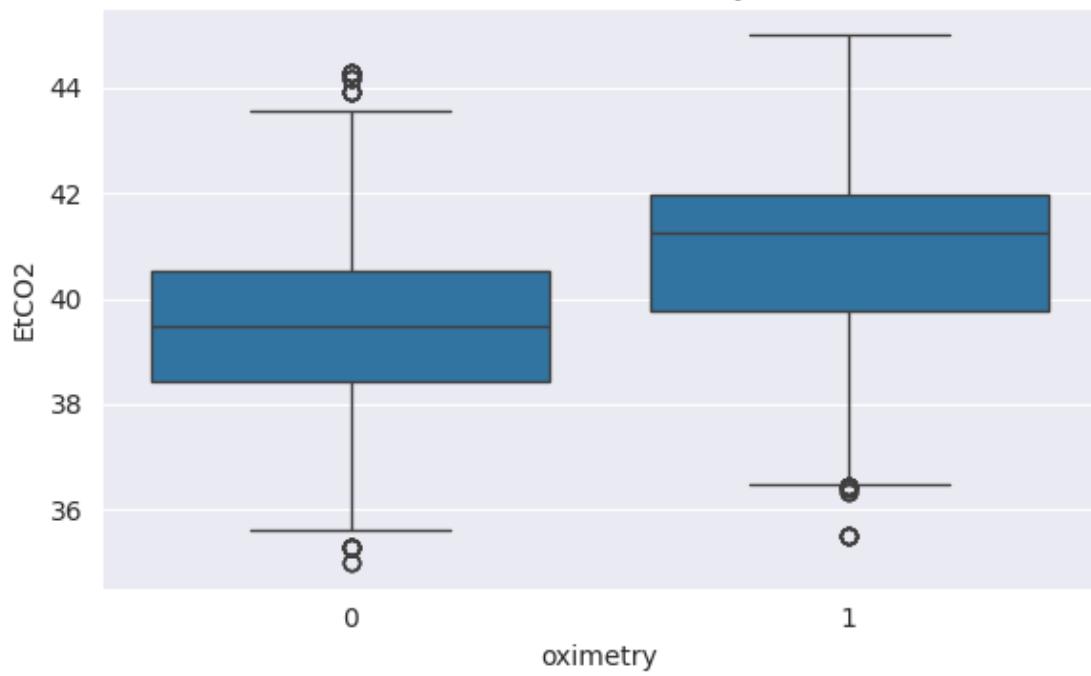
SpO<sub>2</sub> vs. oximetry

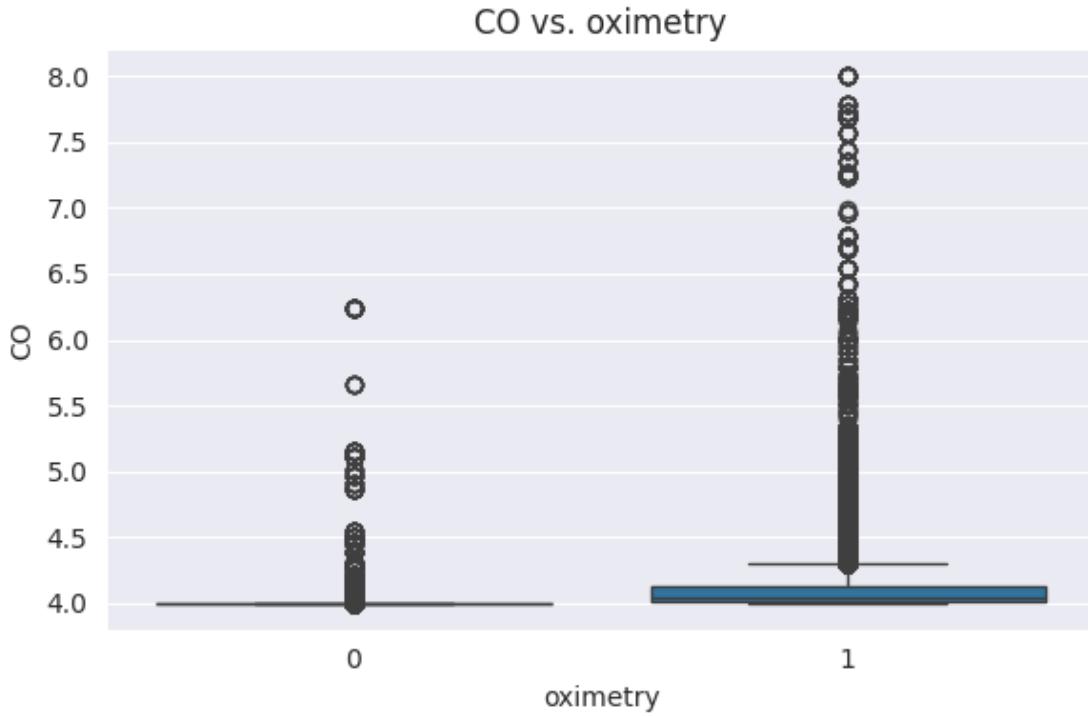


FiO<sub>2</sub> vs. oximetry



EtCO<sub>2</sub> vs. oximetry





[262]: # HR (Heart Rate) showed several extreme values . Very high for oximetry = 0 and very low for oximetry = 1.  
#These outliers can affect model accuracy, so in the next phase we will clean them using the IQR method and winsorization (5th-95th percentile) to keep only realistic heart rate values.

#The CO variable shows a highly right-skewed distribution with a long upper tail, indicating a large number of potential outliers compared to other attributes.

#These extreme values are likely to distort the model, so CO will require normalization or outlier treatment (IQR filtering or winsorization).

```
# Calculate 5th and 95th percentiles
low, high = df['HR'].quantile([0.05, 0.95])

# before winsorization
before = ((df['HR'] < low) | (df['HR'] > high)).sum()

# apply wins.
df_win = df.copy()
df_win['HR'] = df_win['HR'].clip(lower=low, upper=high)

# after winsorization
```

```

after = ((df_win['HR'] < low) | (df_win['HR'] > high)).sum()

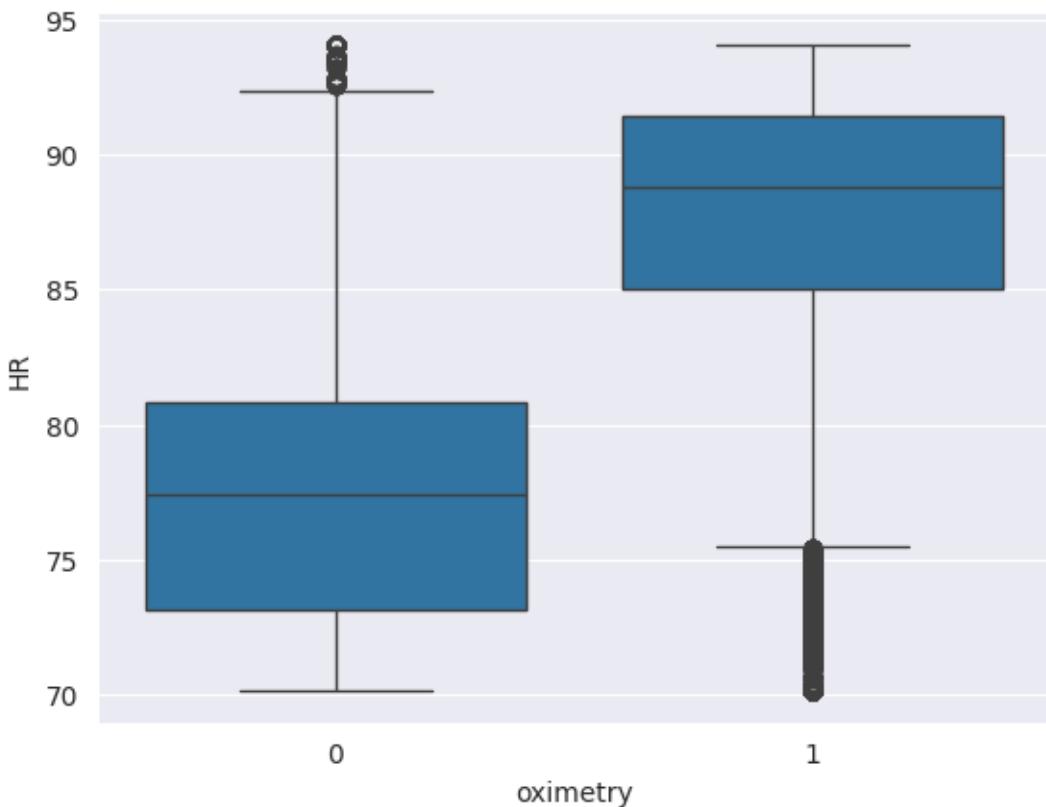
print(f"Before winsorization: {before} outliers")
print(f"After winsorization: {after} outliers")

sns.boxplot(data = df_win, x='oximetry', y='HR')

```

Before winsorization: 6696 outliers  
After winsorization: 0 outliers

[263]: <Axes: xlabel='oximetry', ylabel='HR'>



Now I can see that there is not such a values that are higher or lower to quintile range from 0.05 to 0.95 that's why we did not get any after winsortization value on the other hand we caught some outliers with the IQR method on df[“CO”] attribute because he included many abnormal values

[264]: # IQR bounders  
Q1 = df['CO'].quantile(0.25)  
Q3 = df['CO'].quantile(0.75)  
IQR = Q3 - Q1  
lower\_bound = Q1 - 1.5 \* IQR

```

upper_bound = Q3 + 1.5 * IQR

print(f"CO bounds (IQR): {lower_bound:.2f} - {upper_bound:.2f}")

# delete values out of range
df_iqr = df[(df['CO'] >= lower_bound) & (df['CO'] <= upper_bound)]

print(f"Removed {len(df) - len(df_iqr)} rows")
print(len(df_iqr))

```

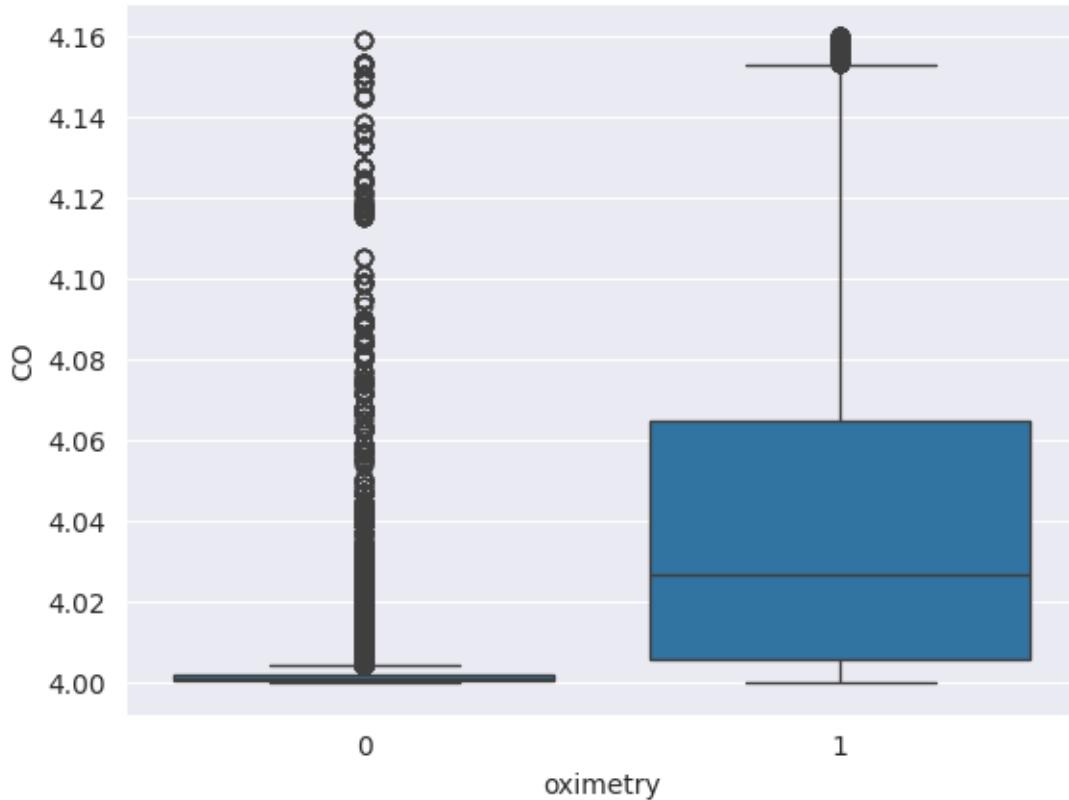
CO bounds (IQR): 3.91 - 4.16

Removed 8460 rows

58513

[265]: sns.boxplot(data=df\_iqr, x='oximetry', y='CO')

[265]: <Axes: xlabel='oximetry', ylabel='CO'>



On the graph above we can see extreme difference on outlier values because the previous graph had values close to 9 and this has maximum values 4.16

```
[266]: # all number columns except oximetry
num_cols = df.select_dtypes(include='number').columns.drop('oximetry', errors='ignore')

#add columns if in range of upper or lower bounder
iqr_summary = []
for col in num_cols:
    q1, q3 = df[col].quantile([0.25, 0.75])
    iqr = q3 - q1
    lower, upper = q1 - 1.5 * iqr, q3 + 1.5 * iqr
    outliers = ((df[col] < lower) | (df[col] > upper)).sum()
    iqr_summary.append([col, outliers, round(outliers / len(df) * 100, 2)])

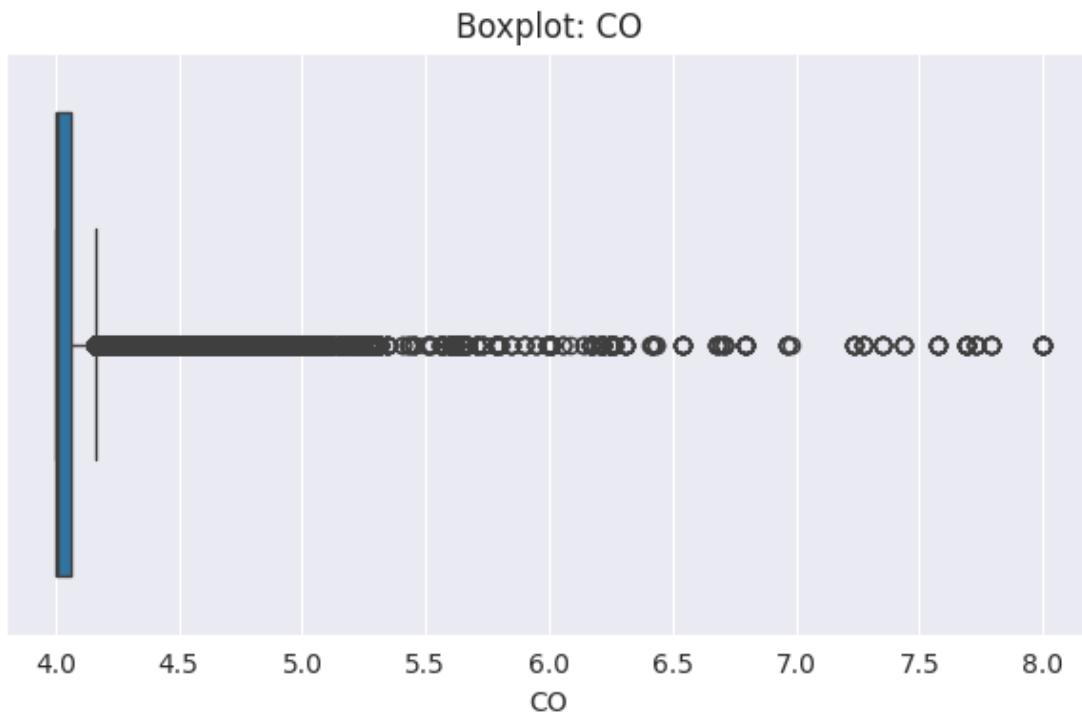
# Display as a small DataFrame
iqr_df = pd.DataFrame(iqr_summary, columns=['Column', 'Outliers', 'Percent'])
iqr_df.sort_values('Percent', ascending=False, inplace=True)
iqr_df
```

	Column	Outliers	Percent
13	CO	8460	12.63
2	PI	785	1.17
6	PRV	633	0.95
20	latitude	627	0.94
14	Blood Flow Index	602	0.90
0	SpO2	587	0.88
7	BP	560	0.84
15	PPG waveform features	549	0.82
21	longitude	544	0.81
17	Respiratory effort	468	0.70
9	Motion/Activity index	472	0.70
12	SV	439	0.66
8	Skin Temperature	432	0.65
11	Hb level	352	0.53
16	Signal Quality Index	351	0.52
10	PVI	344	0.51
5	FiO2	5	0.01
4	EtCO2	0	0.00
1	HR	0	0.00
3	RR	0	0.00
19	SNR	0	0.00
18	O2 extraction ratio	0	0.00
22	station_ID	0	0.00
23	user_id	0	0.00

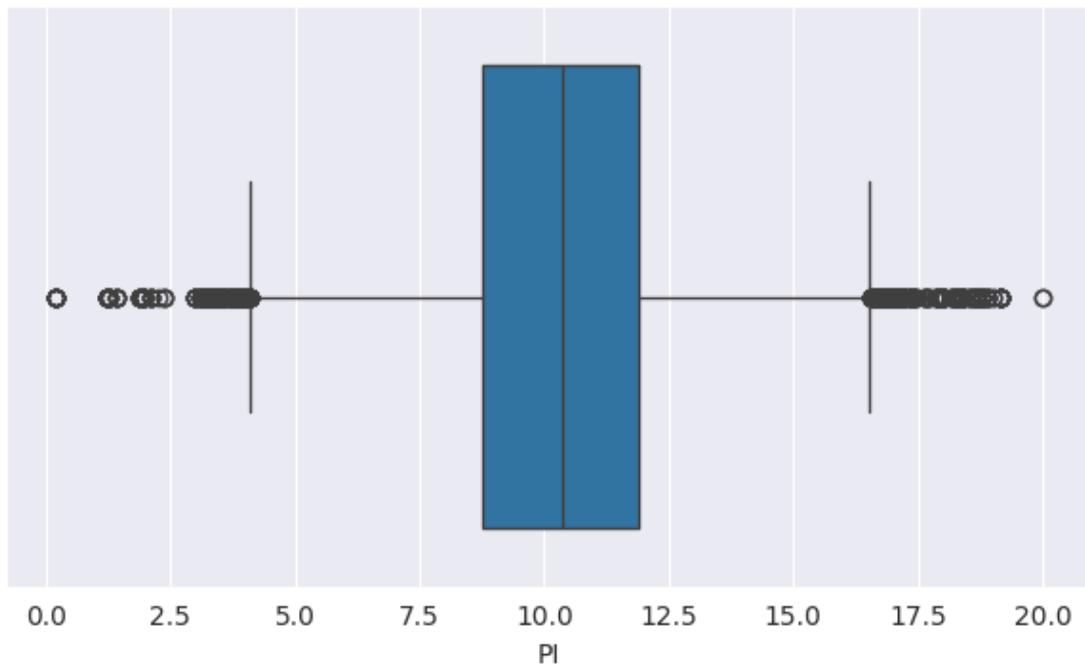
It's bad practice to analyze a attribute one by one so I created a list of numeric values that I can analyze at once. I calculated they're IQR based on dividing they're quantile (0.25 and 0.75). If any value is less than lower value ( $<0.25$ ) that it is added to list of outliers this process also include calculating if value is greater than upper bounder ( $>0.75$ ).

```
[267]: suspects = iqr_df[ iqr_df['Outliers'] > 0 ]['Column'].head(6).tolist()

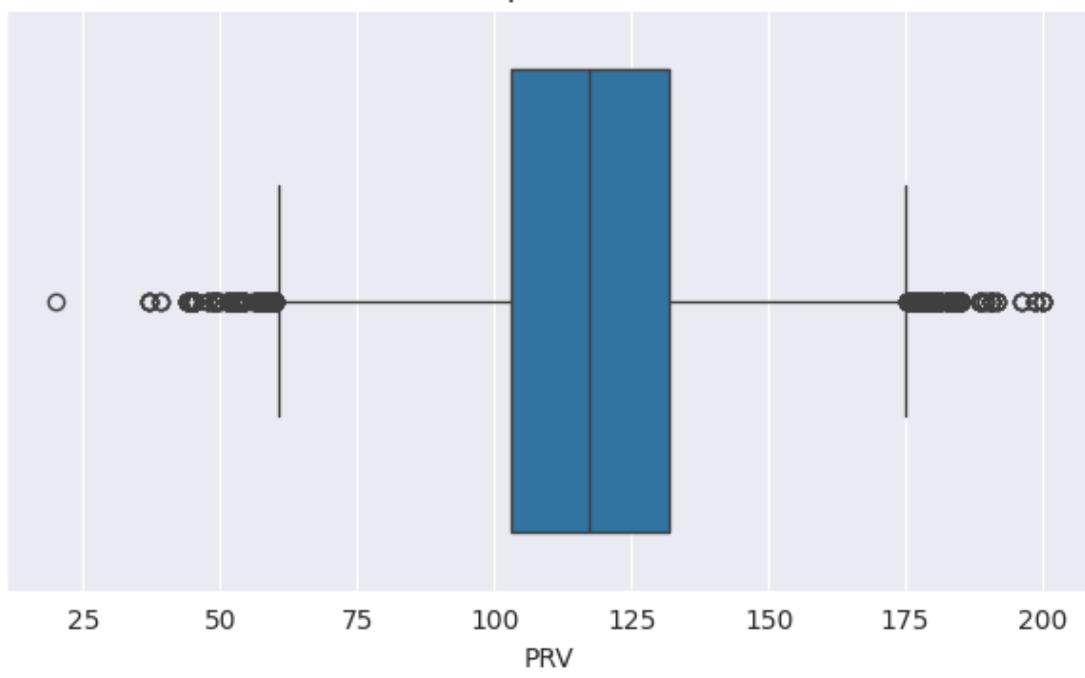
for col in suspects:
    plt.figure(figsize=(6,4))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot: {col}')
    plt.tight_layout()
    plt.show()
```



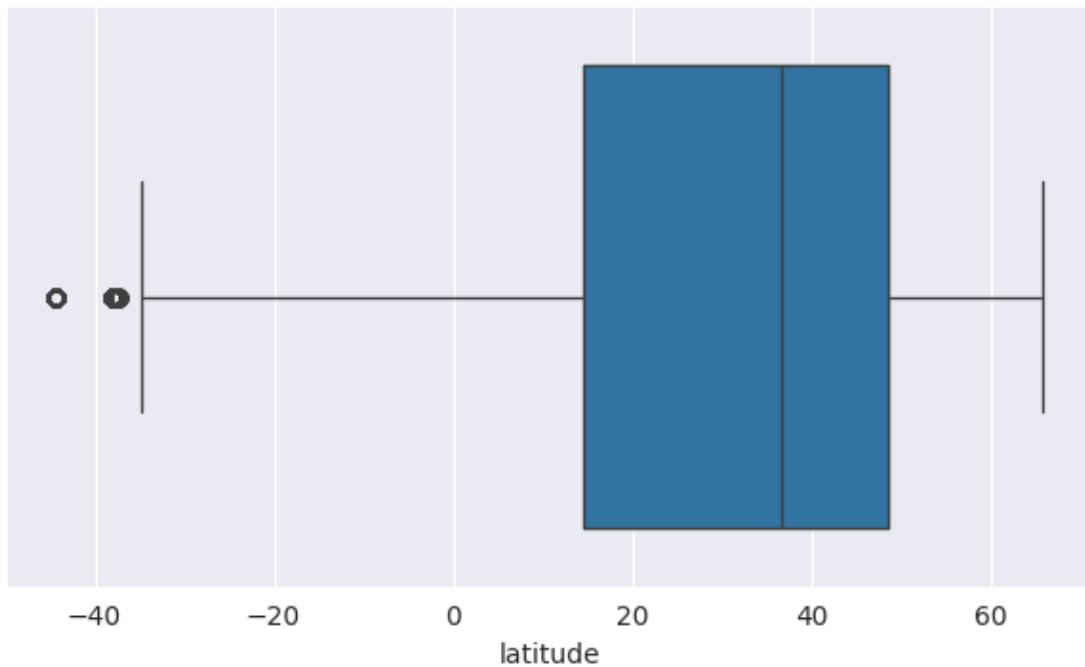
Boxplot: PI



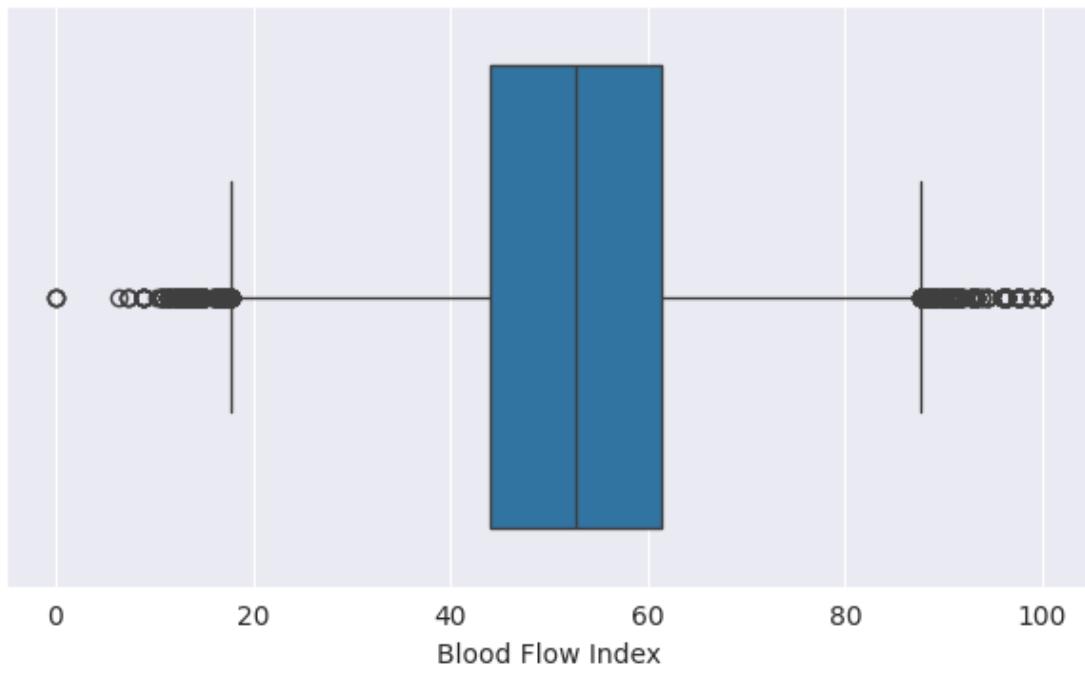
Boxplot: PRV

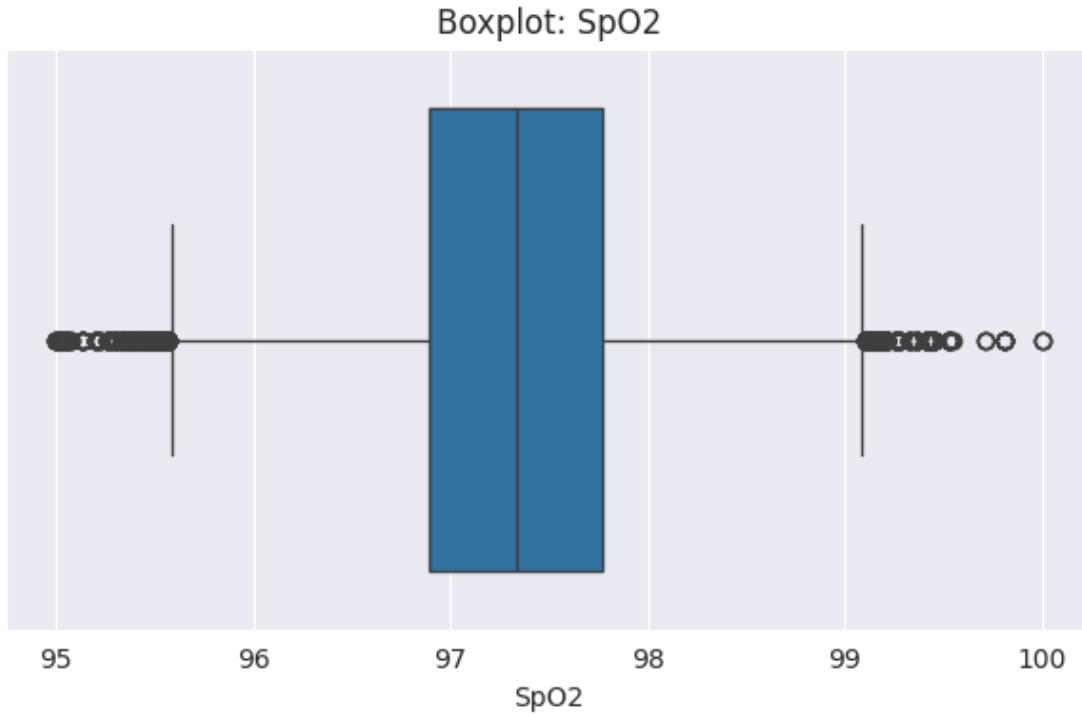


Boxplot: latitude



Boxplot: Blood Flow Index





### 3.3 1.3 Formulácia a štatistické overenie hypotéz o dátach

#### 3.3.1 Zle hypotezy

**Hypotéza 1 – rozdiel v tepovej frekvencii podľa stavu saturácie**

**H (Null Hypothesis):** The mean heart rate (HR) is the same for both groups — patients with oximetry = 0 and oximetry = 1.

**H (Alternative Hypothesis):** The mean heart rate (HR) differs between the groups — patients with oximetry = 0 and oximetry = 1.

```
[268]: const_shapiro = 5000
# I will create 2 datasets hr00 (HE and oximetry = 0) and hr01 (HE column and oximetry = 1)
hr0 = df[df['oximetry'] == 0]['HR']
hr1 = df[df['oximetry'] == 1]['HR']

sample_hr0 = hr0.sample(const_shapiro, random_state=42)
sample_hr1 = hr1.sample(const_shapiro, random_state=42)

print(shapiro(sample_hr0))
print(shapiro(sample_hr1))
```

```
ShapiroResult(statistic=np.float64(0.9964622127654921),  
pvalue=np.float64(1.50860565376863e-09))  
ShapiroResult(statistic=np.float64(0.9381229280127523),  
pvalue=np.float64(1.7803348598149635e-41))
```

```
[269]: #we can see that the pvalues from both samples are lower than 0.05 which  
    ↳ indicates to not normal distributed data  
#based on that we need to perform Mann-whitney U test not basic T-test that  
    ↳ need normal distribution  
  
stat, p = stats.mannwhitneyu(sample_hr0, sample_hr1)  
print(f"Mann-Whitney U-test p-value: {p:.3e}")  
  
# the value is super close to 0 that means that we denied H0 => there is a  
    ↳ statistically significant difference  
# in heart rate (HR) between the groups (oximetry = 0 and oximetry = 1).
```

Mann-Whitney U-test p-value: 0.000e+00

## Hypothesis 2 – Difference in EtCO based on oxygen saturation (oximetry)

**H (Null Hypothesis):** The mean EtCO level is the same for both groups — patients with oximetry = 0 and oximetry = 1.

**H (Alternative Hypothesis):** The mean EtCO level differs between the groups — patients with oximetry = 0 and oximetry = 1.

```
[270]: df.columns
```

```
[270]: Index(['SpO2', 'HR', 'PI', 'RR', 'EtCO2', 'FiO2', 'PRV', 'BP',  
    'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',  
    'CO', 'Blood Flow Index', 'PPG waveform features',  
    'Signal Quality Index', 'Respiratory effort', 'O2 extraction ratio',  
    'SNR', 'oximetry', 'latitude', 'longitude', 'station_ID', 'QoS', 'code',  
    'user_id', 'blood_group'],  
    dtype='object')
```

```
[271]: etco0 = df[df['oximetry'] == 0]['EtCO2']  
etco1 = df[df['oximetry'] == 1]['EtCO2']  
  
#sample for max 5000  
sample_etco0 = etco0.sample(5000, random_state=42)  
sample_etco1 = etco1.sample(5000, random_state=42)  
  
# normality check  
print(shapiro(sample_etco0))  
print(shapiro(sample_etco1))
```

```
# if both p > 0.05 → data are normally distributed
# then perform Student's T-test
stat, p = stats.mannwhitneyu(etco0, etco1)
print(f"U-test p-value: {p:.3e}")
```

```
ShapiroResult(statistic=np.float64(0.9938690077488066),
pvalue=np.float64(8.316180710196192e-14))
ShapiroResult(statistic=np.float64(0.9191799162237433),
pvalue=np.float64(1.0762260080747649e-45))
U-test p-value: 0.000e+00
```

there is a statistically significant difference in mean EtCO between patients with oximetry = 0 and oximetry = 1.

### 3.3.2 A) Hypothesis 1 – Difference in SPO based on oxygen saturation (oximetry)

**H (Null Hypothesis):** The mean SPO level is the same for both groups — patients with oximetry = 0 and oximetry = 1.

**H (Alternative Hypothesis):** The mean SPO level differs between the groups — patients with oximetry = 0 and oximetry = 1.

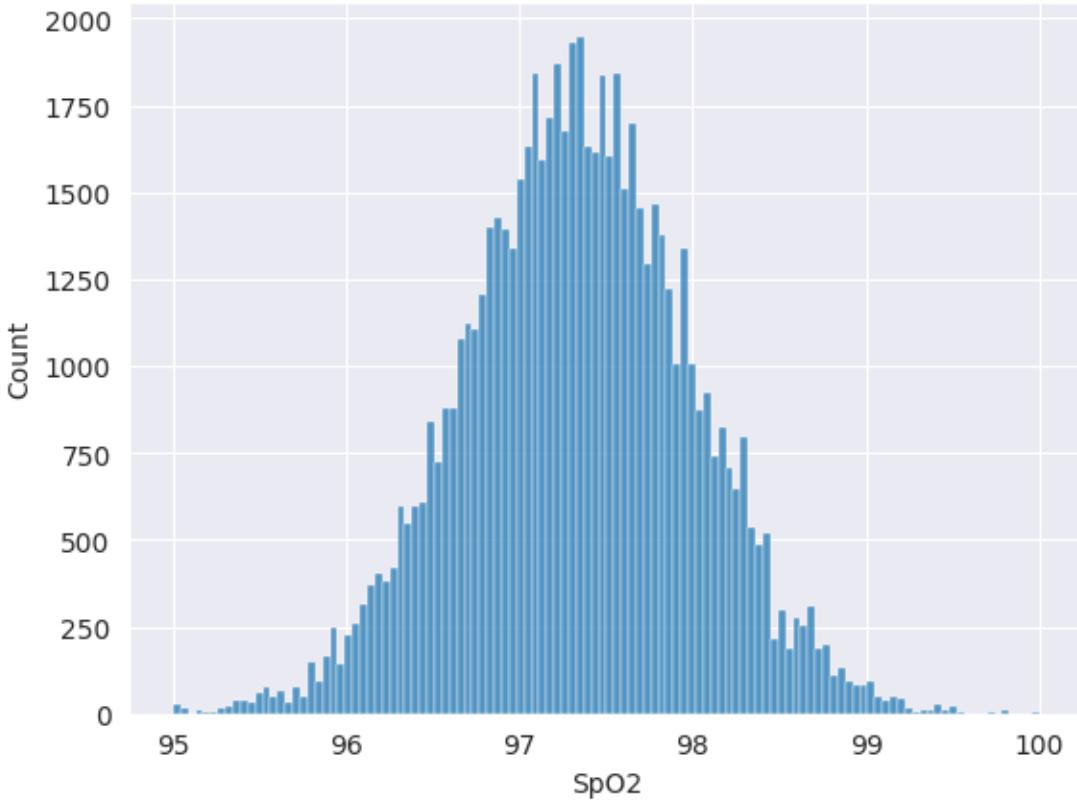
```
[272]: #division of the data into oximetry == 1 and oximetry == 0 groups
g1 = df[df['oximetry'] == 1]['SpO2']
g0 = df[df['oximetry'] == 0]['SpO2']

#doing the shapiro test -> pvalue for both values
p1 = shapiro(g1).pvalue
p0 = shapiro(g0).pvalue
print('shapiro:', p1, p0)
# even if the p values do not indicate normal division of the data, we will
# continue with levenne and ttest since from the histogram we can see that the
# data is indeed rather normally distributed, this error is probably caused by
# the table creation
```

```
shapiro: 9.472557968212563e-07 0.0018560650823116259
C:\Users\mudry\AppData\Local\Programs\Python\Python313\Lib\site-
packages\scipy\stats\_axis_nan_policy.py:579: UserWarning: scipy.stats.shapiro:
For N > 5000, computed p-value may not be accurate. Current N is 40029.
    res = hypotest_fun_out(*samples, **kwds)
C:\Users\mudry\AppData\Local\Programs\Python\Python313\Lib\site-
packages\scipy\stats\_axis_nan_policy.py:579: UserWarning: scipy.stats.shapiro:
For N > 5000, computed p-value may not be accurate. Current N is 26944.
    res = hypotest_fun_out(*samples, **kwds)
```

```
[273]: sns.histplot(df['SpO2'])
```

```
[273]: <Axes: xlabel='SpO2', ylabel='Count'>
```



[274]: #due to the findings in the previous cell, we found out the merging of the  
 ↪tables reshaped the distribution of the data into not so normally  
 ↪distributed, this however could be fixed by performing the tests on the  
 ↪observation\_df (non merged table).

```

g1 = observation_df[observation_df['oximetry'] == 1]['SpO2']
g0 = observation_df[observation_df['oximetry'] == 0]['SpO2']

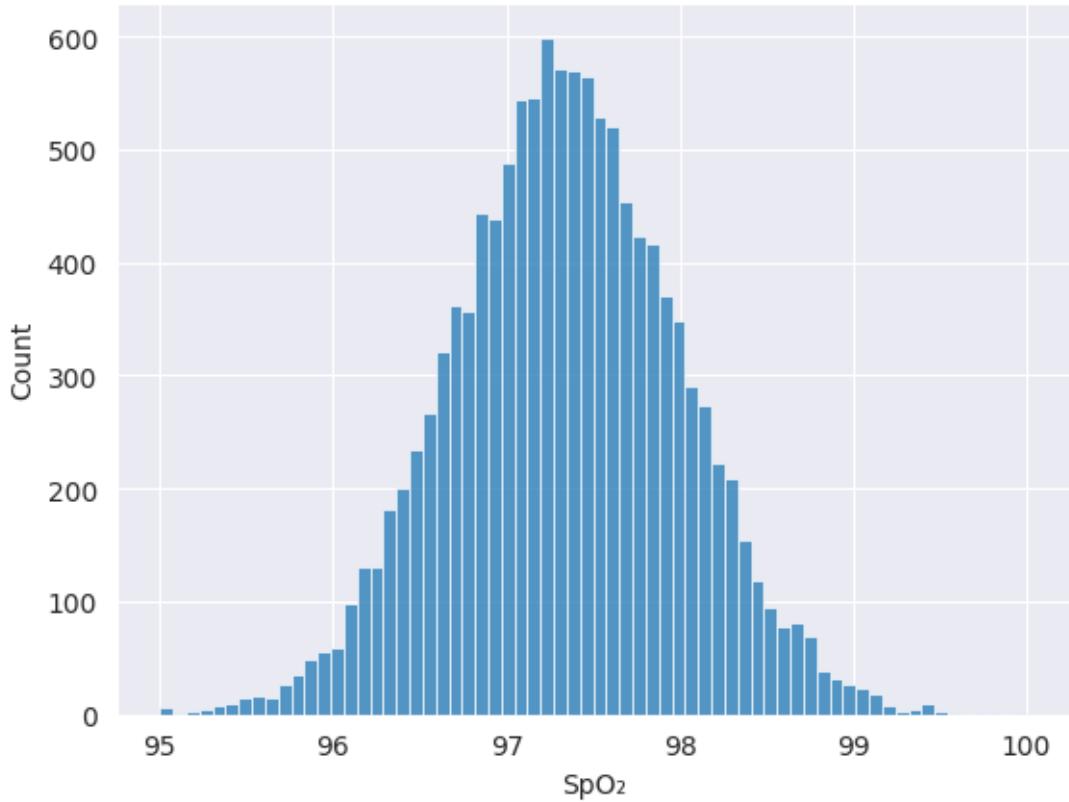
#we need to set sample to max 5000 so the data are in range of the recommended
↪sample value
sample_g1 = g1.sample(3000, random_state=42)
sample_g0 = g0.sample(3000, random_state=42)

#doing the shapiro test -> pvalue for both values
p1 = shapiro(sample_g1).pvalue
p0 = shapiro(sample_g0).pvalue
print('shapiro:', p1, p0)
#after performing the shapiro test we can see that the p values from both the
↪groups are larger than 0.05 which proves that the normal distribution is
↪present which leads us to perform the levene and t test
    
```

shapiro: 0.10880973933264598 0.6169824476877354

```
[275]: sns.histplot(observation_df['SpO'])
```

```
[275]: <Axes: xlabel='SpO', ylabel='Count'>
```

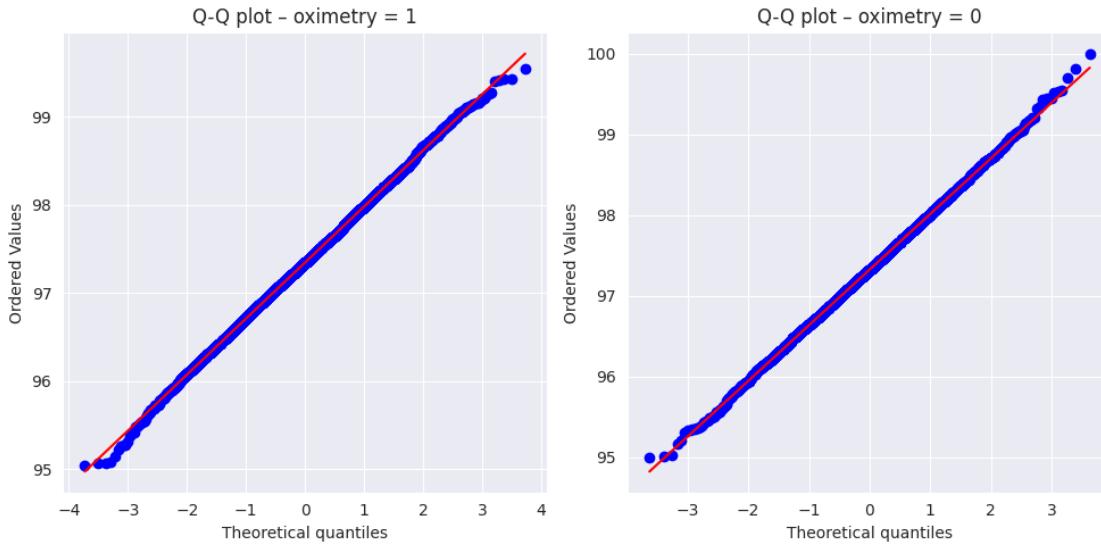


```
[276]: #lets verify the distributions of both the subsets via qqplot
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
```

```
stats.probplot(g1, dist="norm", plot=ax[0])
ax[0].set_title("Q-Q plot - oximetry = 1")
```

```
stats.probplot(g0, dist="norm", plot=ax[1])
ax[1].set_title("Q-Q plot - oximetry = 0")
```

```
plt.tight_layout()
plt.show()
#they indeed are
```



```
[277]: #levene tests, this whole block is executed since the data are normally distributed
p_lev = stats.levene(sample_g1,sample_g0).pvalue
print('levene:', p_lev)

if p_lev > 0.05:
    # since the variance is the same we can perfrom the usual t-test
    t, p = stats.ttest_ind(sample_g1, sample_g0)
    print('t-test', t, p)
else:
    # variances differ, use Welch's t-test - we know this wasnt in our class notebooks, but we needed to find out if the two groups differ even though they have different variance, so we found this method
    t, p = stats.ttest_ind(sample_g1, sample_g0, equal_var=False)
    print('Welch t-test (unequal variances):', t, p)
```

levene: 1.253188761462382e-05

Welch t-test (unequal variances): 0.09803193034222664 0.9219102349142181

```
[278]: # Both groups are normally distributed (Shapiro > 0.05), but their variances differ (Levene < 0.05). The Mann-Whitney U test (p = 0.0984 > 0.05) shows no statistically significant difference in SpO levels between oximetry = 0 and oximetry = 1.
```

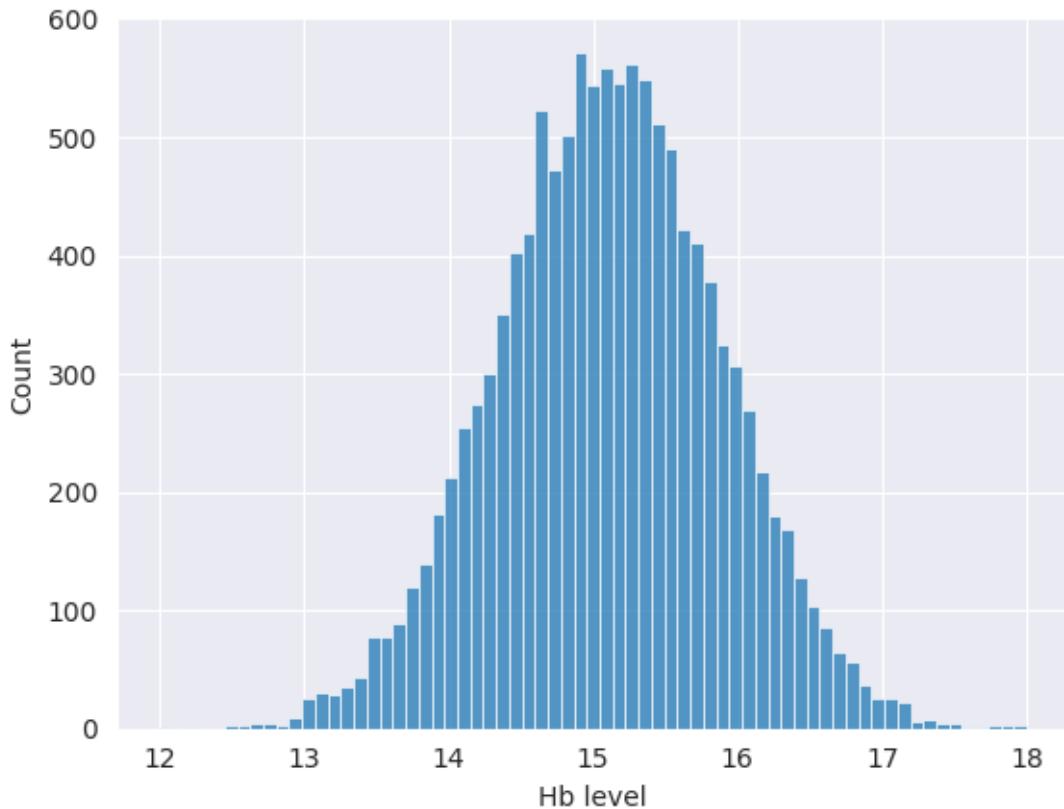
### 3.3.3 A) Hypothesis 2 - Comparison of Hemoglobin (Hb) Levels Between Oximetry Groups

**H (Null Hypothesis):** The mean hemoglobin (Hb) level is the same for both groups — patients with oximetry = 0 and oximetry = 1.

**H (Alternative Hypothesis):** The mean hemoglobin (Hb) level differs between the groups — patients with oximetry = 0 and oximetry = 1.

```
[279]: sns.histplot(observation_df['Hb level'])
```

```
[279]: <Axes: xlabel='Hb level', ylabel='Count'>
```



```
[280]: hb0 = observation_df[observation_df['oximetry'] == 0]['Hb level']
hb1 = observation_df[observation_df['oximetry'] == 1]['Hb level']

#we need to set sample to max 5000 so the data are in range of the recommended
#sample value
sample_hb0= hb0.sample(2000, random_state=42)
sample_hb1 = hb1.sample(2000, random_state=42)

#doing the shapiro test -> pvalue for both values
p0 = shapiro(sample_hb0).pvalue
p1 = shapiro(sample_hb1).pvalue
print('shapiro:\n{} \n{}'.format(p0,p1))
```

```
shapiro:
p0:0.11878125043102106
```

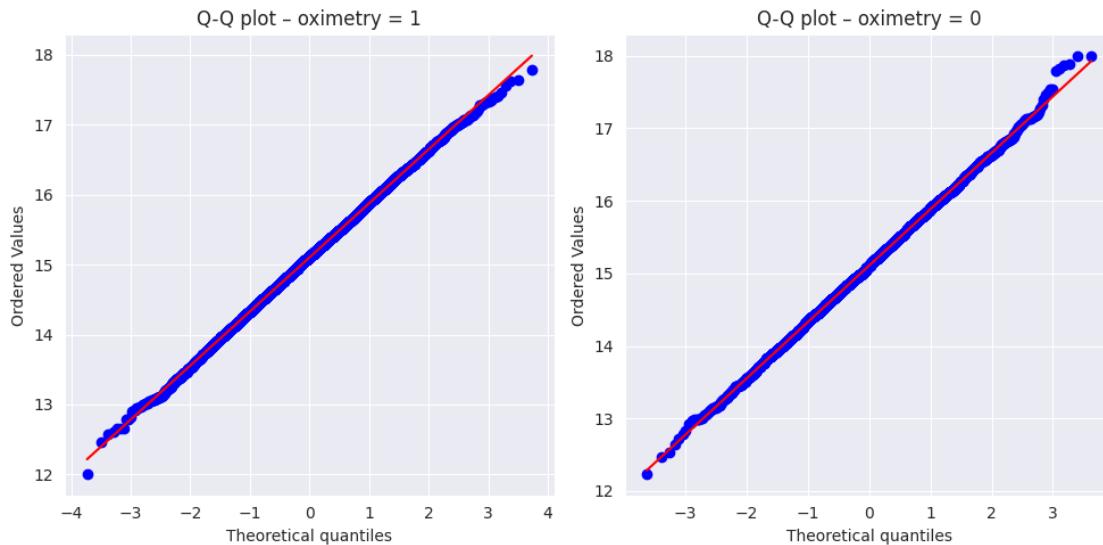
```
p1:0.85391674151546
```

```
[281]: #lets verify the distributions of both the subsets via qqplot
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

stats.probplot(hb1, dist="norm", plot=ax[0])
ax[0].set_title("Q-Q plot - oximetry = 1")

stats.probplot(hb0, dist="norm", plot=ax[1])
ax[1].set_title("Q-Q plot - oximetry = 0")

plt.tight_layout()
plt.show()
#they indeed are
```



```
[282]: #same as H0
p_lev = stats.levene(sample_hb1,sample_hb0).pvalue
print('levene:', p_lev)

if p_lev > 0.05:
    # this was explained in H0
    t, p = stats.ttest_ind(sample_hb1, sample_hb0)
    print('t-test ', t, p)
else:
    # this was explained in H0
    t, p = stats.ttest_ind(sample_hb1, sample_hb0, equal_var=False)
    print('Welch t-test (unequal variances):', t, p)
```

```
levene: 0.6624508281537517
```

```
t-test -1.588383168023951 0.11227879476272101
```

```
[283]: # Both groups are normally distributed (Shapiro > 0.05), with their variance being very similar as well. Students t test shows that the difference of the means was not statistically significant.
```

### 3.3.4 B) Overte či Vaše štatistické testy majú dostatok podpory z dát, teda či majú dostatočne silnú štatistickú silu.

```
[284]: # SpO2 attribute
y0 = sample_g0.values
y1 = sample_g1.values

# base stats
m0, m1 = np.mean(y0), np.mean(y1)
s0, s1 = np.std(y0, ddof=1), np.std(y1, ddof=1)
n0, n1 = len(y0), len(y1)

# CI
se = np.sqrt(s0**2/n0 + s1**2/n1)
df_temp = (s0**2/n0 + s1**2/n1)**2 / ((s0**4)/((n0**2)*(n0-1)) + (s1**4)/((n1**2)*(n1-1)))
ci = stats.t.interval(0.95, df_temp, loc=(m1 - m0), scale=se)
print(f"95% CI for delta means: {ci[0]:.3f} to {ci[1]:.3f}")
#very little difference -> strong statistical power
```

95% CI for delta means: -0.032 to 0.035

```
[285]: #Cohen
sp = np.sqrt(((n0 - 1)*s0**2 + (n1 - 1)*s1**2) / (n0 + n1 - 2))
d = (m1 - m0) / sp
print(f"Cohen's d: {d:.3f}")
#almost 0 difference -> strong statistical power
```

Cohen's d: 0.003

CI ranging from -0.032 to 0.035 and cohens'd being 0.003 shows us that there is strong statistical power and that the tests have enough support from the data to be legitimate

```
[286]: # HB level attribute
y0 = sample_hb0.values
y1 = sample_hb1.values

# base stats
m0, m1 = np.mean(y0), np.mean(y1)
s0, s1 = np.std(y0, ddof=1), np.std(y1, ddof=1)
n0, n1 = len(y0), len(y1)

# CI
```

```

se = np.sqrt(s0**2/n0 + s1**2/n1)
df_temp = (s0**2/n0 + s1**2/n1)**2 / ((s0**4)/((n0**2)*(n0-1)) + (s1**4)/
    ((n1**2)*(n1-1)))
ci = stats.t.interval(0.95, df_temp, loc=(m1 - m0), scale=se)
print(f"95% CI for delta means: {ci[0]:.3f} to {ci[1]:.3f}")
#same result as with SP02

```

95% CI for delta means: -0.087 to 0.009

[287]: #Cohen

```

sp = np.sqrt(((n0 - 1)*s0**2 + (n1 - 1)*s1**2) / (n0 + n1 - 2))
d = (m1 - m0) / sp
print(f"Cohen's d: {d:.3f}")
#almost 0 difference -> strong statistical power

```

Cohen's d: -0.050

CI ranging from -0.087 to 0.009 and cohens'd being -0.050 shows us that there is strong statistical power and that the tests have enough support from the data to be legitimate

## 4 Fáza 2

### 4.1 2.1 Realizácia predspracovania dát

#### 4.1.1 A)

Before doing the magic with splitting data we need to set them up by filtering only attributes we need and are usefull in future analysis

[288]: df\_2 = df.copy()

```

# we want to work with the data that has some connectivity to "health"
#specification not anything like position ...
drop_cols = [c for c in ['user_id', 'station_ID', 'latitude', 'longitude'] if c in
    df_2.columns]
df_2 = df_2.drop(columns=drop_cols)

df_2 = df_2.drop_duplicates()

```

[289]: target = "oximetry"

```

#split features and target
X = df_2.drop(columns=[target])
y = df_2[target]

```

[290]: X\_train, X\_test, y\_train, y\_test = train\_test\_split(
 X,
 y,
 test\_size=0.2,

```

    random_state=42,
    stratify=y
)

print("Train dataframe:", X_train.shape)
print("Test dataframe:", X_test.shape)
print("Oximetry=1/total for train:", y_train.mean().round(3))
print("Oximetry=1/total for test:", y_test.mean().round(3))
#we can see the division of the original df into train and test subdatasets, we
↳can also see the stratification focused on the oximetry attribute worked

```

Train dataframe: (41501, 23)  
Test dataframe: (10376, 23)  
Oximetry=1/total for train: 0.597  
Oximetry=1/total for test: 0.597

#### 4.1.2 B)

[291]: X\_test.head()

	Sp02	HR	PI	RR	EtCO2	Fi02	\
49445	96.601411	80.344377	8.884552	15.344047	38.825321	55.089020	
30741	97.993688	83.684406	11.323195	15.208839	41.193087	63.886566	
52230	96.805991	89.893353	7.491702	14.655980	42.801724	62.273550	
50948	97.293984	77.630562	11.833415	16.148954	42.182432	65.501698	
58059	97.263038	84.972915	10.198301	14.049168	39.057607	59.162991	

	PRV	BP	Skin Temperature	Motion/Activity index	...	\
49445	84.643454	105.944792	34.263087	10.094809	...	
30741	111.221376	106.803949	34.890998	9.349818	...	
52230	108.099974	101.153579	36.156039	10.064168	...	
50948	130.012206	100.104509	35.036113	8.346491	...	
58059	129.506948	99.689616	36.135584	9.809290	...	

	C0	Blood Flow Index	PPG waveform features	\
49445	4.001692	67.333233	38.870747	
30741	4.005741	42.993997	33.119117	
52230	4.067662	50.017832	37.370695	
50948	4.000921	49.619197	60.764507	
58059	4.009634	60.365075	58.944290	

	Signal Quality Index	Respiratory effort	O2 extraction ratio	\
49445	76.138614	40.726619	0.268550	
30741	24.206459	33.059317	0.256908	
52230	59.558249	44.561747	0.210238	
50948	42.742079	46.305055	0.219876	
58059	55.162309	69.790015	0.245487	

	SNR	QoS	code	blood_group
49445	25.952849	good	AM	AB+
30741	31.628228	excellent	PH	O-
52230	28.975340	excellent	AU	O+
50948	29.580558	maintenance	FR	O+
58059	22.505126	excellent	IN	A+

[5 rows x 23 columns]

[292]: X\_train.columns

```
[292]: Index(['SpO2', 'HR', 'PI', 'RR', 'EtCO2', 'FiO2', 'PRV', 'BP',
   'Skin Temperature', 'Motion/Activity index', 'PVI', 'Hb level', 'SV',
   'CO', 'Blood Flow Index', 'PPG waveform features',
   'Signal Quality Index', 'Respiratory effort', 'O2 extraction ratio',
   'SNR', 'QoS', 'code', 'blood_group'],
  dtype='object')
```

[293]: `print(X_train.dtypes)`

*#every attribute is either category or float64, however this was not the case*

- ↳ *when we started working with the dataset, QoS, code and blood group were*
- ↳ *changed into category datatype in 1.2 A), 3rd cell -> we now have to turn*
- ↳ *category attributes into numeric attributes*

SpO2	float64
HR	float64
PI	float64
RR	float64
EtCO2	float64
FiO2	float64
PRV	float64
BP	float64
Skin Temperature	float64
Motion/Activity index	float64
PVI	float64
Hb level	float64
SV	float64
CO	float64
Blood Flow Index	float64
PPG waveform features	float64
Signal Quality Index	float64
Respiratory effort	float64
O2 extraction ratio	float64
SNR	float64
QoS	category
code	category
blood_group	category

```

dtype: object

[294]: X_train['QoS'].unique()

[294]: ['good', 'excellent', 'acceptable', 'maintenance']
Categories (4, object): ['acceptable', 'excellent', 'good', 'maintenance']

[295]: #It makes sense to use ordinal encoding on QoS, since we can set which is the
       ↴best(excellent) and which is the worst(maintenece)
qos_order = CategoricalDtype(
    categories=["maintenance", "acceptable", "good", "excellent"],
    ordered=True
)

X_train["QoS"] = X_train["QoS"].astype(qos_order)

X_train["QoS_ord"] = X_train["QoS"].cat.codes
X_train = X_train.drop(columns=["QoS"])

```

```

[296]: X_train['blood_group'].unique()

[296]: ['A-', 'B+', 'AB+', 'A+', 'AB-', 'O+', 'O-', 'B-']
Categories (8, object): ['A+', 'A-', 'AB+', 'AB-', 'B+', 'B-', 'O+', 'O-']

[297]: X_train = pd.get_dummies(X_train, columns=["blood_group"], prefix="bg")

[298]: X_train['code'].unique()

[298]: ['KR', 'CN', 'BJ', 'AR', 'US', ..., 'EC', 'HK', 'CH', 'MK', 'CF']
Length: 97
Categories (96, object): ['AF', 'AM', 'AO', 'AR', ..., 'VE', 'VN', 'VU', 'ZA']

[299]: #we also remember that the code attribute is the only attribute that has a
       ↴missing value which belonged to Okinawa station, we will simply drop these
       ↴rows
#I did think about imputing the station with a new code for Japan, but its
       ↴really only 100 rows and we cant use an imputing strategy that could be
       ↴later replicated as this is Okinawa only...
X_train = X_train.dropna()

```

```

[300]: #after further analysis we decided to drop code column altogether, as the only
       ↴viable encoding option was frequency encoding but that would eventually make
       ↴no sense...
#We know we should not change te test dataset in this part but this is not
       ↴really a preprocessing method
X_train = X_train.drop(columns=["code"])
X_test = X_test.drop(columns=["code"])

```

```
[301]: print(X_test.dtypes)
#We can see that there are no more non numeric attributes!
```

```
SpO2           float64
HR            float64
PI            float64
RR            float64
EtCO2          float64
FiO2          float64
PRV           float64
BP            float64
Skin Temperature    float64
Motion/Activity index float64
PVI           float64
Hb level       float64
SV             float64
CO             float64
Blood Flow Index    float64
PPG waveform features float64
Signal Quality Index float64
Respiratory effort    float64
O2 extraction ratio   float64
SNR            float64
QoS             category
blood_group     category
dtype: object
```

4.1.3 C)

4.1.4 D)

## 4.2 2.2 Výber atribútov pre strojové učenie

4.2.1 A)

4.2.2 B)

4.2.3 C)

## 4.3 2.3 Replikovateľnosť predspracovania

4.3.1 A)

4.3.2 B)