PTBxl data

November 23, 2021

[1]:

```
import wfdb
import numpy as np
import os
import pandas as pd
import math
import h5py
import matplotlib.pyplot as plt
import re
```

[2]:

```
def print_object_attributes(obj): #https://stackoverflow.com/questions/192109/
 ↪is-there-a-built-in-function-to-print-all-the-current-properties-and-values-of-a
    for attr in dir(obj):
        print("obj.%s = %r" % (attr, getattr(obj, attr)))
```

0.0.1 Read MIT format .dat ecg data files and .hea headers

[3]:

```
#Download first at: https://physionet.org/content/ptbdb/1.0.0/
#BASE_DIR = '/media/julian/Volume/data/ECG/mit-bih-arrhythmia-database-1.0.0/'␣
 ↪#Arrhythmia
BASE_DIR = '/media/julian/Volume/data/ECG/
 ↪ptb-xl-a-large-publicly-available-electrocardiography-dataset-1.0.1/'
def get_file_list(BASE_DIR, relative=True, filter_function=None):
    record_files = []
    #file_endings = ['.dat', '.hea', '.xyz']
    with open(os.path.join(BASE_DIR, 'RECORDS')) as recs:
        record_files = recs.read().splitlines()
    if filter_function:
        record_files = list(filter(filter_function, record_files))
    if not relative:
        record_files = [os.path.join(BASE_DIR, f) for f in record_files]
    return record_files


record_files = get_file_list(BASE_DIR)
print(len(record_files), 'files found')
```

```
43674 files found
```

0.0.2 Extract signal from *.dat files & Read annotations & Read comments

[4]:
```
def read_comment_map_PTB(record_path):
    #print(record_path)
    record = wfdb.rdrecord(record_path)
    comment_map = {}
    for c in record.comments:
        e = c.split(':')
        comment_map[e[0]] = e[1].strip()
    return comment_map
```

[5]:
```
def filter_comment(comment, key):
    c = comment
    if key == 'Reason for admission':
        if 'Cardiomyopathy' in c or 'Heart failure' in c:
            return 'Cardiomyopathy'
        elif 'n/a' in k or 'Palpitation' in k:
            return 'Miscellaneous'
        elif 'angina' in k:
            new_comments['Angina'] = comments[k]
        else:
            new_comments[k] = comments[k]
```

[6]:
```
def read_comment(record_path):
    record = wfdb.rdrecord(record_path)
    return record.comments
```

[7]:
```
def read_header(record_path):
    record = wfdb.rdheader(record_path, rd_segments=True)
    return record.comments
```

[8]:
```
def read_signal(record_path, physical=True):
    #print(record_path)
    record = wfdb.rdrecord(record_path, physical=physical)
    #print_object_attributes(record)
    if physical:
        data = record.p_signal
    else:
        data = record.d_signal
    return data
```

[9]:
```
def read_annotation(record_path, physical=True):
    try:
        annotation = wfdb.rdann(record_path, 'hea',␣
 ↪return_label_elements=['symbol', 'label_store', 'description'])
        #print(record_path)
        #print('sample:', annotation.sample, 'symbol', annotation.symbol,␣
↪'contained labels', annotation.description)
        return (annotation.sample, annotation.symbol, annotation.label_store,␣
↪annotation.description)
    except ValueError as ve:
        print(record_path, ' annotation read failed:', ve)
        return None
```

Save all signals and attributes in file_data (also note how many had functioning annotations)

[10]:

```
def read_ptbxl_database():
    csvfile = os.path.join(BASE_DIR, 'ptbxl_database.csv')
    dataframe = pd.read_csv(csvfile)
    return dataframe

def read_ptbxl_scp_statements():
    csvfile = os.path.join(BASE_DIR, 'scp_statements.csv')
    dataframe = pd.read_csv(csvfile)
    return dataframe

def train_test_split(record_files_relative):
    df = read_ptbxl_csv()
    train, val, test = [], [], []
    for rf in record_files_relative:
        temp = rf.replace('resampled', '') #resampled file contains 'resampled'␣
↪but csv not
        row = df.loc[(df['filename_hr'].str.contains(temp)) |␣
↪(df['filename_lr'].str.contains(temp))]
        if len(row) < 1:
            print('no row found containing file', rf)
        fold = row['strat_fold'].values[0]
        if fold <= 8: #https://physionet.org/content/ptb-xl/1.0.1/␣
↪#Cross-validation Folds
            train.append(rf)
        elif fold == 9:
            val.append(rf)
        elif fold == 10:
            test.append(rf)
        else:
            print('found unknown strat fold number', fold)
        print("final split: train %d; validation %d; test %d" % (len(train),␣
↪len(val), len(test)))
    return train, val, test

def read_label(ptbxl_database_dataframe, spc_codes_dataframe, rf,␣
↪likelihood_threshold=0.0):
    df = ptbxl_database_dataframe
    spc_df = spc_codes_dataframe
    temp = rf.replace('resampled', '') #resampled file contains 'resampled' but␣
↪csv not
    row = df.loc[(df['filename_hr'].str.contains(temp)) | (df['filename_lr'].
↪str.contains(temp))]
    if len(row) < 1:
        print(filename, 'not found in dataframe')
        return
    code = row['scp_codes'].values[0]
    labels = [(re.sub(r'\W+', '', c.split(':')[0]), c.split(':')[1].
↪replace('}', '').strip()) for c in code.split(',')]
    diagnostic_classes = []
    for l, p in labels:
        if float(p) > likelihood_threshold:
            scp_row = spc_df.loc[(scp_df.iloc[:, 0] == l) |␣
↪(spc_df['diagnostic_subclass'] == l)]
            if len(scp_row) < 1:
                print(l, 'not found in scp_statements')
                break
            diagnostic_classes.append((scp_row['diagnostic_class'].values[0],␣
↪p))
    return sorted(diagnostic_classes, key=lambda x: x[1], reverse=True)

record_files500 = get_file_list(BASE_DIR, filter_function=lambda x:␣
↪'records500' in x)

db_df = read_ptbxl_database()
scp_df = read_ptbxl_scp_statements()
print(scp_df.iloc[:, 0])
read_label(db_df, scp_df, 'records500/00000/00001_hr')
```

```
0       NDT
1      NST_
2       DIG
3     LNGQT
4      NORM
      …
66     BIGU
67     AFLT
68    SVTAC
69     PSVT
70    TRIGU
Name: Unnamed: 0, Length: 71, dtype: object
```

```
[10]: [('NORM', '100.0')]
```

[75]:

```
import ast
from collections import defaultdict
dbdf = read_ptbxl_database()
all_labels = defaultdict(set)

#Collect all possible labels
codes = dbdf['scp_codes']
for row in codes:
    lbl_dict = ast.literal_eval(row)
    for k,v in lbl_dict.items():
        all_labels[k].update({v})

#Filter out labels that have only 0 probablity
all_labels = {k: v for k, v in all_labels.items() if max(v) > 0.0}

#Build a dict with filename as key and scp code as values
filenames = dbdf[['filename_hr', 'scp_codes']]
file_codes = defaultdict(dict)
for i, (f, c) in filenames.iterrows():
    lbl_dict = ast.literal_eval(c)
    for k, v in lbl_dict.items():
        if k in all_labels and v > 0.0: #First check not necessary
            file_codes[f][k] = v/100.0

code_indices = dict(zip(all_labels.keys(), range(len(all_labels.keys()))))
file_codes_onehot = dict()
for k, v in file_codes.items():
    hot_prob = np.zeros(len(code_indices))
    for ck, cv in v.items():
        hot_prob[code_indices[ck]] = cv
    file_codes_onehot[k] = hot_prob
print(len(filenames), len(file_codes_onehot))
```

```
21837 21837
```

[ ]:

```
[a[0] for a in sorted(code_indices.items(), key=lambda x: x[1])]
```

[11]:

```
import glob
p = '/media/julian/Volume/data/ECG/
 ↪ptb-xl-a-large-publicly-available-electrocardiography-dataset-1.0.1/
 ↪generated/1000'
for f in glob.glob(os.path.join(p, '*_hr.*')):
    fn = os.path.basename(f)
    fp = os.path.dirname(f)
    os.rename(f, os.path.join(fp, os.path.splitext(fn)[0]+'resampled'+os.path.
 ↪splitext(fn)[1]))
```

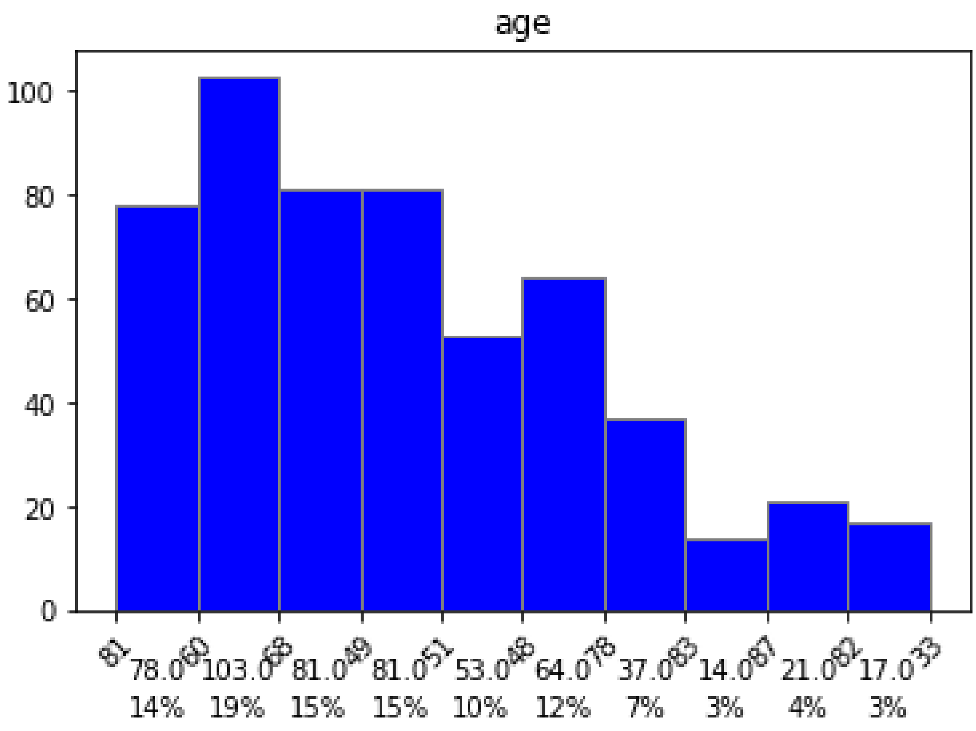
0.1 Playing around with PTB Comments

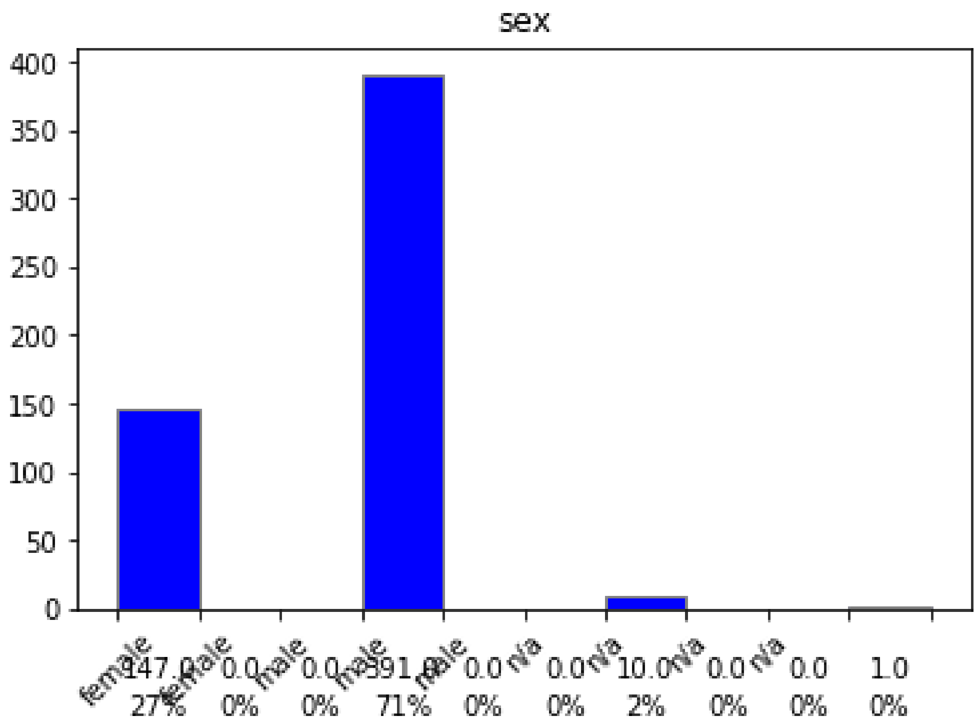
```
[20]: comment_data = []
      success = 0
      record_files = get_file_list(BASE_DIR, filter_function=lambda x: 'records500'␣
       ↪in x)
      for f in record_files:
          p = os.path.join(BASE_DIR, f)
          d = read_comment_map_PTB(p)
          comment_data.append(d)
      print(comment_data[0])
```

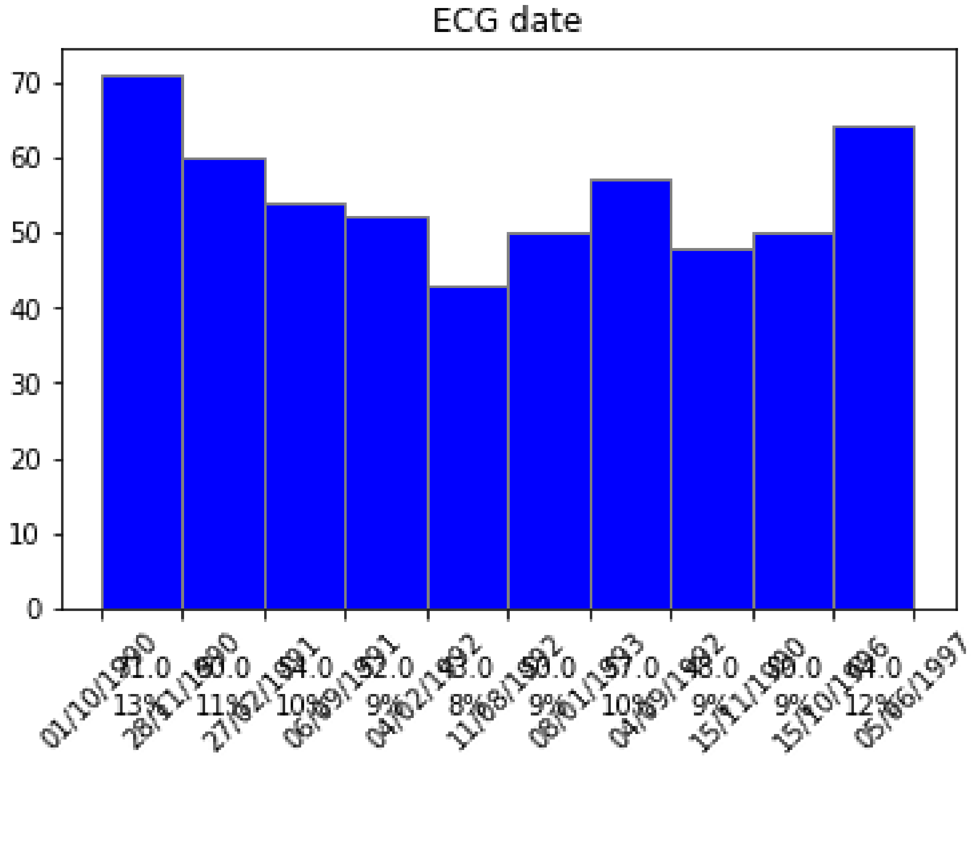
```
{}
```

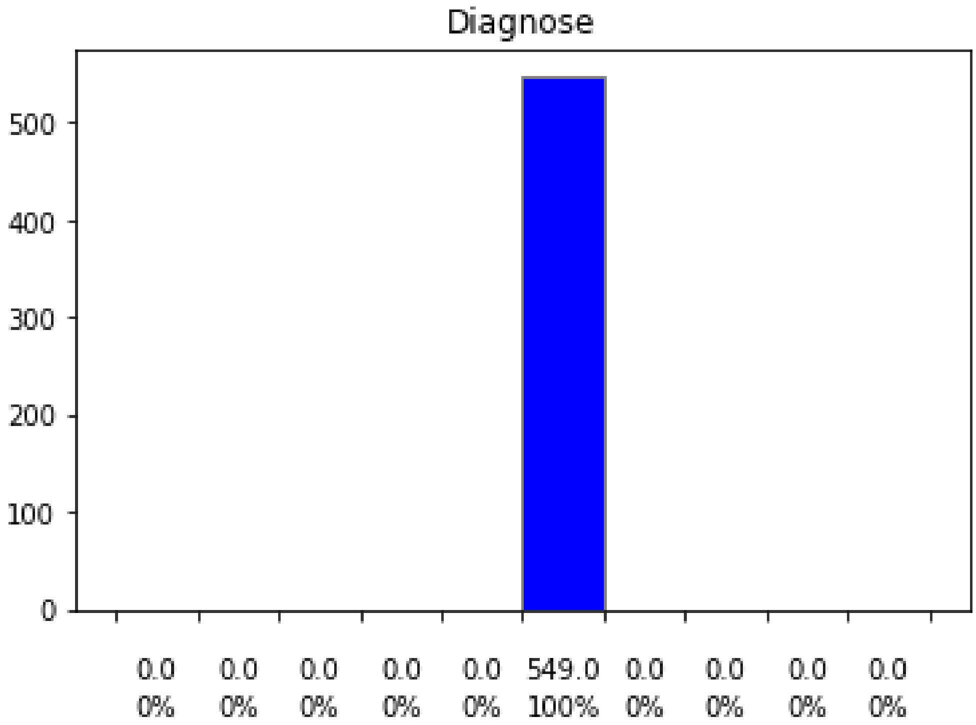
```
[ ]: comment_data
```

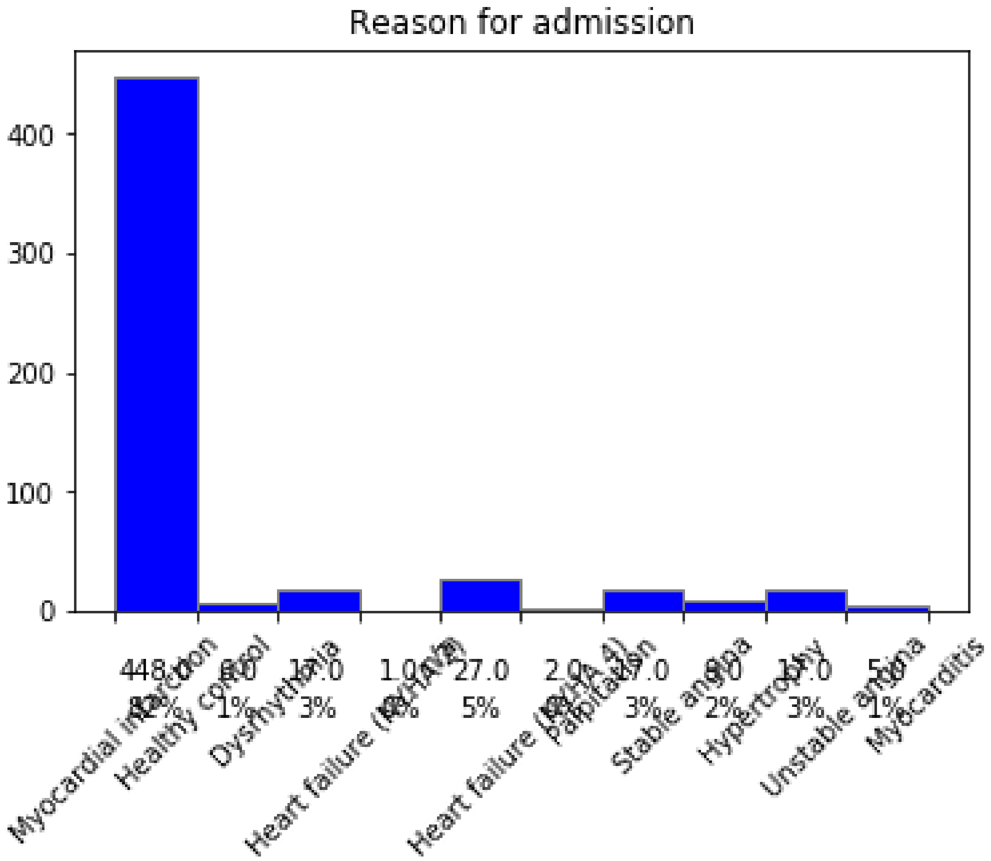
```
[34]: for k in comment_data[0].keys():
          plot([c[k] for c in comment_data], k)
```

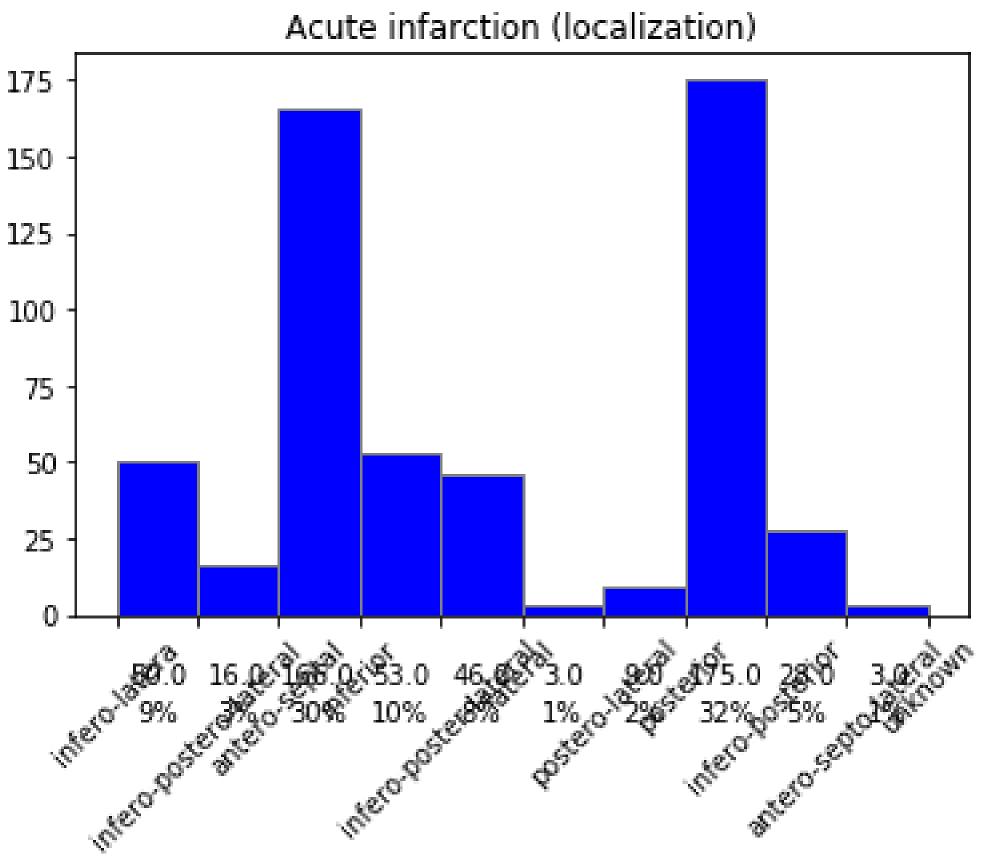


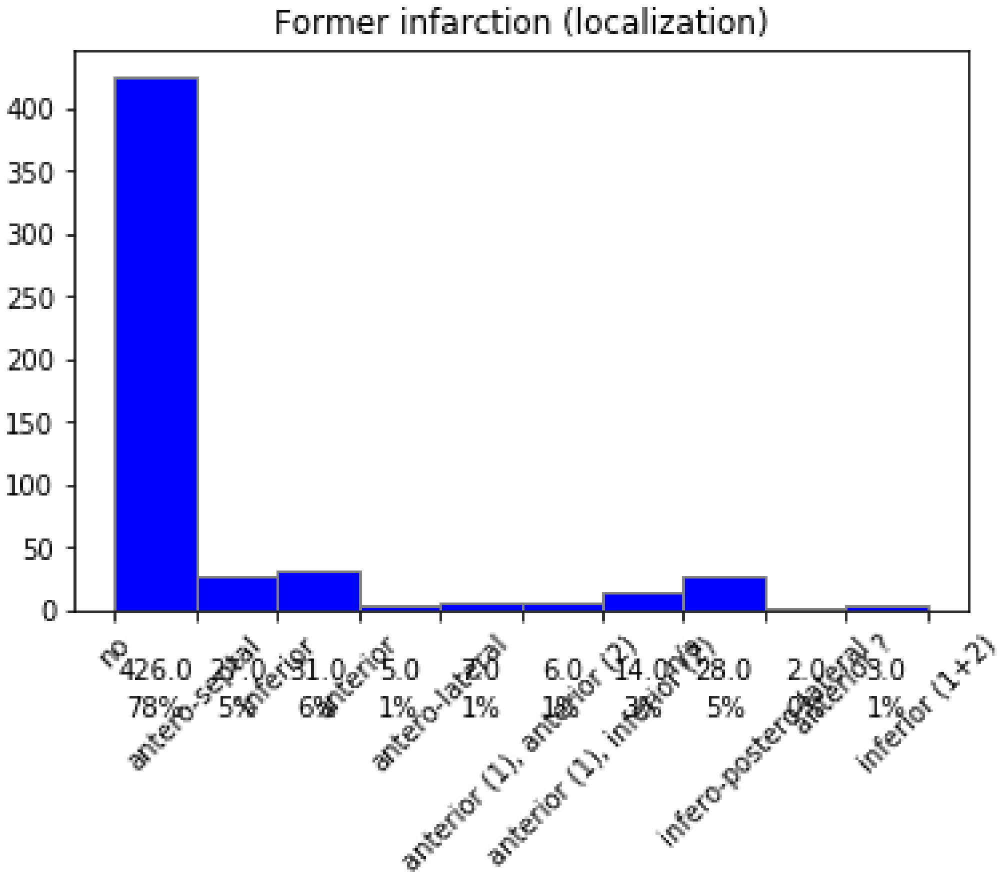

sex
400
350
300
250
200
150
100
50
0
female
female
male
male
male
n/a
n/a
n/a
n/a
147.0
27%
0.0
0%
0.0
0%
391.0
71%
0.0
0%
0.0
0%
10.0
2%
0.0
0%
0.0
0%
1.0
0%

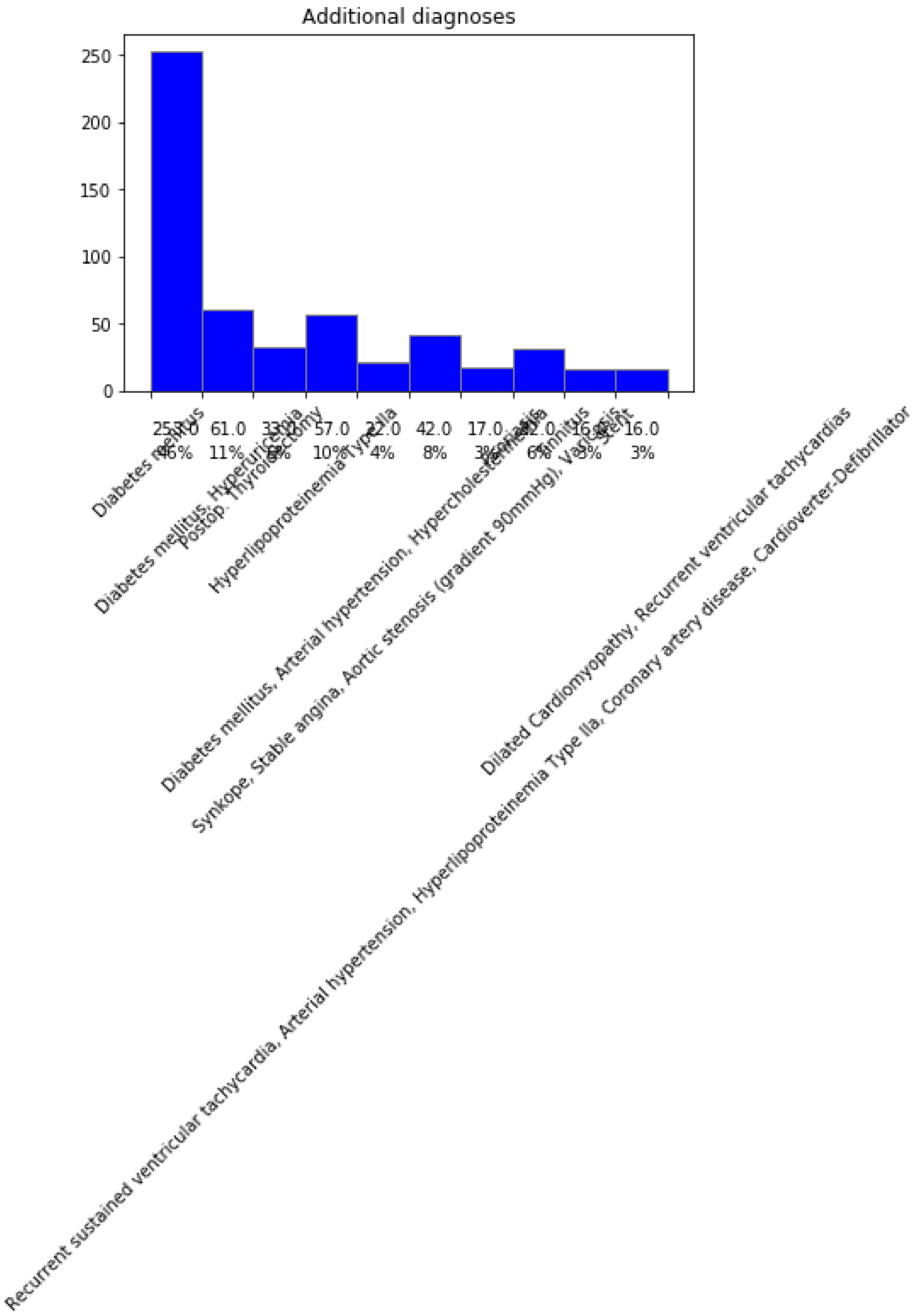

ECG date
70
60
50
40
30
20
10
0
01/10/1990
28/11/1990
27/02/1991
06/09/1991
04/02/1992
11/08/1992
08/01/1993
04/09/1992
15/11/1990
15/10/1996
05/06/1997
71.0
13%
60.0
11%
54.0
10%
52.0
9%
43.0
8%
50.0
9%
57.0
10%
48.0
9%
50.0
9%
64.0
12%

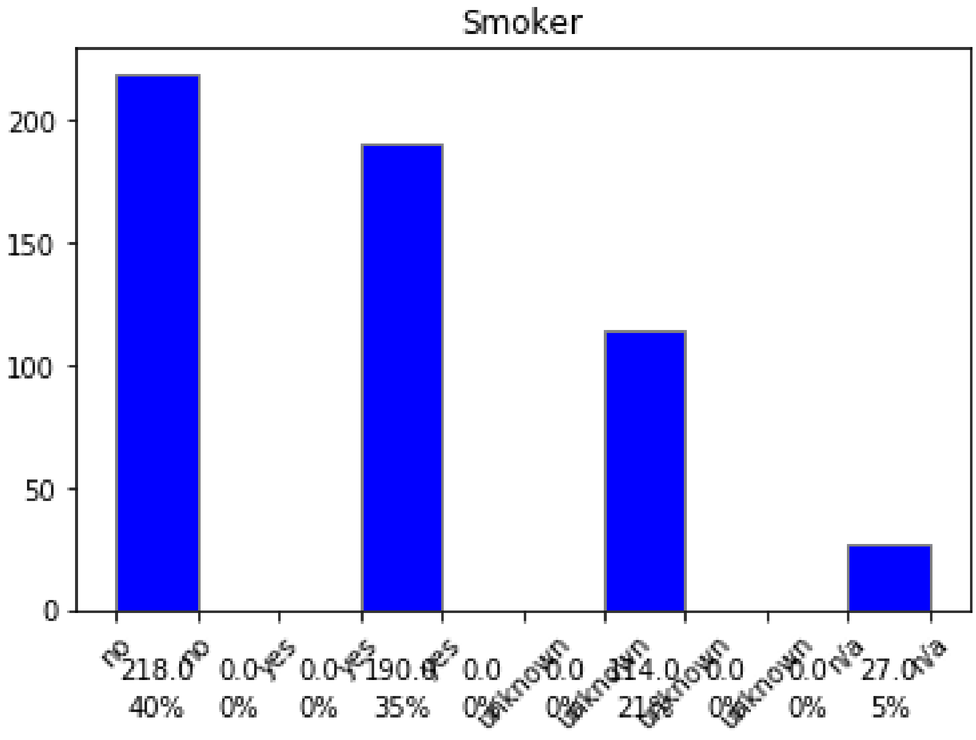

Diagnose
500
400
300
200
100
0
0.0
0%
0.0
0%
0.0
0%
0.0
0%
0.0
0%
549.0
100%
0.0
0%
0.0
0%
0.0
0%
0.0
0%

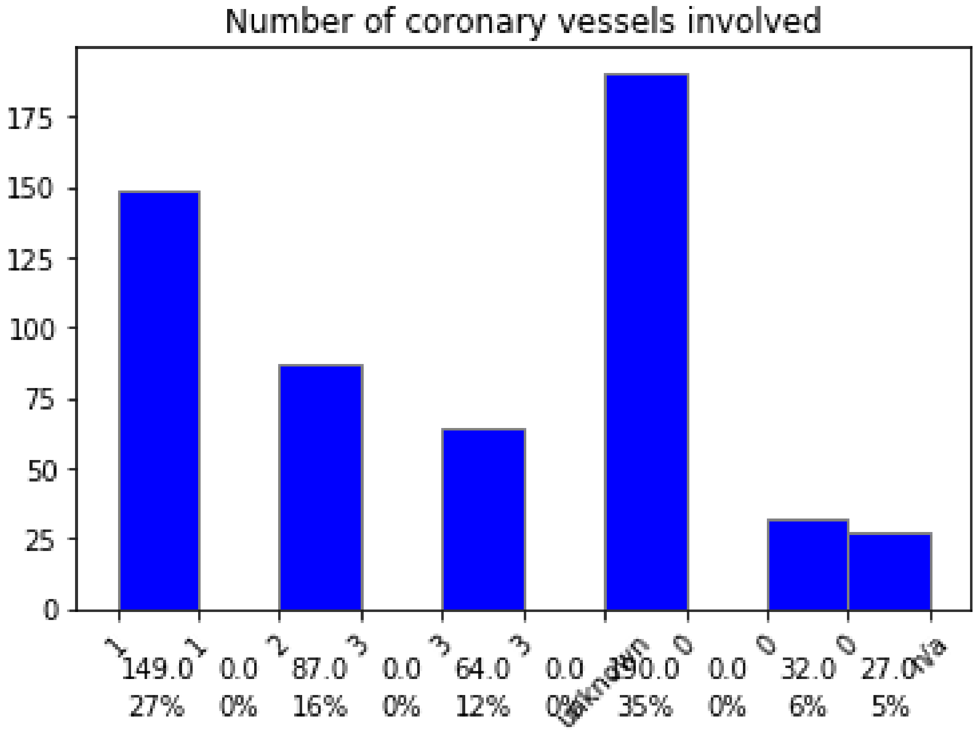

Reason for admission
0
100
200
300
400
448.0
82%
6.0
1%
17.0
3%
1.0
0%
27.0
5%
2.0
0%
17.0
3%
9.0
2%
17.0
3%
5.0
1%
Myocardial infarction
Healthy control
Dysrhythmia
Heart failure (NYHA 2)
Heart failure (NYHA 4)
Palpitation
Stable angina
Hypertrophy
Unstable angina
Myocarditis

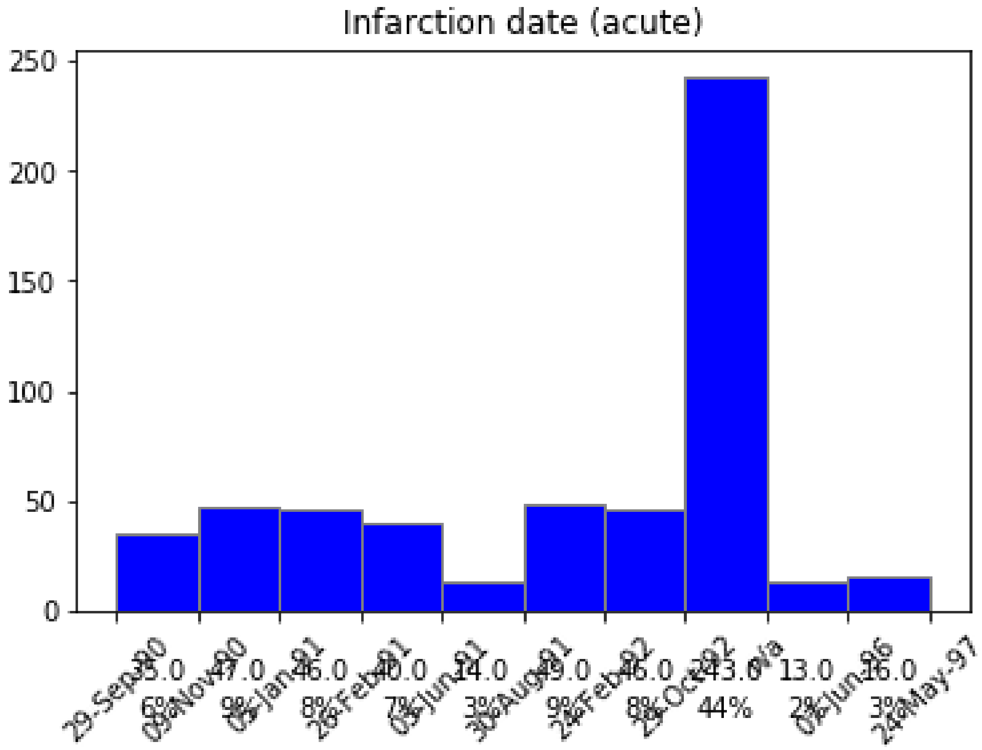

Acute infarction (localization)
175
150
125
100
75
50
25
0
50.0
9%
16.0
3%
166.0
30%
53.0
10%
46.0
8%
3.0
1%
9.0
2%
175.0
32%
28.0
5%
3.0
1%
infero-latera
infero-postero-lateral
antero-septal
inferior
infero-postero-lateral
lateral
postero-lateral
posterior
infero-posterior
antero-septo-lateral
unknown

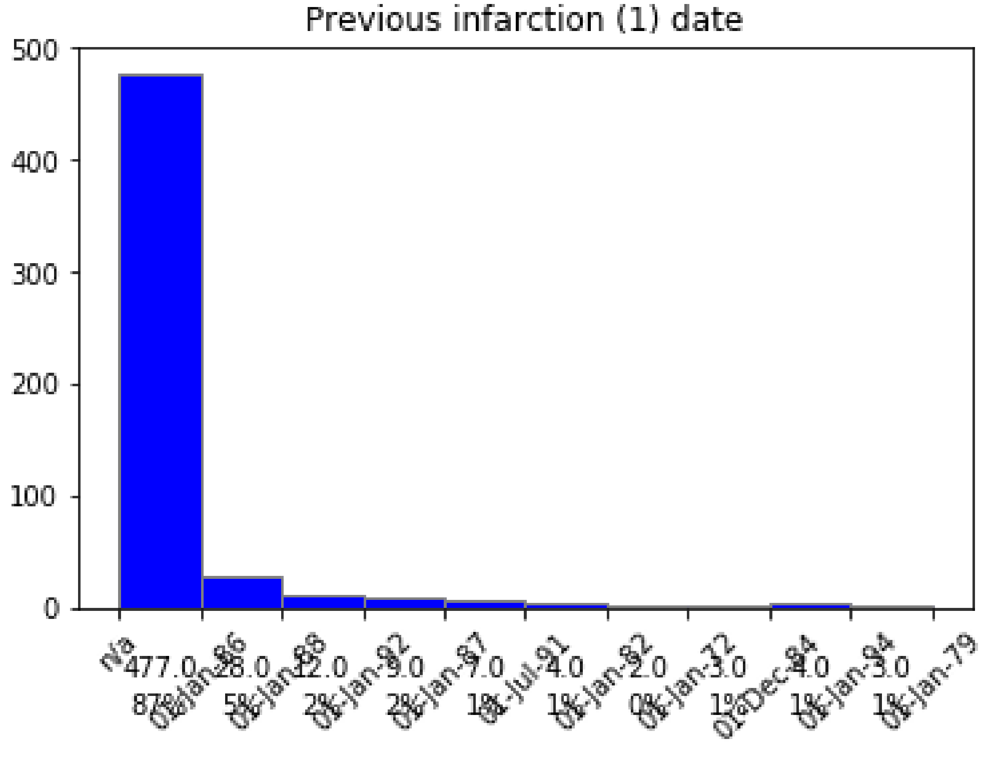

Former infarction (localization)
0
50
100
150
200
250
300
350
400
no
antero-septal
inferior
anterior
antero-lateral
anterior (1), anterior (2)
anterior (1), inferior (2)
infero-posterolateral
anterior ?
inferior (1+2)
426.0
78%
27.0
5%
31.0
6%
5.0
1%
7.0
1%
6.0
1%
14.0
3%
28.0
5%
2.0
0%
3.0
1%

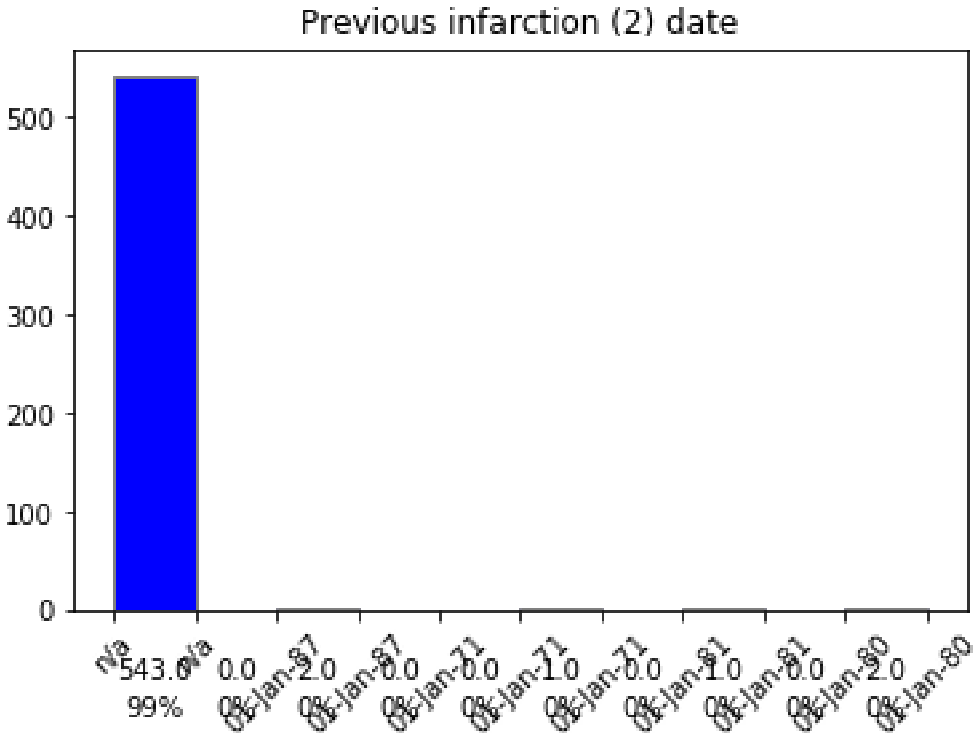

Additional diagnoses
250
200
150
100
50
0
253.0
46%
61.0
11%
33.0
6%
57.0
10%
22.0
4%
42.0
8%
17.0
3%
32.0
6%
16.0
3%
16.0
3%
Diabetes mellitus
Diabetes mellitus, Hyperuricemia
Postop. Thyroidectomy
Hyperlipoproteinemia Type IIa
Diabetes mellitus, Arterial hypertension, Hypercholesterinemia
Synkope, Stable angina, Aortic stenosis (gradient 90mmHg), Varicosis
Tinnitus
Dilated Cardiomyopathy, Recurrent ventricular tachycardias
Recurrent sustained ventricular tachycardia, Arterial hypertension, Hyperlipoproteinemia Type IIa, Coronary artery disease, Cardioverter-Defibrillator

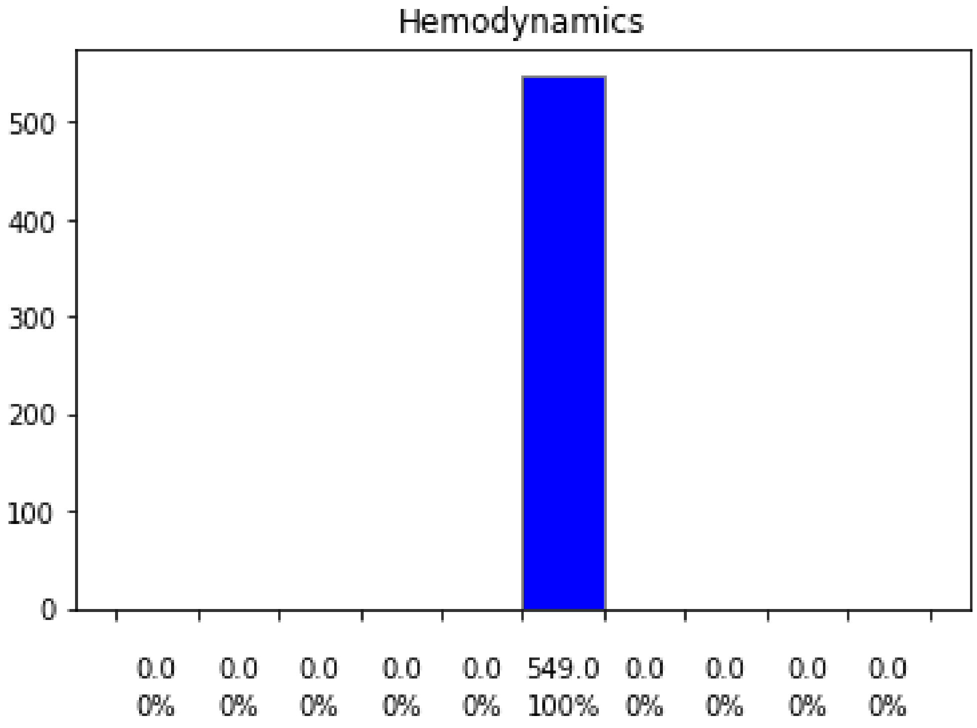

Smoker
0
50
100
150
200
no
no
yes
yes
yes
unknown
unknown
unknown
unknown
n/a
n/a
218.0
40%
0.0
0%
0.0
0%
190.0
35%
0.0
0%
0.0
0%
114.0
21%
0.0
0%
0.0
0%
27.0
5%

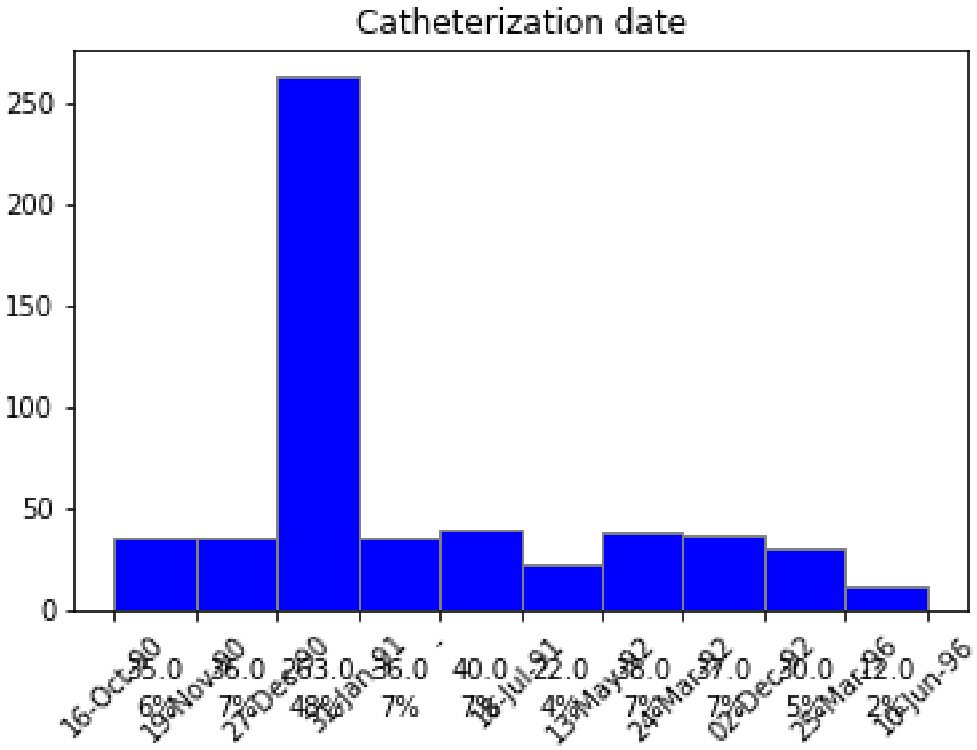

Number of coronary vessels involved
0
25
50
75
100
125
150
175
1
1
2
3
3
3
unknown
0
0
0
n/a
149.0
27%
0.0
0%
87.0
16%
0.0
0%
64.0
12%
0.0
0%
190.0
35%
0.0
0%
32.0
6%
27.0
5%

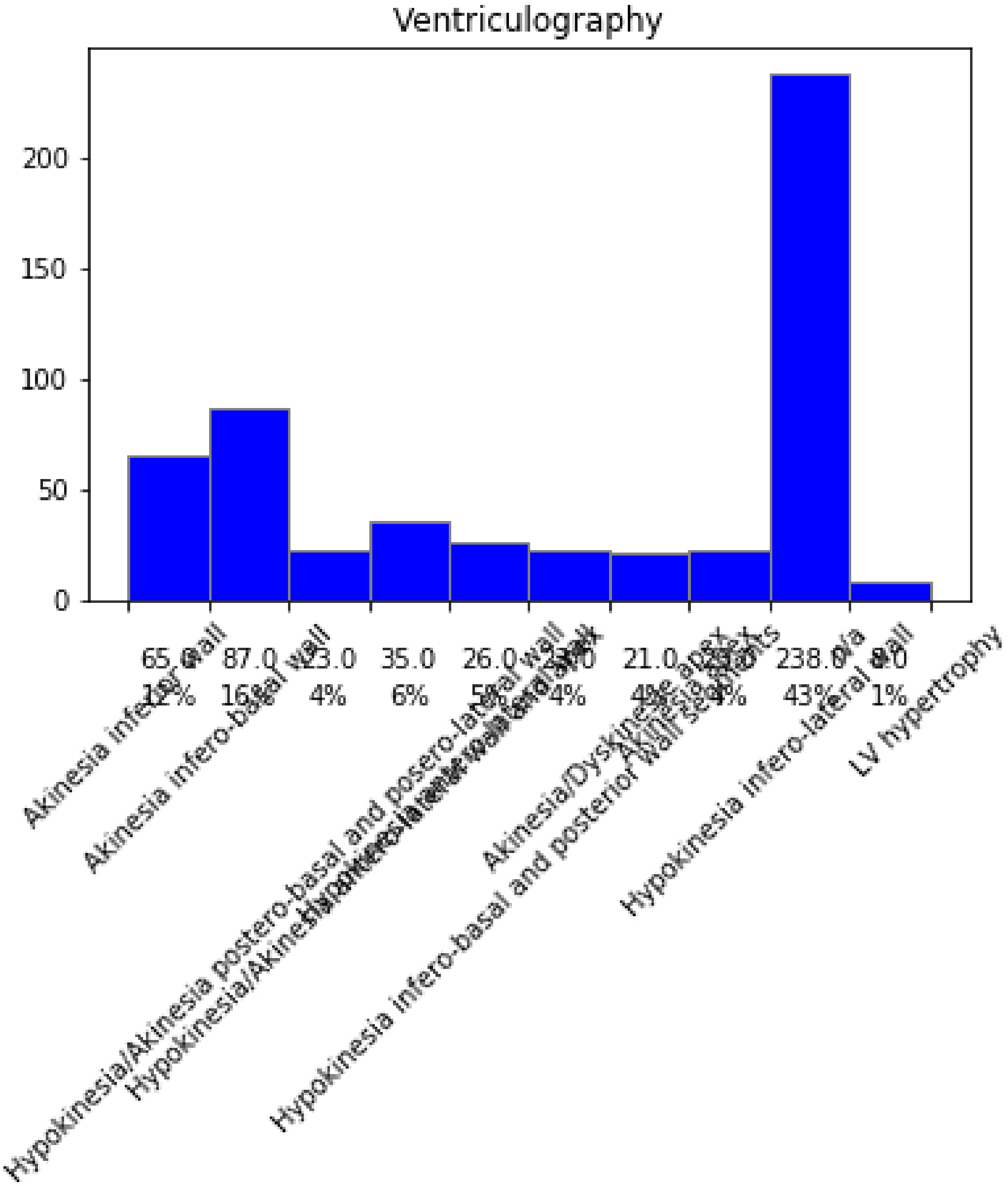

Infarction date (acute)
250
200
150
100
50
0
29-Sep-90
09-Nov-90
05-Jan-91
26-Feb-91
05-Jun-91
30-Aug-91
24-Feb-92
23-Oct-92
n/a
07-Jun-96
24-May-97
35.0
6%
47.0
9%
46.0
8%
40.0
7%
14.0
3%
49.0
9%
46.0
8%
243.0
44%
13.0
2%
16.0
3%

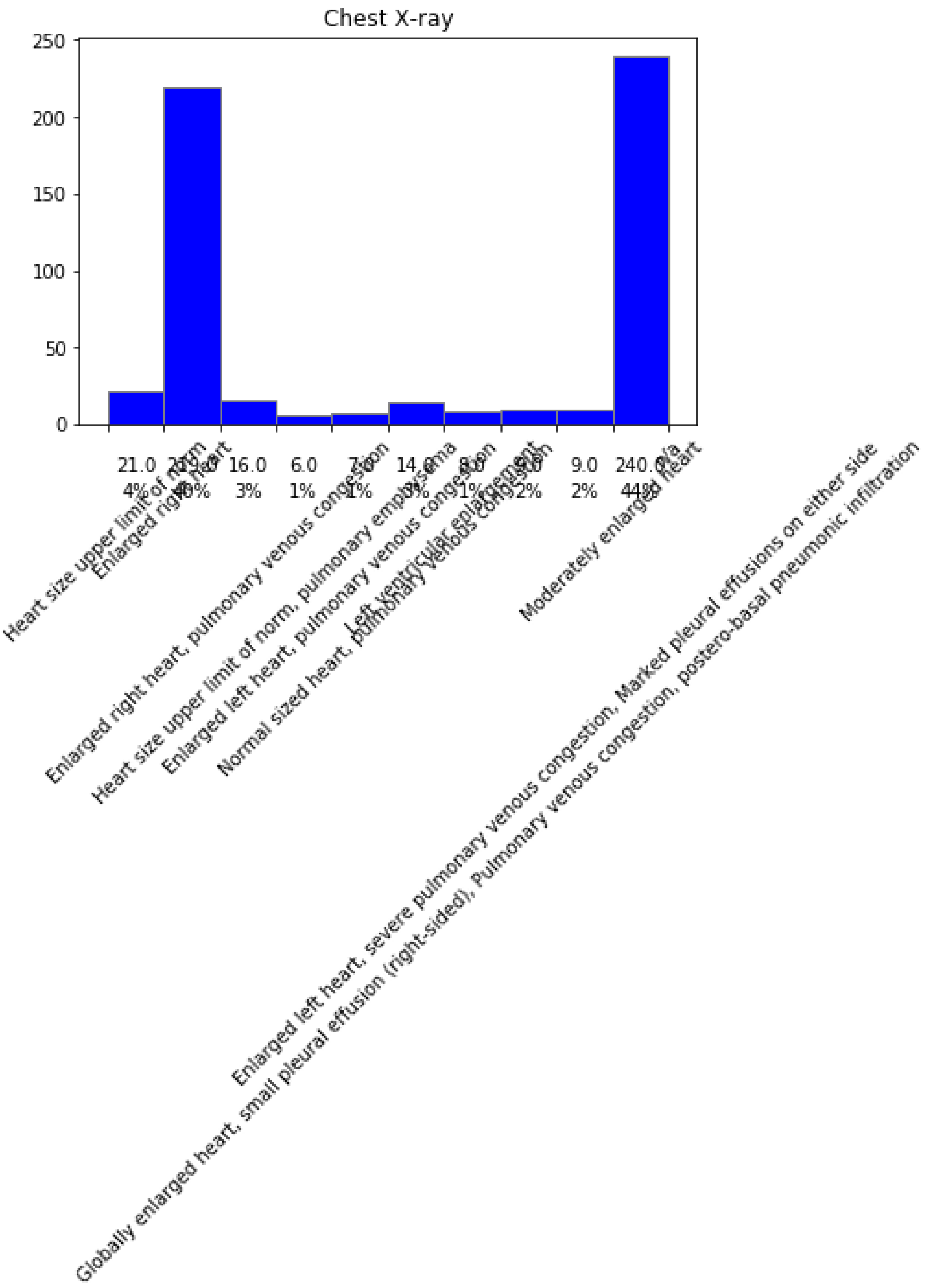

Previous infarction (1) date
500
400
300
200
100
0
n/a
01-Jan-86
01-Jan-88
01-Jan-92
01-Jan-87
01-Jul-91
01-Jan-82
01-Jan-72
01-Dec-84
01-Jan-94
01-Jan-79
477.0
87%
28.0
5%
12.0
2%
9.0
2%
7.0
1%
4.0
1%
2.0
0%
3.0
1%
4.0
1%
3.0
1%

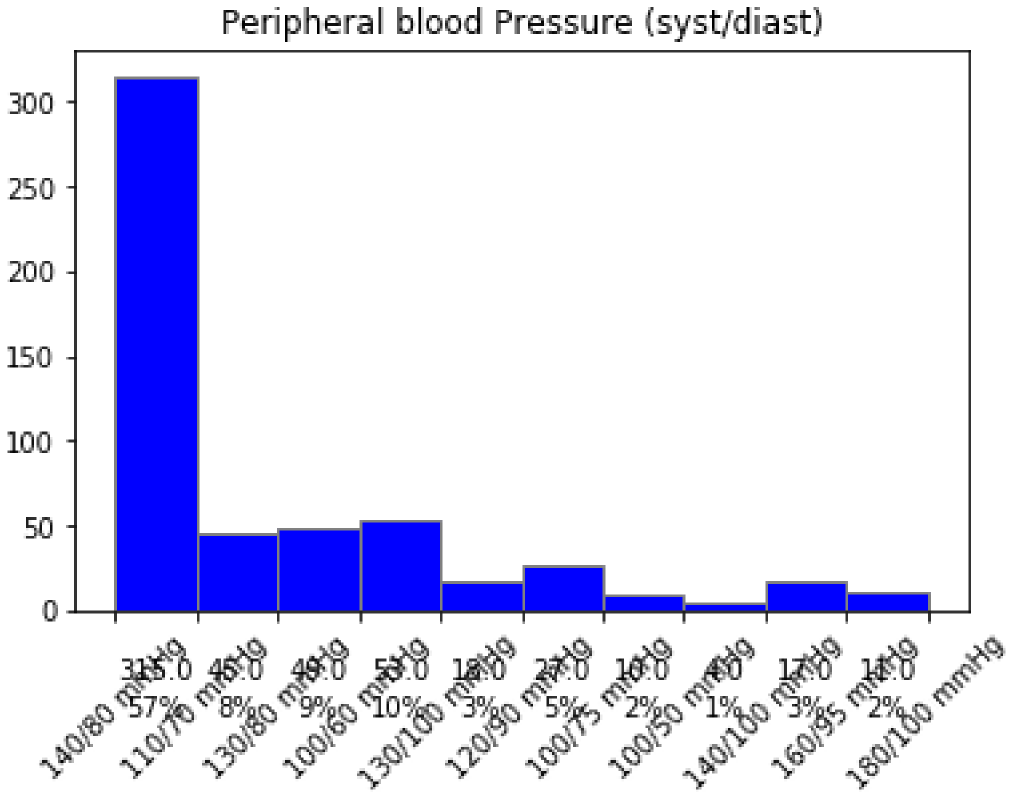

Previous infarction (2) date
500
400
300
200
100
0
n/a
n/a
01-Jan-87
01-Jan-87
01-Jan-71
01-Jan-71
01-Jan-71
01-Jan-81
01-Jan-81
01-Jan-80
01-Jan-80
543.0
99%
0.0
0%
2.0
0%
0.0
0%
0.0
0%
1.0
0%
0.0
0%
1.0
0%
0.0
0%
2.0
0%

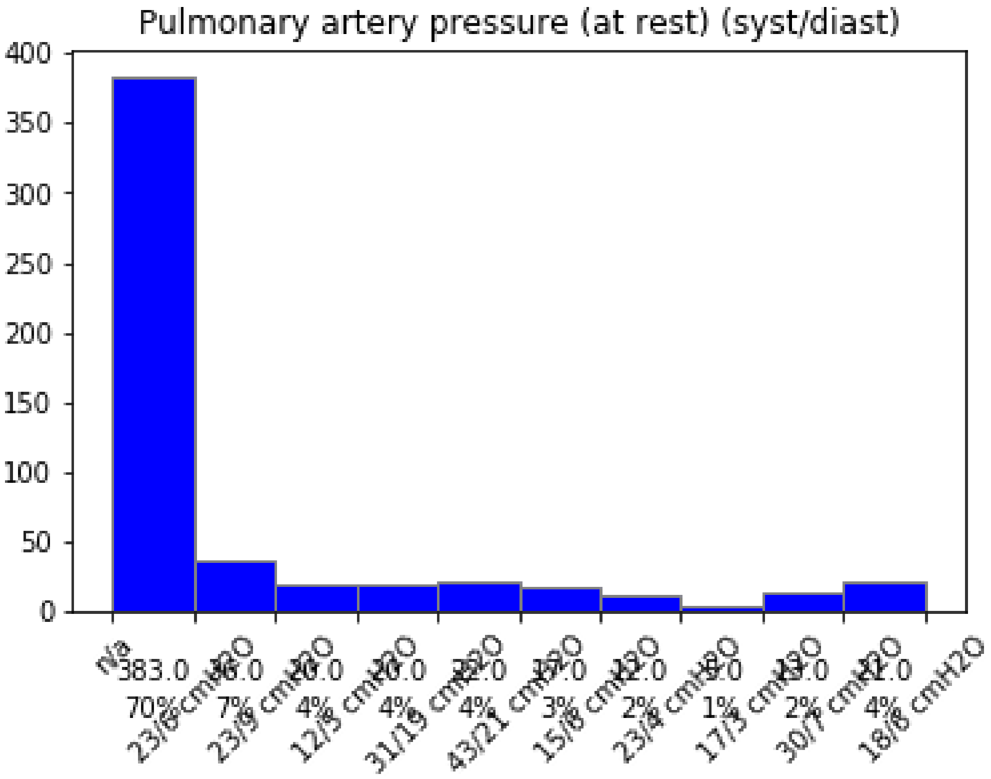

Hemodynamics
500
400
300
200
100
0
0.0
0%
0.0
0%
0.0
0%
0.0
0%
0.0
0%
549.0
100%
0.0
0%
0.0
0%
0.0
0%
0.0
0%

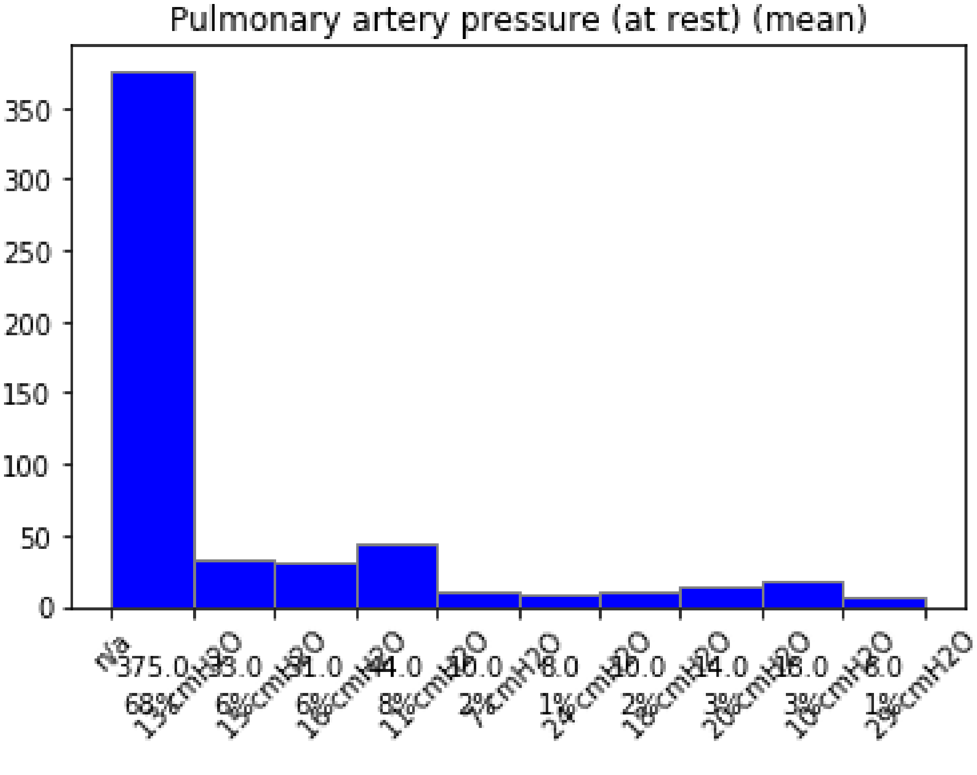

Catheterization date
0
50
100
150
200
250
16-Oct-90
19-Nov-90
27-Dec-90
31-Jan-91
.
16-Jul-91
13-May-92
24-Mar-92
02-Dec-92
25-Mar-96
10-Jun-96
35.0
6%
36.0
7%
263.0
48%
36.0
7%
40.0
7%
22.0
4%
38.0
7%
37.0
7%
30.0
5%
12.0
2%

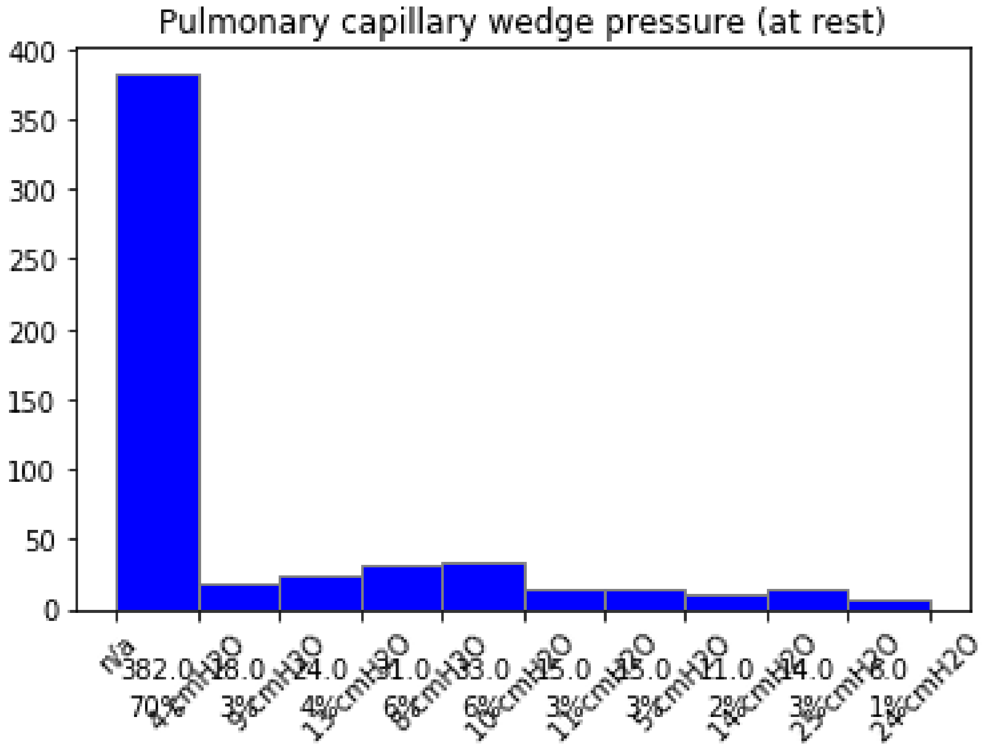

Ventriculography
0
50
100
150
200
65.0
12%
87.0
16%
23.0
4%
35.0
6%
26.0
5%
23.0
4%
21.0
4%
23.0
4%
238.0
43%
8.0
1%
Akinesia inferior wall
Akinesia infero-basal wall
Hypokinesia/Akinesia postero-basal and posero-lateral wall
Hypokinesia/Akinesia anterior wall and apex
Akinesia/Dyskinesia apex
Akinesia apex
Hypokinesia infero-basal and posterior wall segments
n/a
Hypokinesia infero-lateral wall
LV hypertrophy

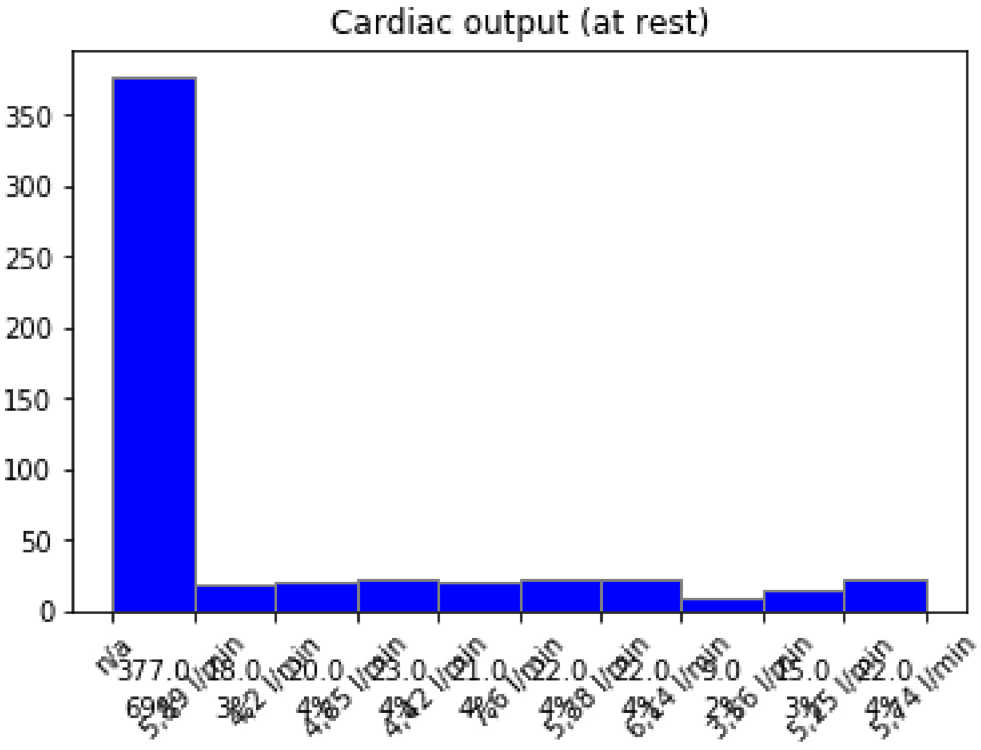

Chest X-ray
250
200
150
100
50
0
21.0
4%
219.0
40%
16.0
3%
6.0
1%
7.0
1%
14.0
3%
8.0
1%
9.0
2%
9.0
2%
240.0
44%
Heart size upper limit of norm
Enlarged right heart
Enlarged right heart, pulmonary venous congestion
Heart size upper limit of norm, pulmonary emphysema
Enlarged left heart, pulmonary venous congestion
Left ventricular enlargement
Normal sized heart, pulmonary venous congestion
n/a
Moderately enlarged heart
Enlarged left heart, severe pulmonary venous congestion, Marked pleural effusions on either side
Globally enlarged heart, small pleural effusion (right-sided), Pulmonary venous congestion, postero-basal pneumonic infiltration

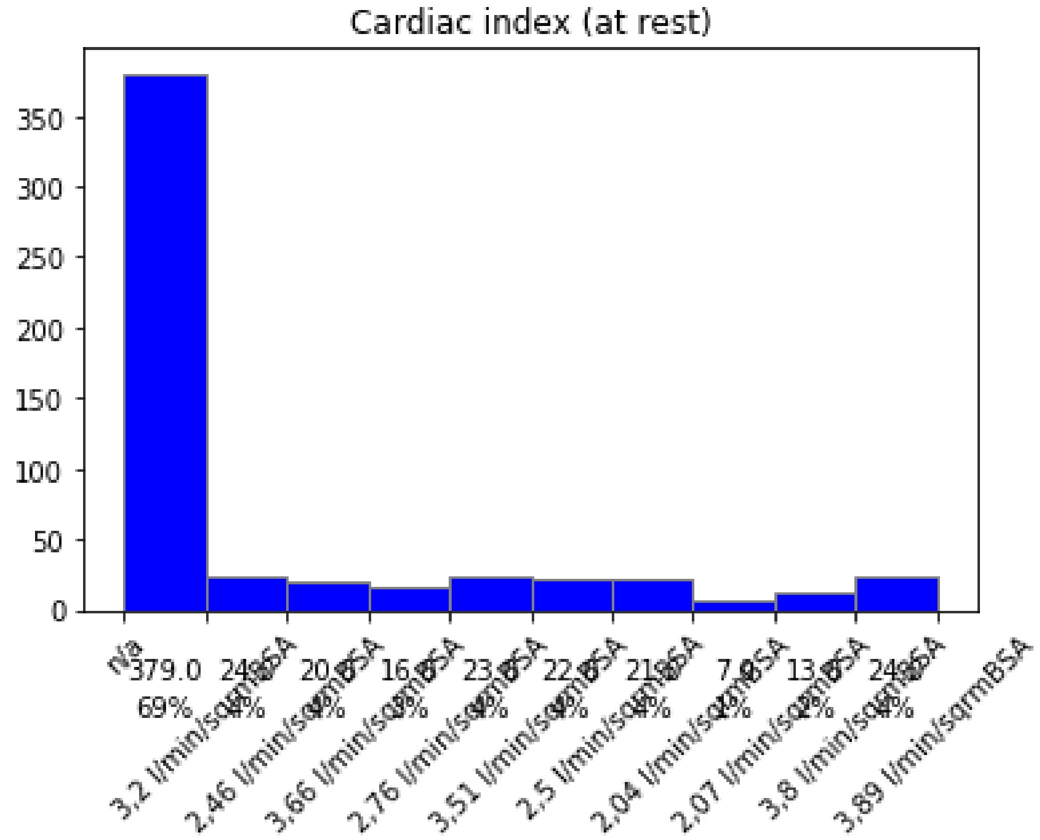

Peripheral blood Pressure (syst/diast)
300
250
200
150
100
50
0
315.0
57%
45.0
8%
49.0
9%
53.0
10%
17.0
3%
27.0
5%
10.0
2%
4.0
1%
17.0
3%
11.0
2%
140/80 mmHg
110/70 mmHg
130/80 mmHg
100/60 mmHg
130/100 mmHg
120/90 mmHg
100/75 mmHg
100/50 mmHg
140/100 mmHg
160/95 mmHg
180/100 mmHg

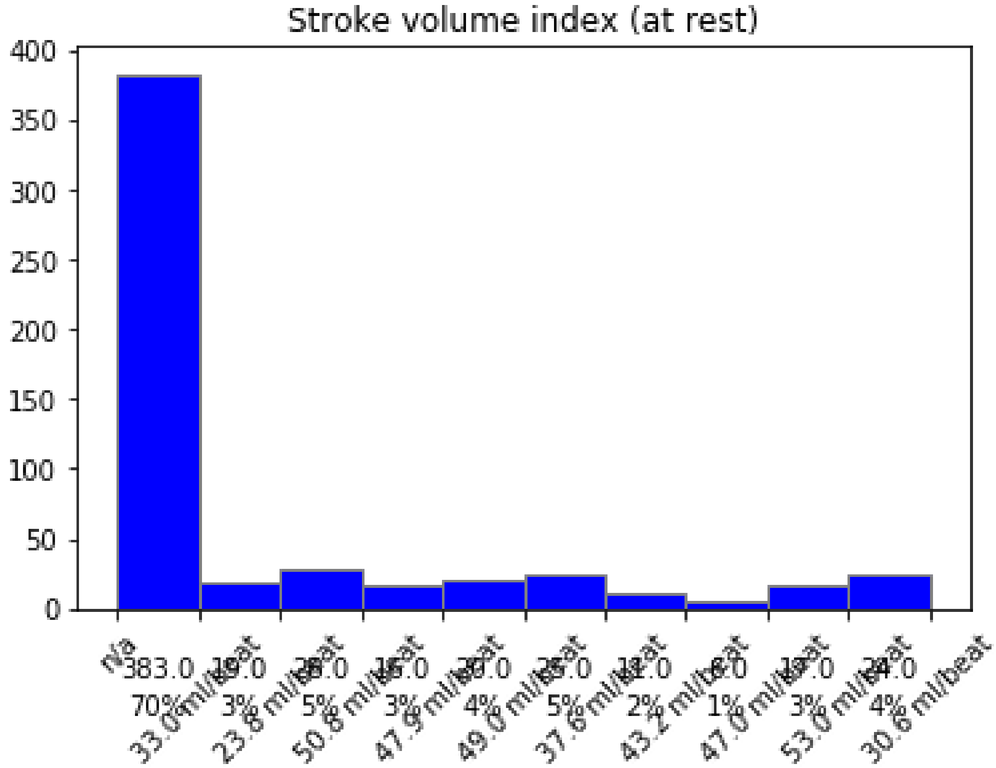

Pulmonary artery pressure (at rest) (syst/diast)
400
350
300
250
200
150
100
50
0
n/a
23/6 cmH2O
23/9 cmH2O
12/5 cmH2O
31/15 cmH2O
43/21 cmH2O
15/8 cmH2O
23/4 cmH2O
17/3 cmH2O
30/7 cmH2O
18/8 cmH2O
383.0
70%
36.0
7%
20.0
4%
20.0
4%
22.0
4%
17.0
3%
12.0
2%
3.0
1%
13.0
2%
21.0
4%

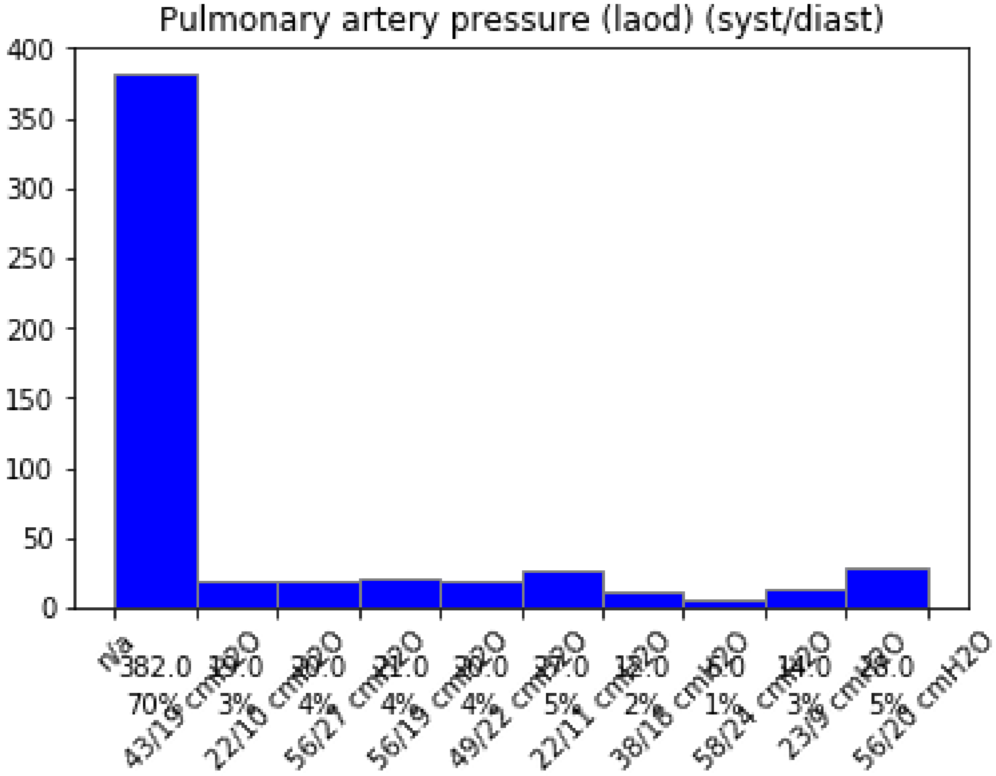

Pulmonary artery pressure (at rest) (mean)
350
300
250
200
150
100
50
0
n/a
13 cmH2O
15 cmH2O
16 cmH2O
11 cmH2O
7 cmH2O
24 cmH2O
18 cmH2O
20 cmH2O
10 cmH2O
29 cmH2O
375.0
68%
33.0
6%
31.0
6%
44.0
8%
10.0
2%
8.0
1%
10.0
2%
14.0
3%
18.0
3%
6.0
1%

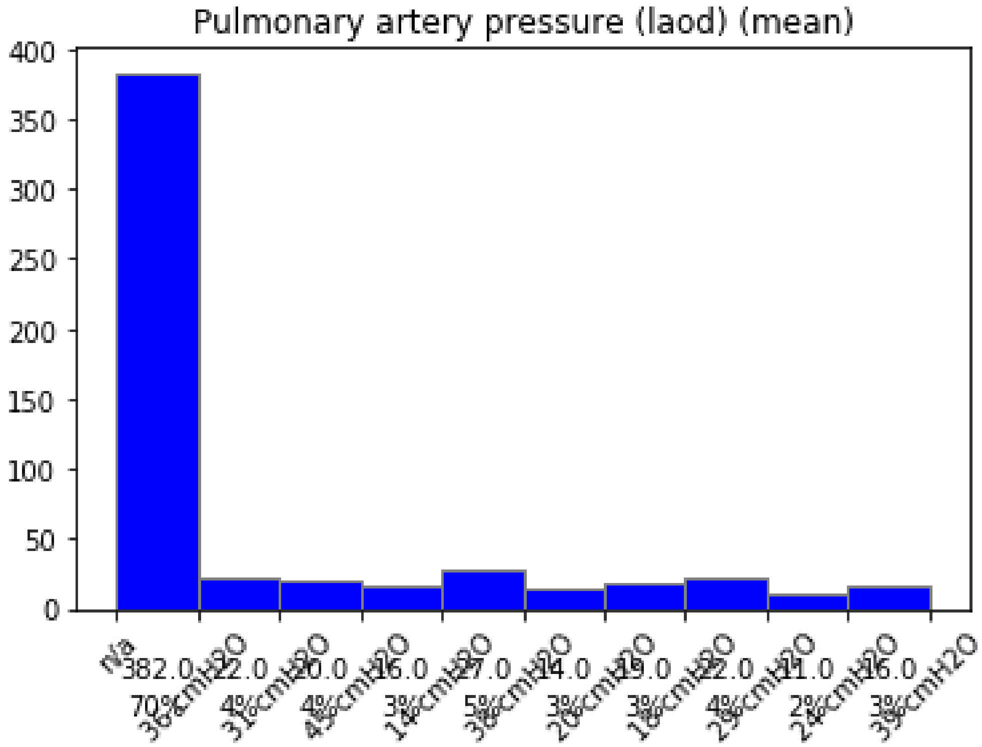

Pulmonary capillary wedge pressure (at rest)
400
350
300
250
200
150
100
50
0
n/a
4 cmH2O
9 cmH2O
1[illegible] cmH2O
8 cmH2O
10 cmH2O
11 cmH2O
5 cmH2O
14 cmH2O
2[illegible] cmH2O
24 cmH2O
382.0
70%
18.0
3%
24.0
4%
31.0
6%
33.0
6%
15.0
3%
15.0
3%
11.0
2%
14.0
3%
6.0
1%

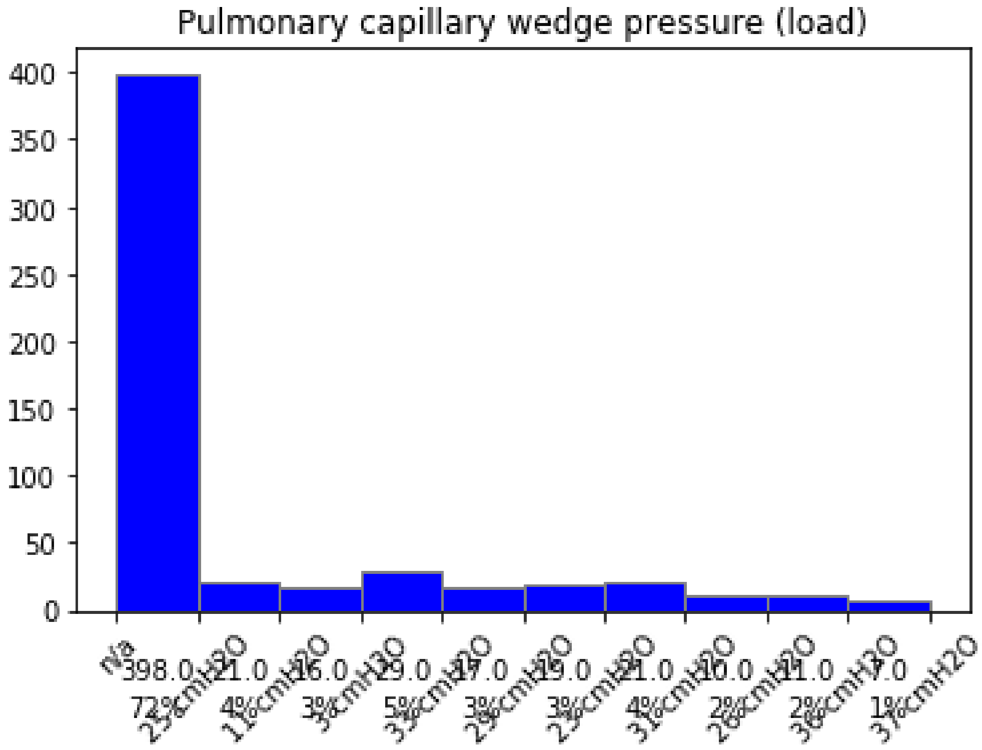

Cardiac output (at rest)
0
50
100
150
200
250
300
350
n/a
5,09 l/min
4,2 l/min
4,65 l/min
4,12 l/min
7,6 l/min
5,88 l/min
6,14 l/min
3,66 l/min
5,25 l/min
5,14 l/min
377.0
69%
18.0
3%
20.0
4%
23.0
4%
21.0
4%
22.0
4%
22.0
4%
9.0
2%
15.0
3%
22.0
4%

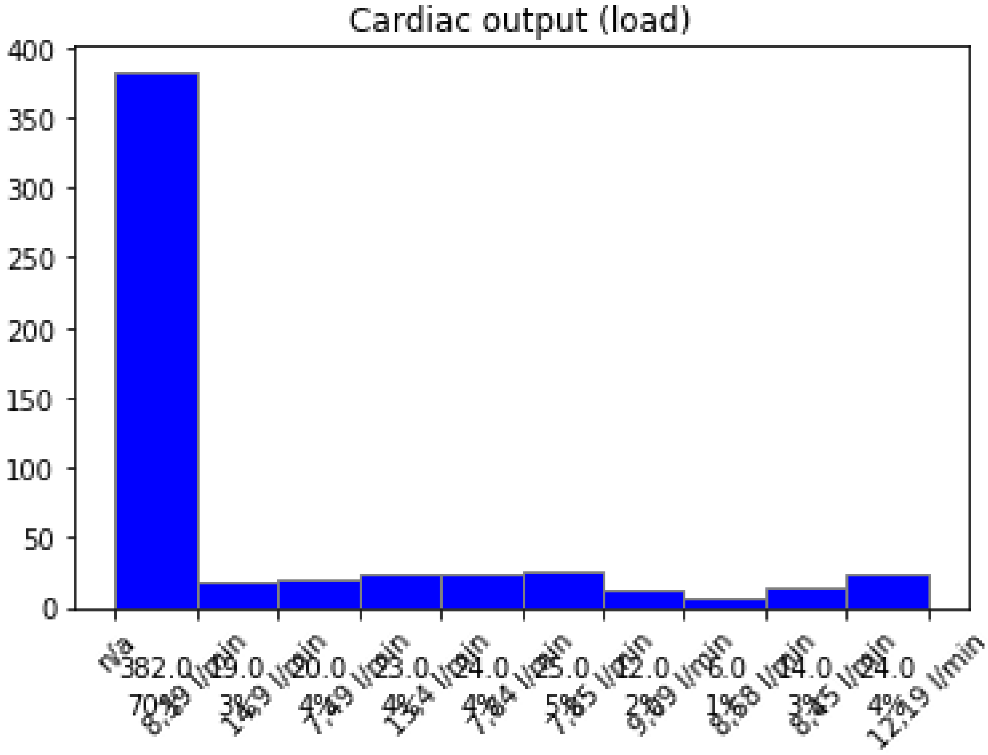

Cardiac index (at rest)
0
50
100
150
200
250
300
350
n/a
3,2 l/min/sqmBSA
2,46 l/min/sqmBSA
3,66 l/min/sqmBSA
2,76 l/min/sqmBSA
3,51 l/min/sqmBSA
2,5 l/min/sqmBSA
2,04 l/min/sqmBSA
2,07 l/min/sqmBSA
3,8 l/min/sqmBSA
3,89 l/min/sqmBSA
379.0
69%
24.0
4%
20.0
4%
16.0
3%
23.0
4%
22.0
4%
21.0
4%
7.0
1%
13.0
2%
24.0
4%

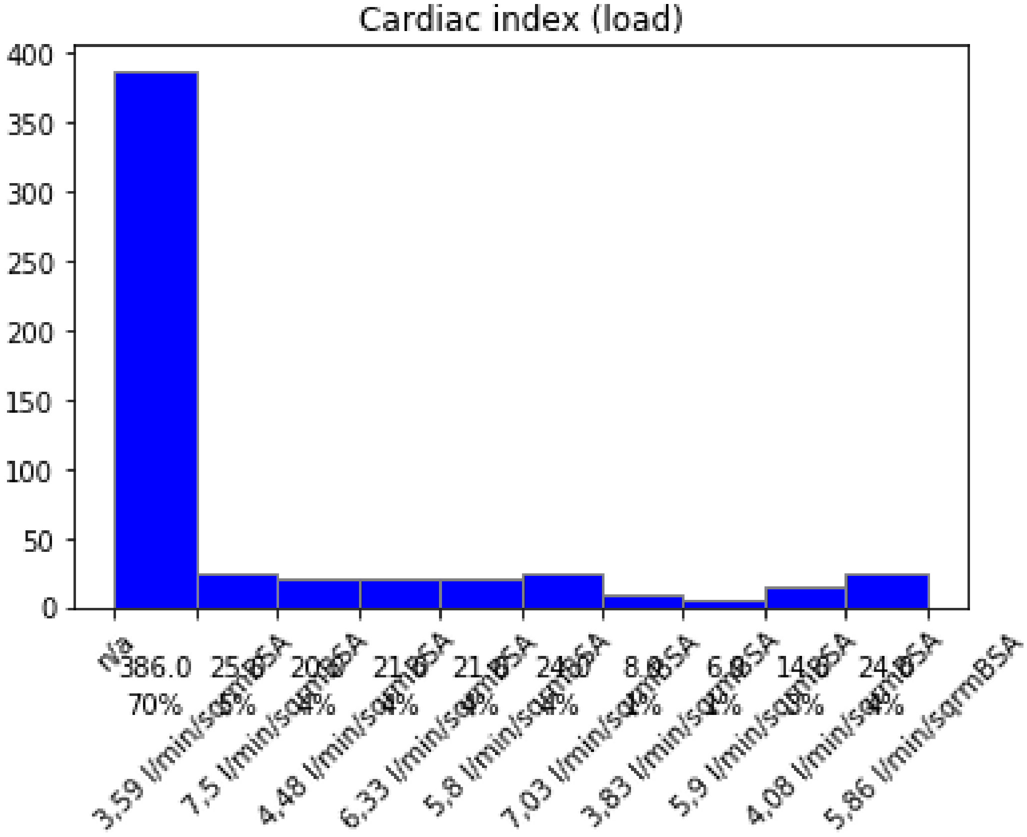

Stroke volume index (at rest)
0
50
100
150
200
250
300
350
400
n/a
33.0 ml/beat
23.8 ml/beat
50.8 ml/beat
47.9 ml/beat
49.0 ml/beat
37.6 ml/beat
43.2 ml/beat
47.0 ml/beat
53.0 ml/beat
30.6 ml/beat
383.0
70%
19.0
3%
28.0
5%
16.0
3%
20.0
4%
25.0
5%
11.0
2%
6.0
1%
17.0
3%
24.0
4%

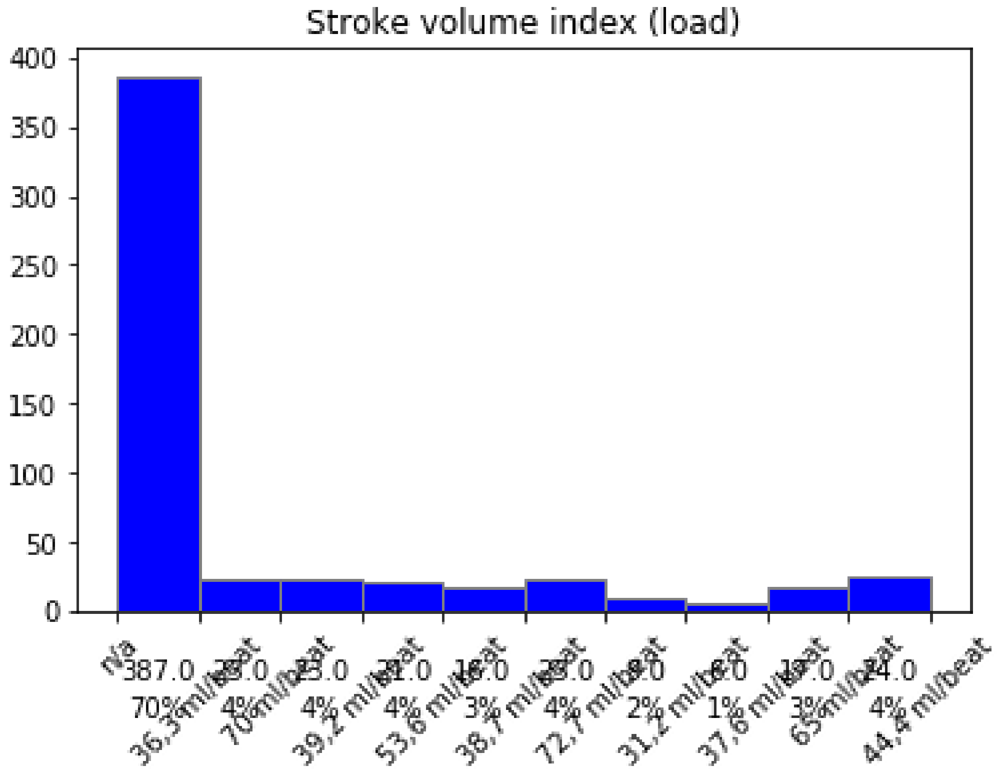

Pulmonary artery pressure (laod) (syst/diast)
0
50
100
150
200
250
300
350
400
n/a
43/19 cmH2O
22/10 cmH2O
56/27 cmH2O
56/19 cmH2O
49/22 cmH2O
22/11 cmH2O
38/18 cmH2O
58/24 cmH2O
23/9 cmH2O
56/20 cmH2O
382.0
70%
19.0
3%
20.0
4%
21.0
4%
20.0
4%
27.0
5%
12.0
2%
6.0
1%
14.0
3%
28.0
5%

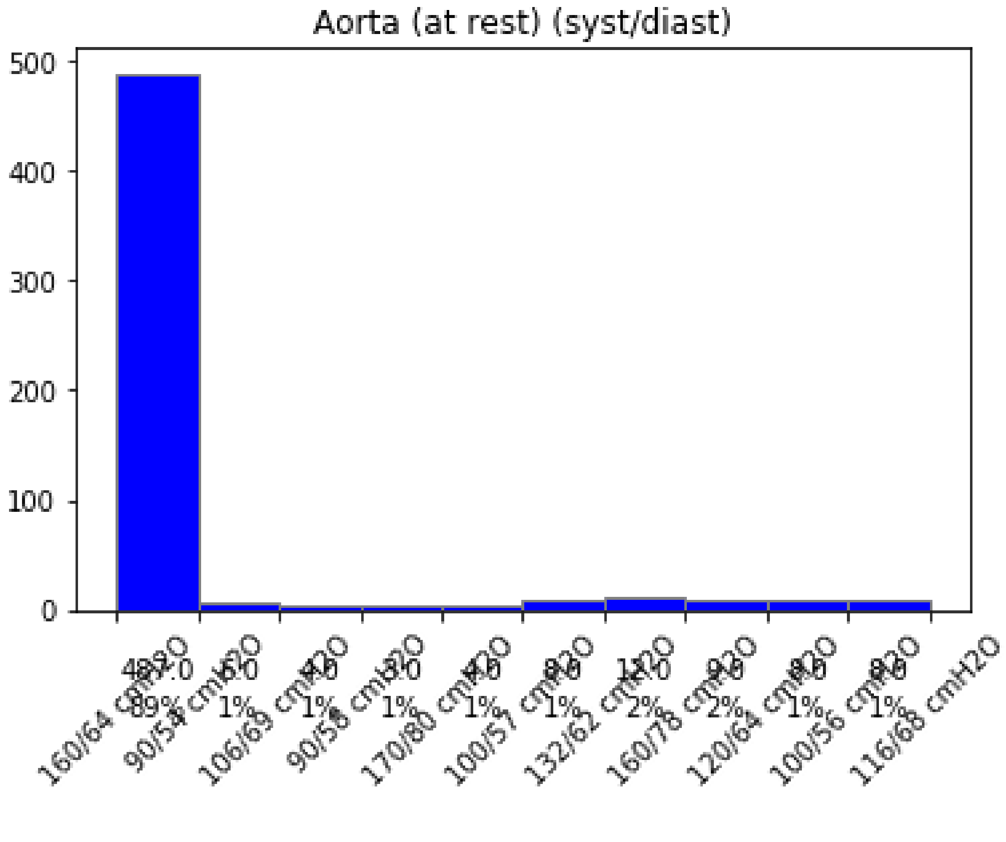

Pulmonary artery pressure (laod) (mean)
400
350
300
250
200
150
100
50
0
n/a
36cmH2O
31cmH2O
4?cmH2O
14cmH2O
38cmH2O
20cmH2O
18cmH2O
29cmH2O
24cmH2O
39cmH2O
382.0
70%
22.0
4%
20.0
4%
16.0
3%
27.0
5%
14.0
3%
19.0
3%
22.0
4%
11.0
2%
16.0
3%

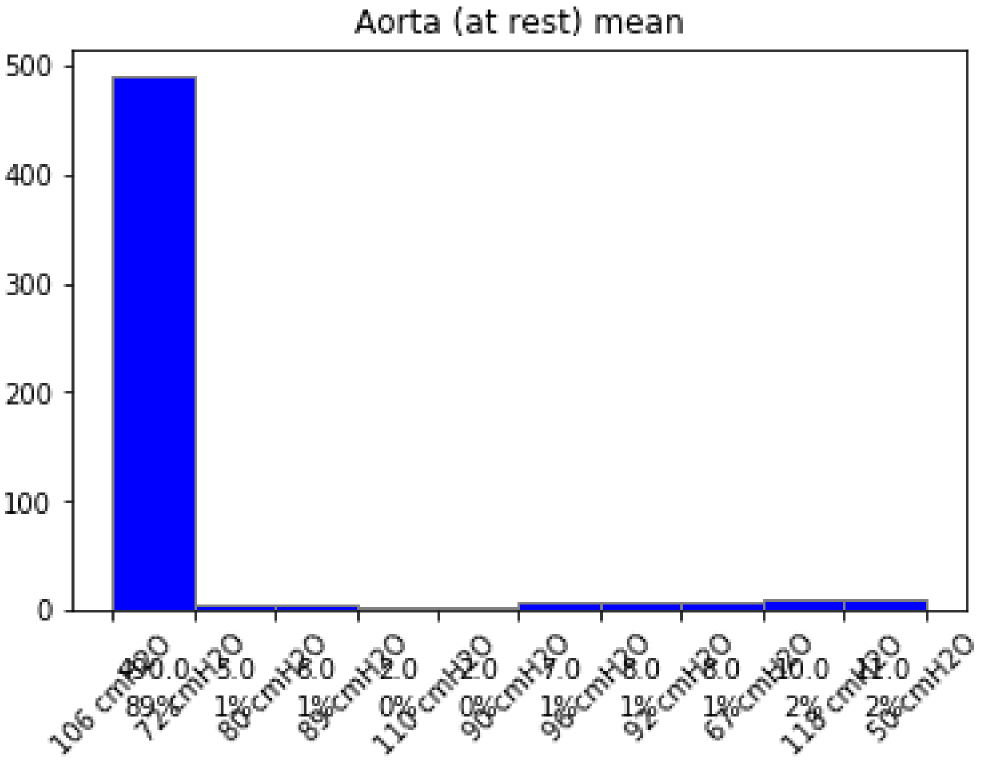

Pulmonary capillary wedge pressure (load)
0
50
100
150
200
250
300
350
400
n/a
25 cmH2O
11 cmH2O
3 cmH2O
33 cmH2O
29 cmH2O
23 cmH2O
31 cmH2O
26 cmH2O
36 cmH2O
37 cmH2O
398.0
72%
21.0
4%
16.0
3%
29.0
5%
17.0
3%
19.0
3%
21.0
4%
10.0
2%
11.0
2%
7.0
1%

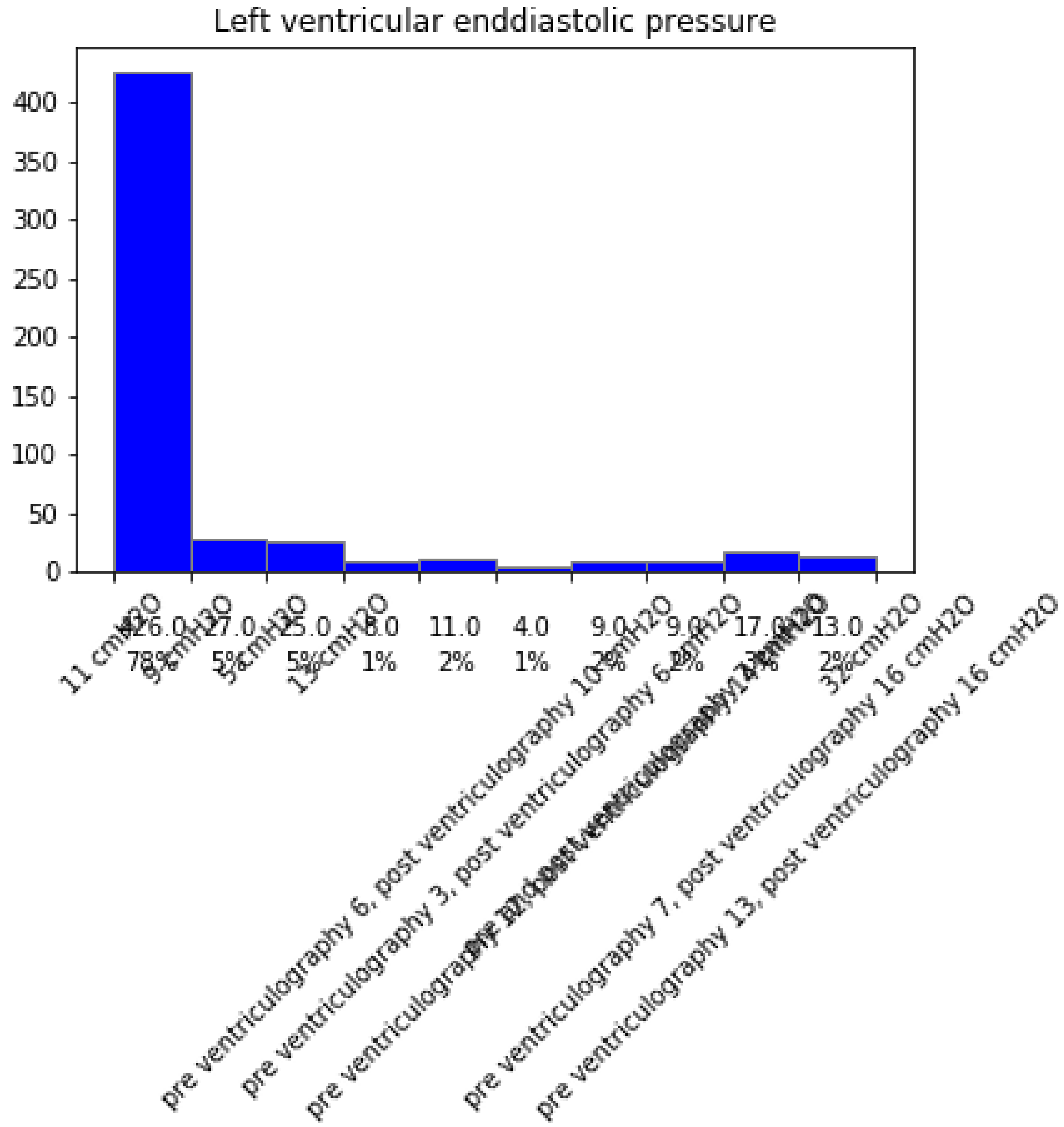

Cardiac output (load)
0
50
100
150
200
250
300
350
400
n/a
8,39 l/min
14,9 l/min
7,49 l/min
13,4 l/min
7,84 l/min
7,65 l/min
9,89 l/min
8,68 l/min
8,45 l/min
12,19 l/min
382.0
70%
19.0
3%
20.0
4%
23.0
4%
24.0
4%
25.0
5%
12.0
2%
6.0
1%
14.0
3%
24.0
4%

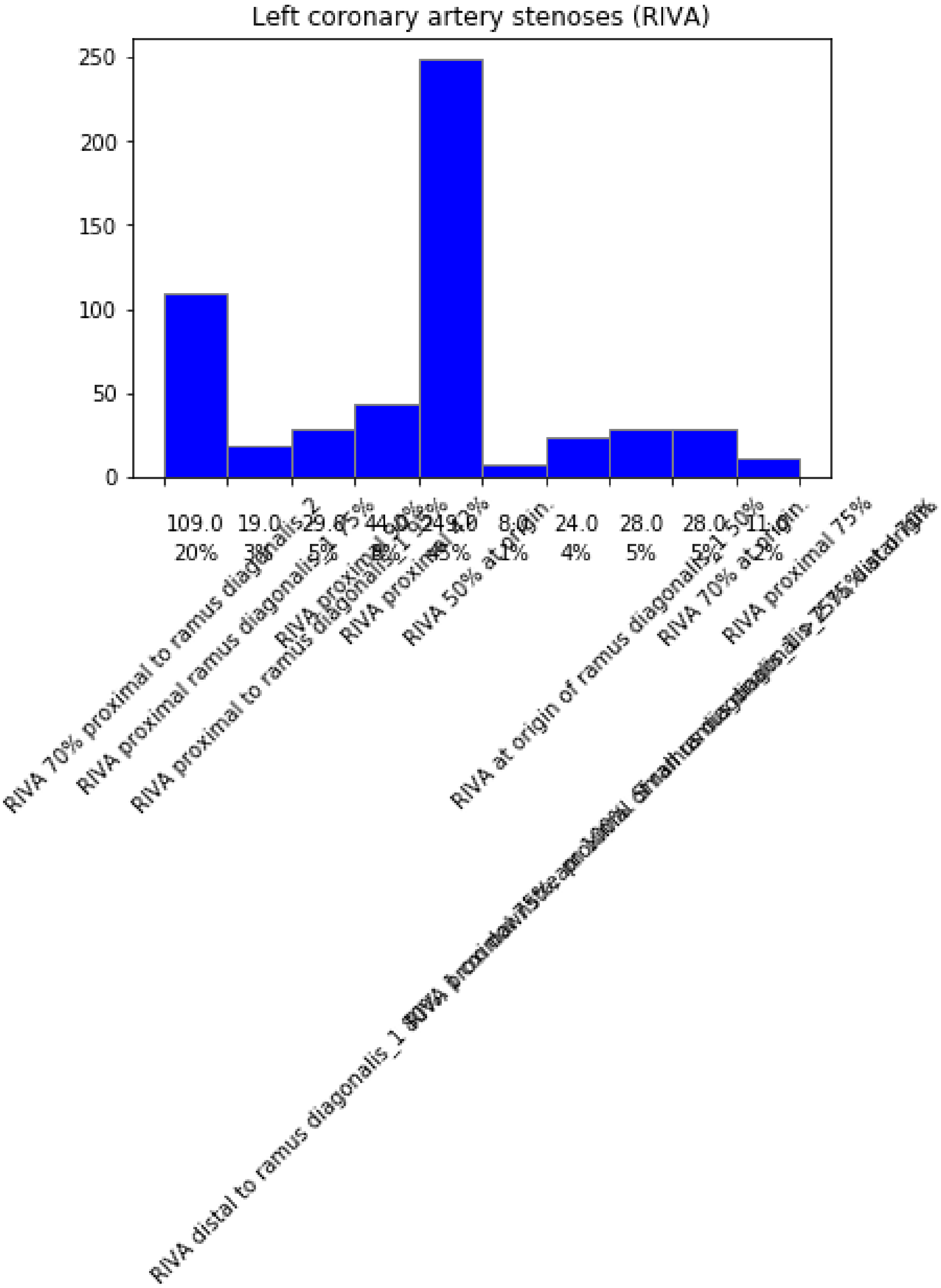

Cardiac index (load)
400
350
300
250
200
150
100
50
0
n/a
3,59 l/min/sqmBSA
7,5 l/min/sqmBSA
4,48 l/min/sqmBSA
6,33 l/min/sqmBSA
5,8 l/min/sqmBSA
7,03 l/min/sqmBSA
3,83 l/min/sqmBSA
5,9 l/min/sqmBSA
4,08 l/min/sqmBSA
5,86 l/min/sqmBSA
386.0
70%
25.0
5%
20.0
4%
21.0
4%
21.0
4%
24.0
4%
8.0
1%
6.0
1%
14.0
3%
24.0
4%

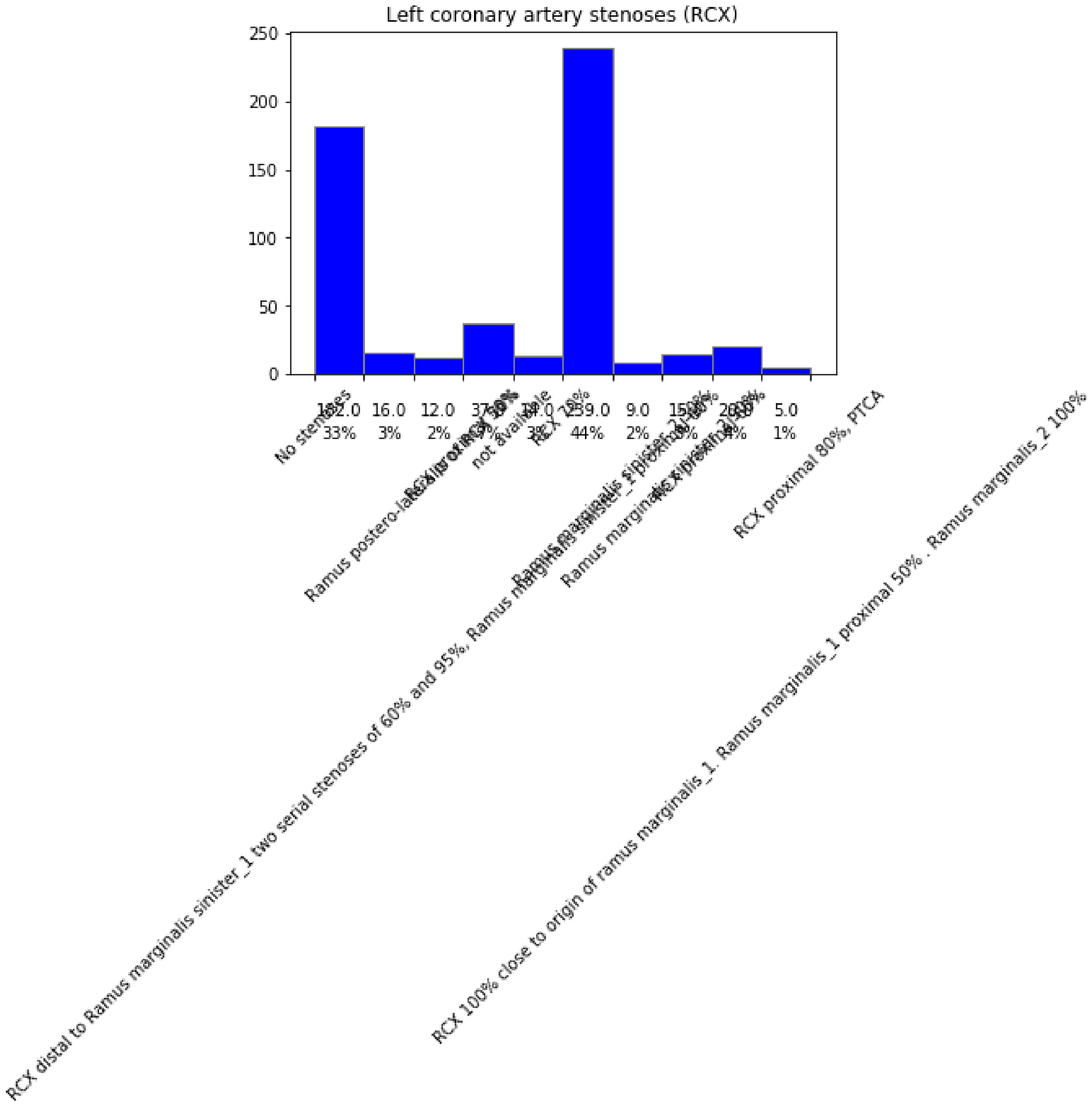

Stroke volume index (load)
0
50
100
150
200
250
300
350
400
n/a
36,3 ml/beat
70 ml/beat
39,2 ml/beat
53,6 ml/beat
38,7 ml/beat
72,7 ml/beat
31,2 ml/beat
37,6 ml/beat
65 ml/beat
44,4 ml/beat
387.0
70%
23.0
4%
23.0
4%
21.0
4%
16.0
3%
23.0
4%
9.0
2%
6.0
1%
17.0
3%
24.0
4%

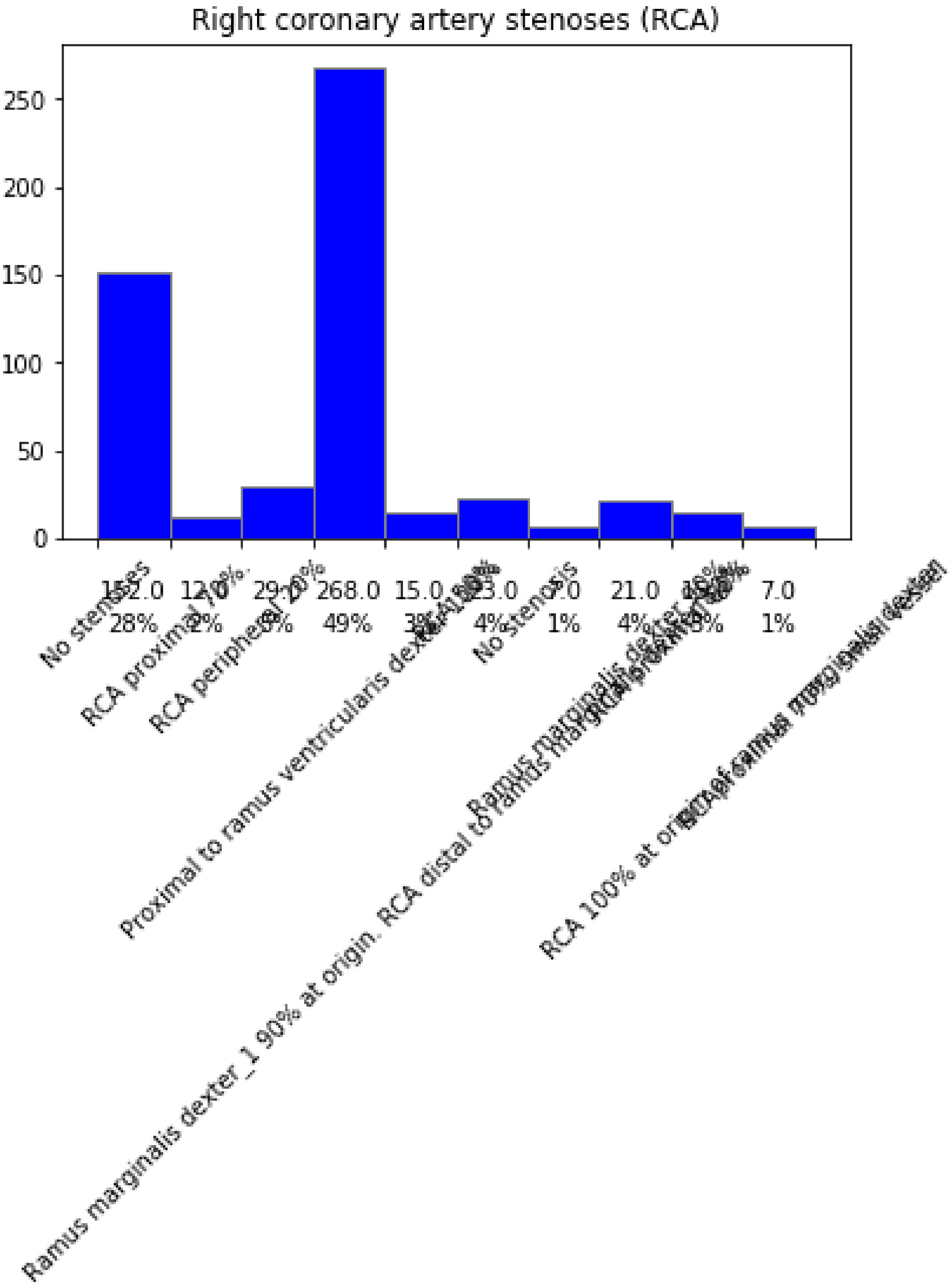

Aorta (at rest) (syst/diast)
0
100
200
300
400
500
160/64 cmH2O
90/54 cmH2O
106/69 cmH2O
90/58 cmH2O
170/80 cmH2O
100/57 cmH2O
132/62 cmH2O
160/78 cmH2O
120/64 cmH2O
100/56 cmH2O
116/68 cmH2O
487.0
89%
6.0
1%
4.0
1%
3.0
1%
4.0
1%
8.0
1%
12.0
2%
9.0
2%
8.0
1%
8.0
1%

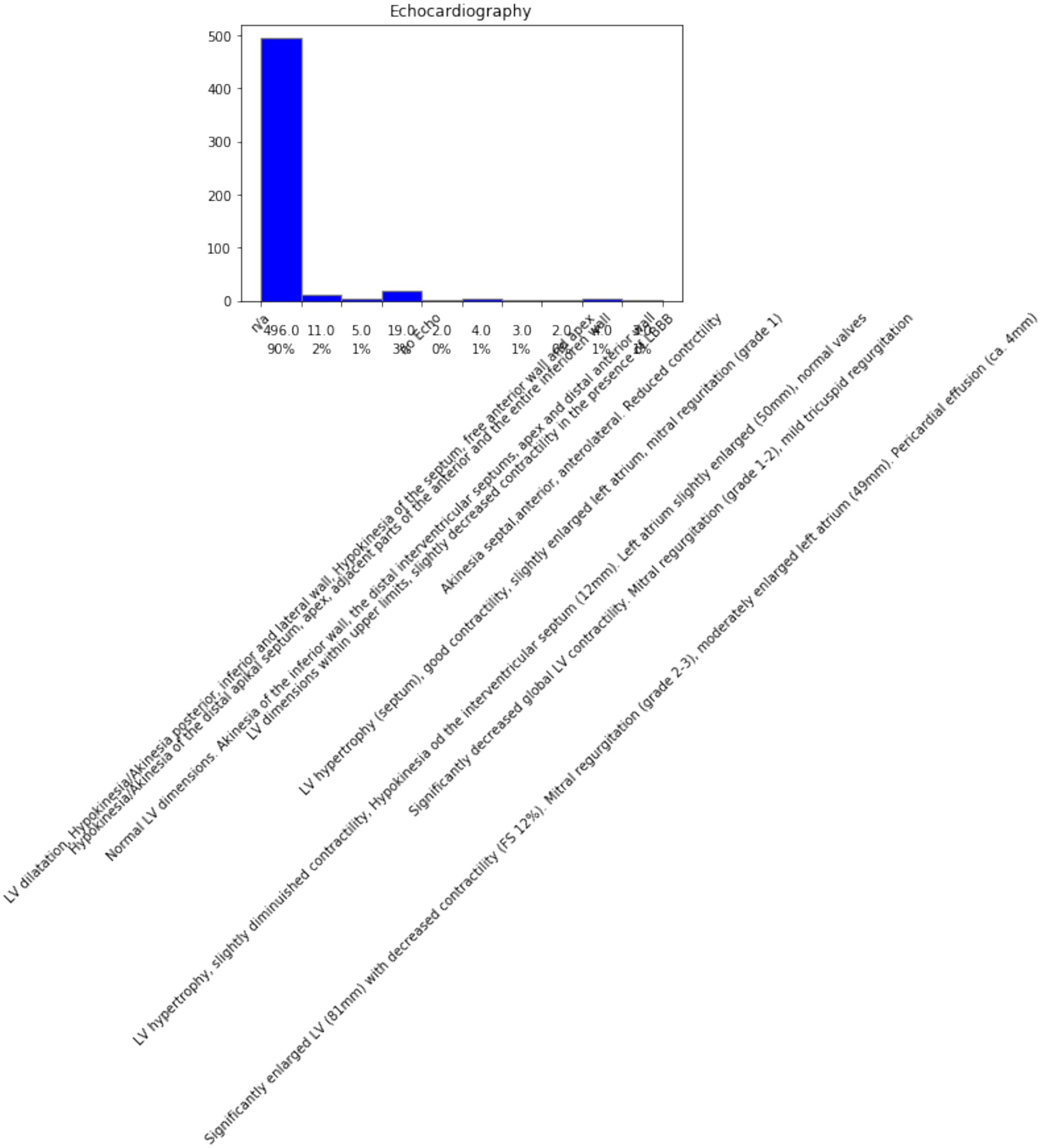

Aorta (at rest) mean
500
400
300
200
100
0
490.0
89%
5.0
1%
6.0
1%
2.0
0%
2.0
0%
7.0
1%
8.0
1%
8.0
1%
10.0
2%
11.0
2%
106 cmH2O
72 cmH2O
80 cmH2O
89 cmH2O
110 cmH2O
90 cmH2O
96 cmH2O
92 cmH2O
67 cmH2O
118 cmH2O
50 cmH2O

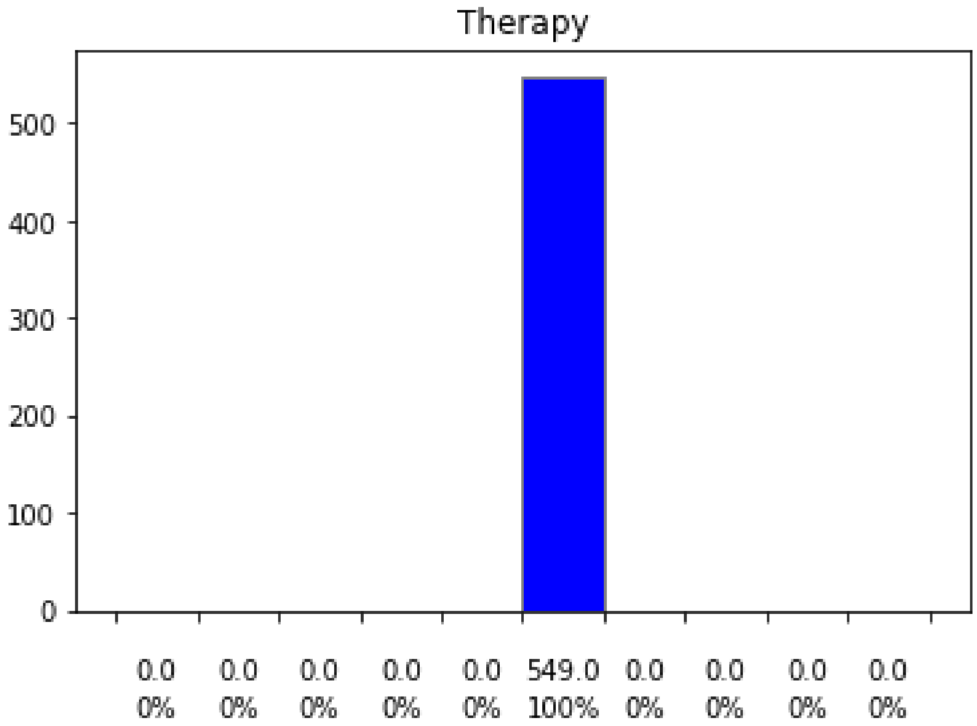

Left ventricular enddiastolic pressure
0
50
100
150
200
250
300
350
400
426.0
78%
27.0
5%
25.0
5%
8.0
1%
11.0
2%
4.0
1%
9.0
2%
9.0
2%
17.0
3%
13.0
2%
11 cmH2O
9 cmH2O
5 cmH2O
13 cmH2O
pre ventriculography 6, post ventriculography 10 cmH2O
pre ventriculography 3, post ventriculography 6 cmH2O
32 cmH2O
pre ventriculography 7, post ventriculography 16 cmH2O
pre ventriculography 13, post ventriculography 16 cmH2O

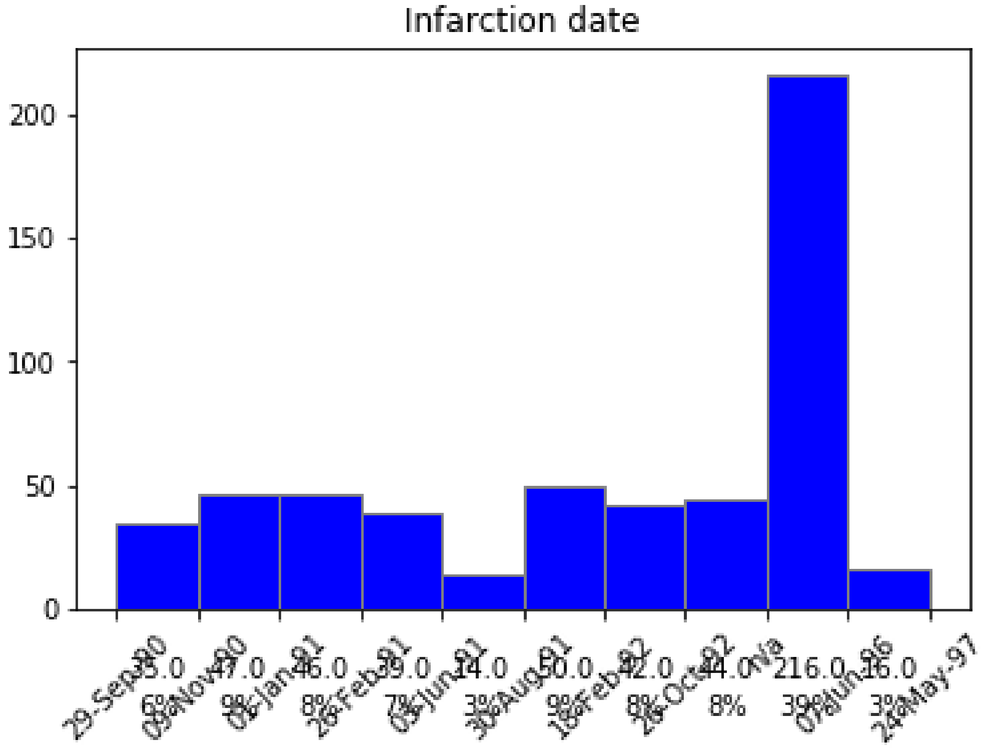
Left coronary artery stenoses (RIVA)
250
200
150
100
50
0
109.0
20%
19.0
3%
29.0
5%
44.0
8%
249.0
45%
8.0
1%
24.0
4%
28.0
5%
28.0
5%
11.0
2%
RIVA 70% proximal to ramus diagonalis_2
RIVA proximal ramus diagonalis_1 75%
RIVA proximal to ramus diagonalis_1 90%
RIVA proximal 90%
RIVA 50% at origin.
RIVA at origin of ramus diagonalis_1 50%
RIVA 70% at origin.
RIVA proximal 75%
RIVA distal to ramus diagonalis_1 50%

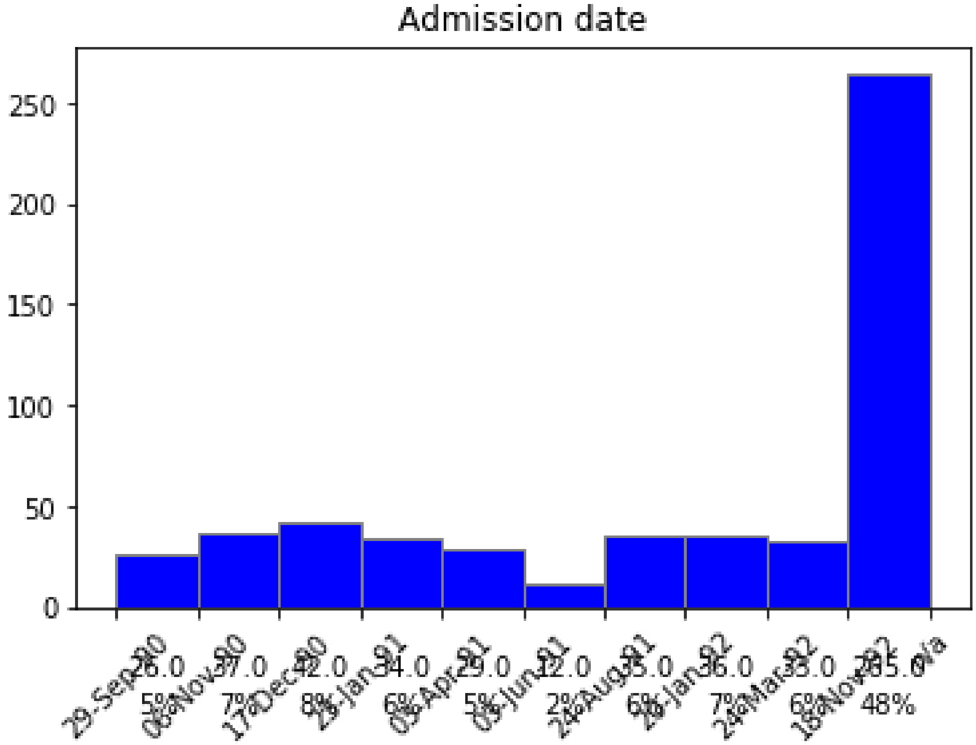

Left coronary artery stenoses (RCX)
250
200
150
100
50
0
182.0
33%
16.0
3%
12.0
2%
37.0
7%
14.0
3%
239.0
44%
9.0
2%
15.0
3%
20.0
4%
5.0
1%
No stenoses
Ramus postero-lateralis
RCX proximal 30%
not available
RCX 10%
RCX distal to Ramus marginalis sinister_1 two serial stenoses of 60% and 95%, Ramus marginalis sinister_1 proximal
Ramus marginalis sinister_1 proximal
Ramus marginalis
RCX proximal
RCX proximal 80%, PTCA
RCX 100% close to origin of ramus marginalis_1. Ramus marginalis_1 proximal 50% . Ramus marginalis_2 100%

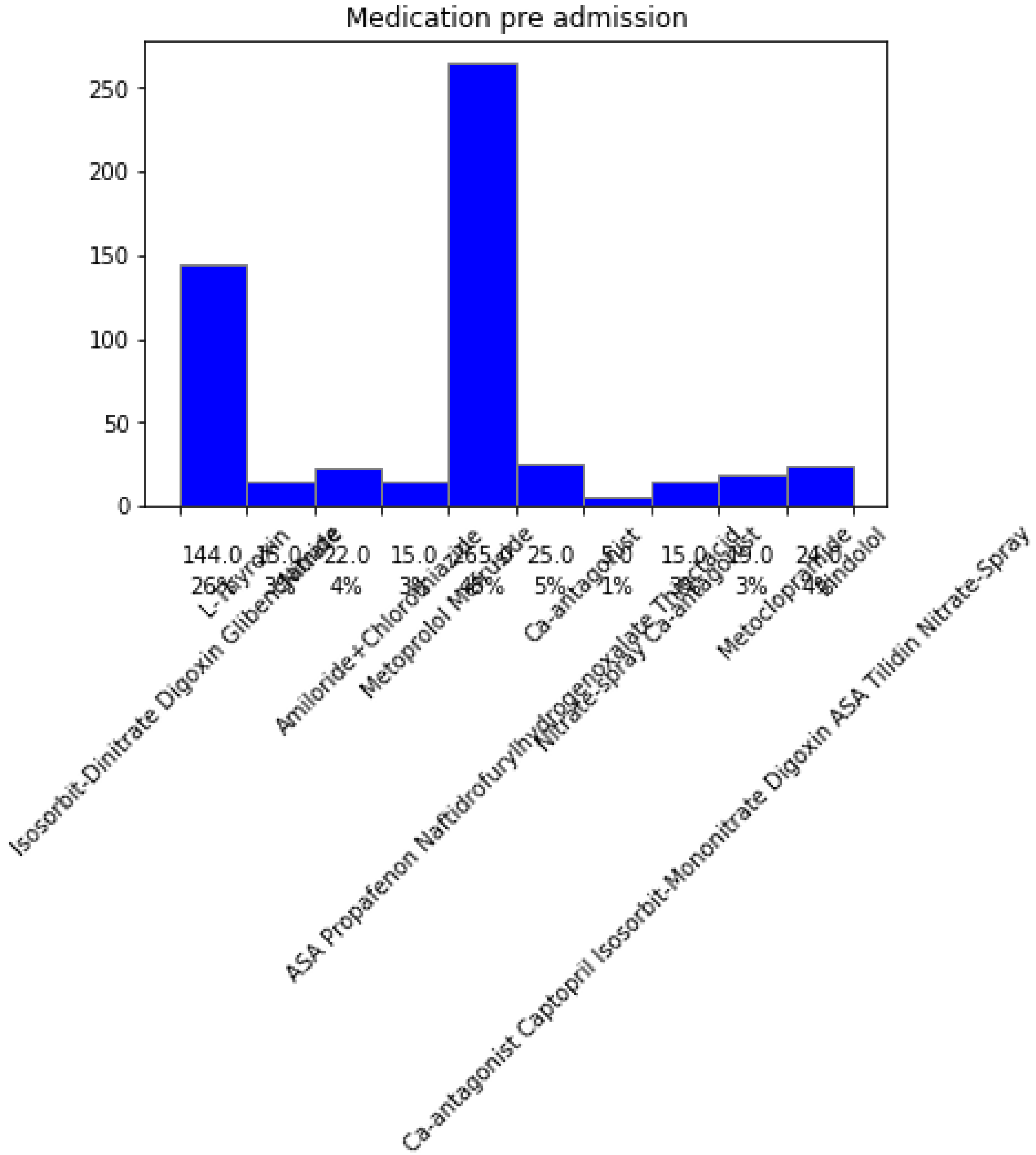

Right coronary artery stenoses (RCA)
0
50
100
150
200
250
152.0
28%
12.0
2%
29.0
5%
268.0
49%
15.0
3%
23.0
4%
7.0
1%
21.0
4%
3%
7.0
1%
No stenoses
RCA proximal 70%.
RCA peripheral 70%
Proximal to ramus ventricularis dexter
No stenosis
Ramus marginalis dexter_1 90% at origin. RCA distal to ramus marginalis dexter
Ramus marginalis dexter
RCA 100% at origin of ramus marginalis dexter

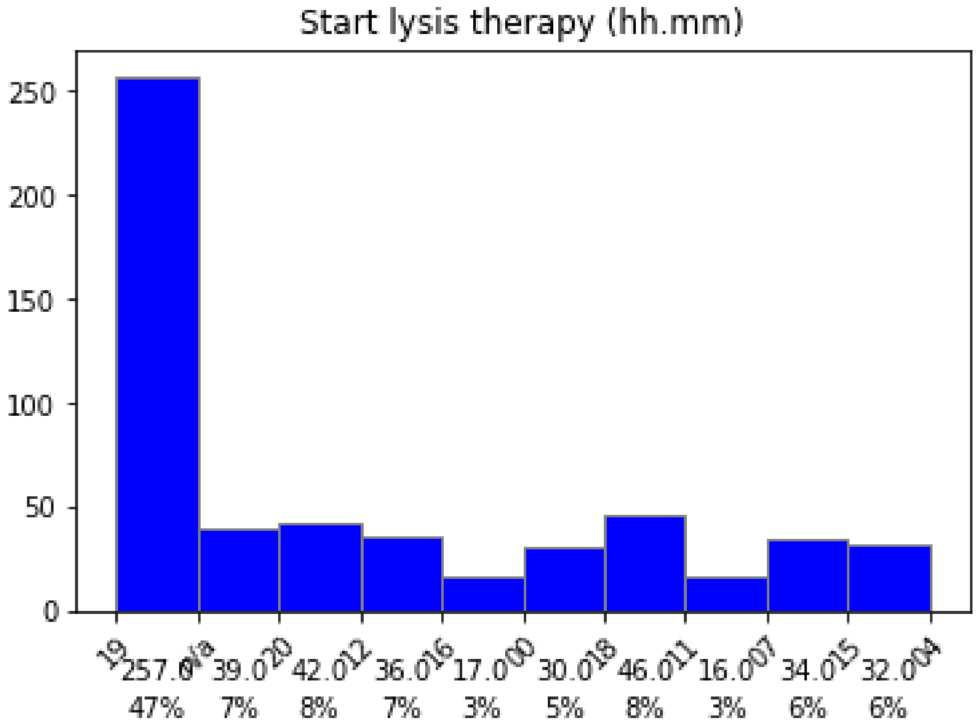

Echocardiography
500
400
300
200
100
0
496.0
90%
11.0
2%
5.0
1%
19.0
3%
2.0
0%
4.0
1%
3.0
1%
2.0
0%
4.0
1%
3.0
1%
n/a
LV dilatation, Hypokinesia/Akinesia posterior, inferior and lateral wall, Hypokinesia of the septum, free anterior wall and apex
Hypokinesia/Akinesia of the distal apikal septum, apex, adjacent parts of the anterior and the entire inferioren wall
no Echo
Normal LV dimensions. Akinesia of the inferior wall, the distal interventricular septums, apex and distal anterior wall
LV dimensions within upper limits, slightly decreased contractility in the presence of LBBB
Akinesia septal,anterior, anterolateral. Reduced contrctility
LV hypertrophy (septum), good contractility, slightly enlarged left atrium, mitral reguritation (grade 1)
LV hypertrophy, slightly diminished contractility, Hypokinesia od the interventricular septum (12mm). Left atrium slightly enlarged (50mm), normal valves
Significantly decreased global LV contractility. Mitral regurgitation (grade 1-2), mild tricuspid regurgitation
Significantly enlarged LV (81mm) with decreased contractility (FS 12%). Mitral regurgitation (grade 2-3), moderately enlarged left atrium (49mm). Pericardial effusion (ca. 4mm)

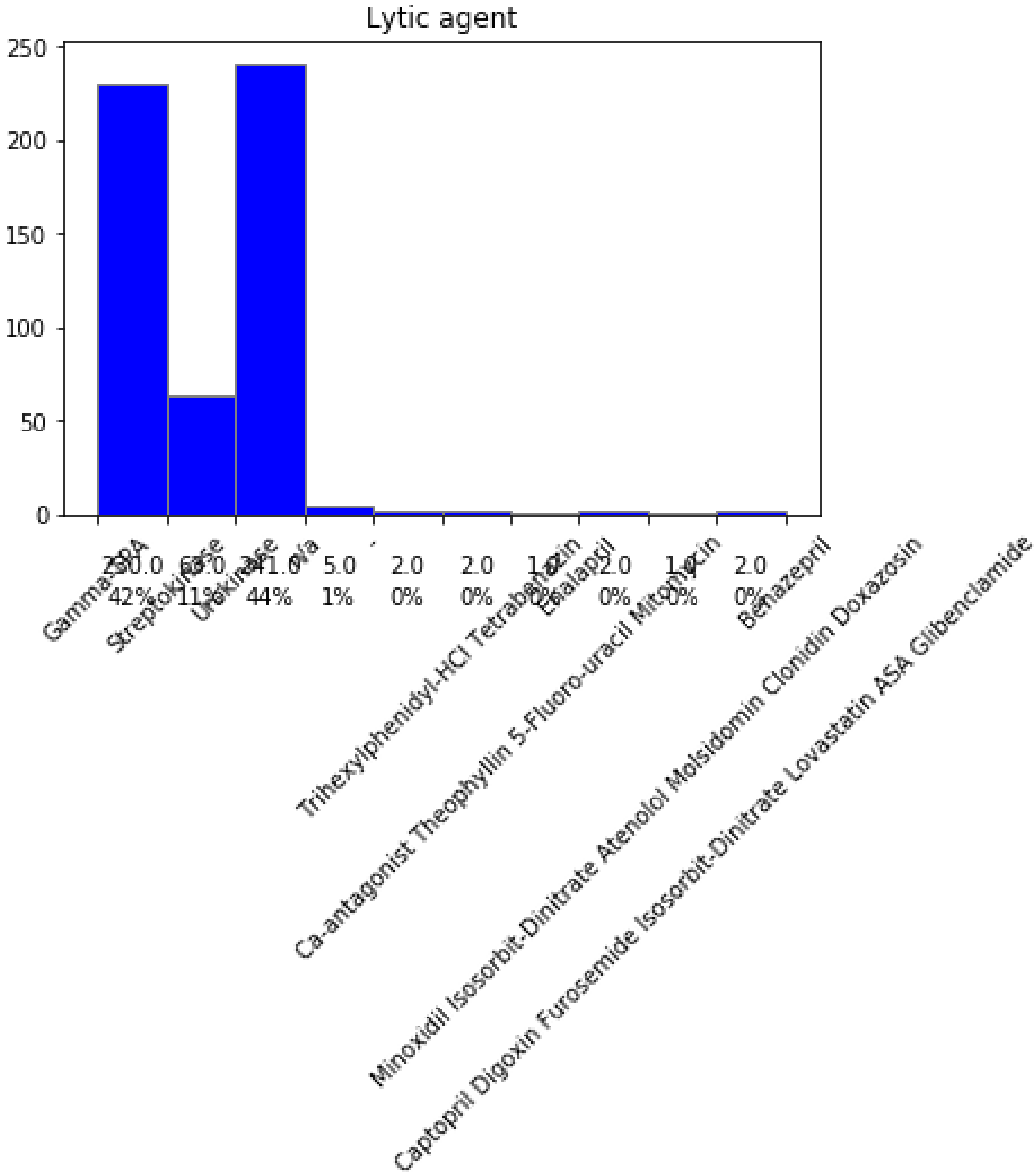

Therapy
500
400
300
200
100
0
0.0
0%
0.0
0%
0.0
0%
0.0
0%
0.0
0%
549.0
100%
0.0
0%
0.0
0%
0.0
0%
0.0
0%

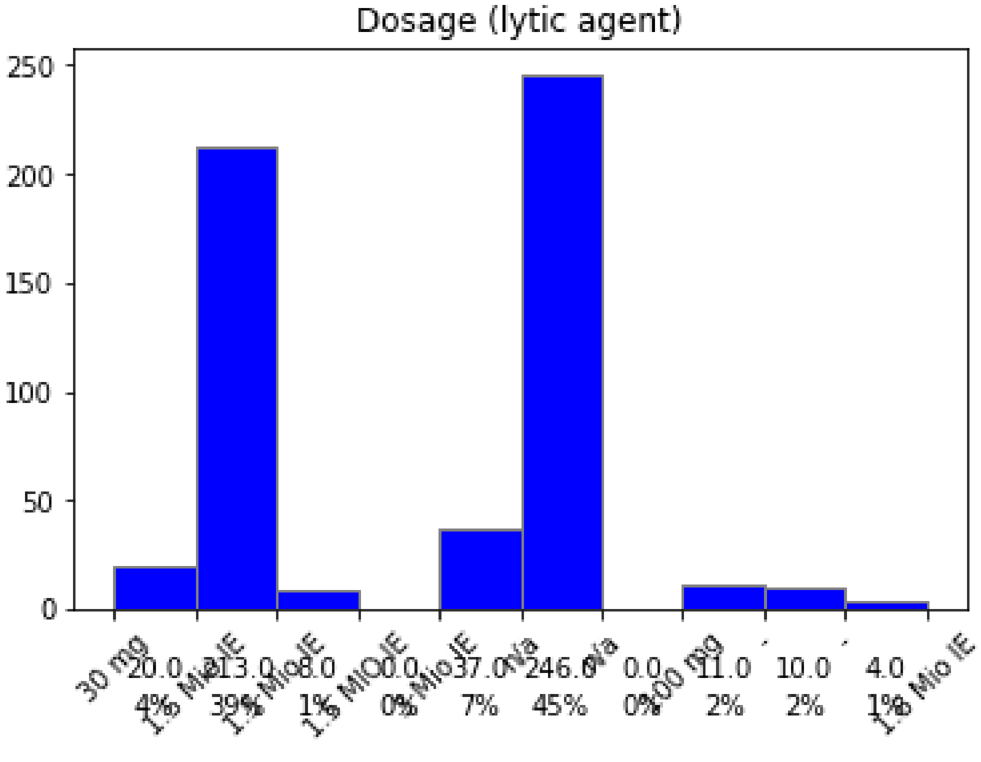

Infarction date
200
150
100
50
0
29-Sep-90
09-Nov-90
01-Jan-91
26-Feb-91
03-Jun-91
30-Aug-91
18-Feb-92
26-Oct-92
n/a
07-Jun-96
24-May-97
35.0
6%
47.0
9%
46.0
8%
39.0
7%
14.0
3%
50.0
9%
42.0
8%
44.0
8%
216.0
39%
16.0
3%

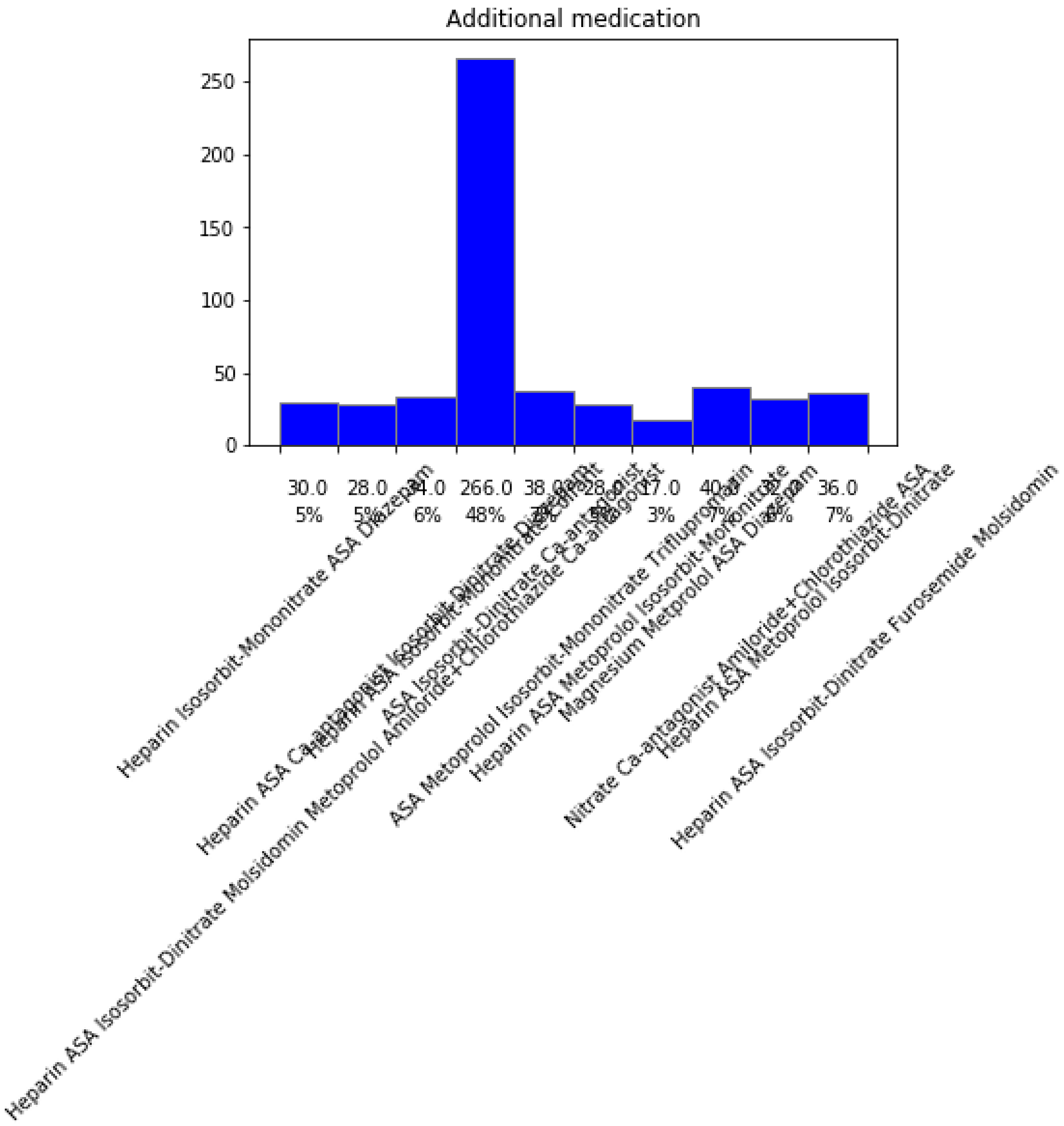

Admission date
250
200
150
100
50
0
29-Sep-90
06-Nov-90
17-Dec-90
29-Jan-91
05-Apr-91
05-Jun-91
24-Aug-91
29-Jan-92
24-Mar-92
18-Nov-92
n/a
26.0
5%
37.0
7%
42.0
8%
34.0
6%
29.0
5%
12.0
2%
35.0
6%
36.0
7%
33.0
6%
265.0
48%


Medication pre admission
0
50
100
150
200
250
144.0
26%
15.0
3%
22.0
4%
15.0
3%
265.0
48%
25.0
5%
5.0
1%
15.0
3%
19.0
3%
24.0
4%
L-Thyroxin
Isosorbit-Dinitrate Digoxin Glibenclamide
Nitrate
Amiloride+Chlorothiazide
Metoprolol Mefruside
Ca-antagonist
ASA Propafenon Naftidrofurylhydrogenoxalate Thioctacid
Nitrate-Spray Ca-antagonist
Metoclopramide
Pindolol
Ca-antagonist Captopril Isosorbit-Mononitrate Digoxin ASA Tilidin Nitrate-Spray

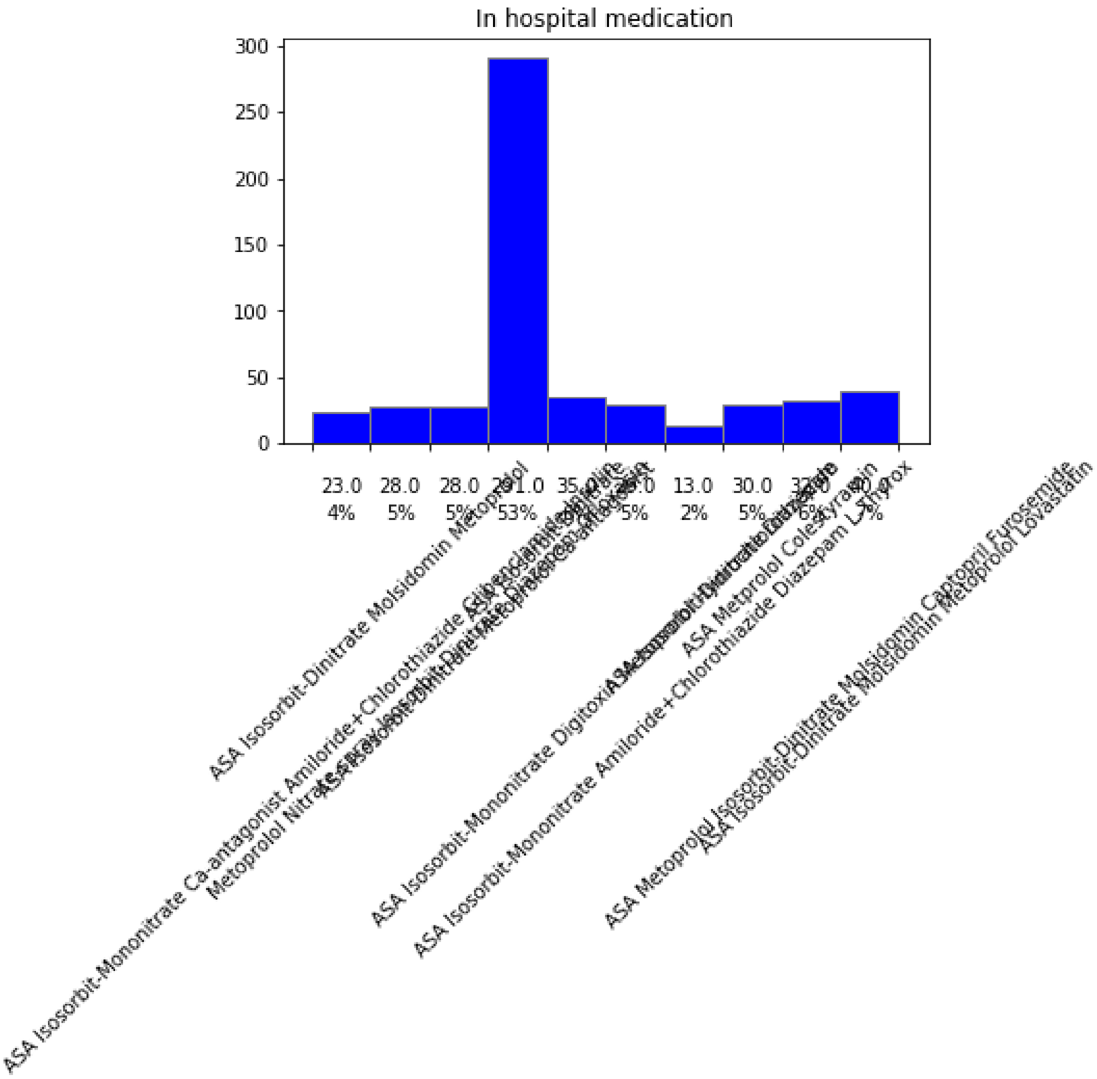

Start lysis therapy (hh.mm)
250
200
150
100
50
0
19
n/a
20
12
16
00
18
11
07
15
04
257.0
47%
39.0
7%
42.0
8%
36.0
7%
17.0
3%
30.0
5%
46.0
8%
16.0
3%
34.0
6%
32.0
6%

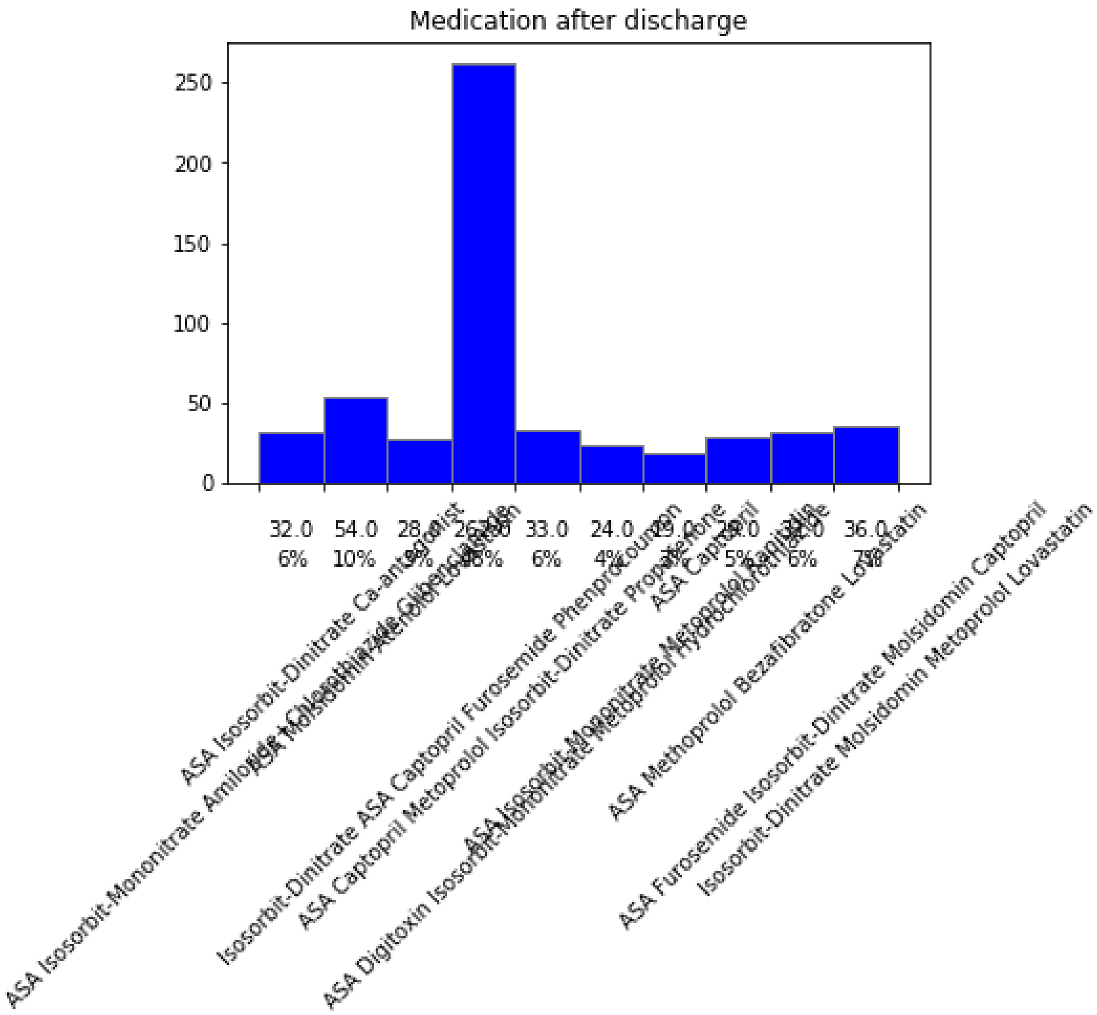

Lytic agent
250
200
150
100
50
0
Gamma-tPA
Streptokinase
Urokinase
n/a
'
Trihexylphenidyl-HCl Tetrabenazin
Ca-antagonist Theophyllin 5-Fluoro-uracil Mitomycin
Enalapril
Minoxidil Isosorbit-Dinitrate Atenolol Molsidomin Clonidin Doxazosin
Captopril Digoxin Furosemide Isosorbit-Dinitrate Lovastatin ASA Glibenclamide
Benazepril
230.0
42%
63.0
11%
241.0
44%
5.0
1%
2.0
0%
2.0
0%
1.0
0%
2.0
0%
1.0
0%
2.0
0%


Dosage (lytic agent)
250
200
150
100
50
0
30 mg
1.5 Mio IE
1.5 Mio IE
1.5 MIO IE
Mio IE
n/a
n/a
100 mg
-
-
1.5 Mio IE
20.0
4%
213.0
39%
8.0
1%
0.0
0%
37.0
7%
246.0
45%
0.0
0%
11.0
2%
10.0
2%
4.0
1%



Additional medication
250
200
150
100
50
0
30.0
5%
28.0
5%
34.0
6%
266.0
48%
38.0
7%
28.0
5%
17.0
3%
40.0
7%
32.0
6%
36.0
7%
Heparin Isosorbit-Mononitrate ASA Diazepam
Heparin ASA Ca-antagonist Isosorbit-Dinitrate Diazepam
Heparin ASA Isosorbit-Mononitrate Digitoxin
Heparin ASA Isosorbit-Dinitrate Molsidomin Metoprolol Amiloride+Chlorothiazide Ca-antagonist
ASA Isosorbit-Dinitrate Ca-antagonist
ASA Metoprolol Isosorbit-Mononitrate Triflupromazin
Heparin ASA Metoprolol Isosorbit-Mononitrate
Magnesium Metprolol ASA Diazepam
Nitrate Ca-antagonist Amiloride+Chlorothiazide ASA
Heparin ASA Metoprolol Isosorbit-Dinitrate
Heparin ASA Isosorbit-Dinitrate Furosemide Molsidomin


In hospital medication
0
50
100
150
200
250
300
23.0
4%
28.0
5%
28.0
5%
291.0
53%
35.0
6%
29.0
5%
13.0
2%
30.0
5%
32.0
6%
40.0
7%
ASA Isosorbit-Dinitrate Molsidomin Metoprolol
ASA Isosorbit-Mononitrate Ca-antagonist Amiloride+Chlorothiazide Glibenclamide
Metoprolol Nitrate spray Isosorbit-Dinitrate
ASA Isosorbit-Dinitrate Metoprolol Ca-antagonist
ASA Isosorbit-Mononitrate Digitoxin
ASA Metprolol Colestyramin
ASA Isosorbit-Mononitrate Amiloride+Chlorothiazide Diazepam L-Thyrox
ASA Metoprolol Isosorbit-Dinitrate Molsidomin Captopril Furosemide
ASA Isosorbit-Dinitrate Molsidomin Metoprolol Lovastatin



[53]:
```
from collections import Counter
k = "Reason for admission"
class_counts = Counter([c[k] for c in comment_data])
print(class_counts)
```

```
Counter({'Myocardial infarction': 368, 'Healthy control': 80, 'n/a': 27,
'Cardiomyopathy': 17, 'Bundle branch block': 17, 'Dysrhythmia': 16,
'Hypertrophy': 7, 'Valvular heart disease': 6, 'Myocarditis': 4, 'Stable
angina': 2, 'Heart failure (NYHA 2)': 1, 'Heart failure (NYHA 3)': 1, 'Heart
failure (NYHA 4)': 1, 'Palpitation': 1, 'Unstable angina': 1})
```

[33]:
```
def plot(data, title): #https://stackoverflow.com/questions/6352740/
↪matplotlib-label-each-bin
    import matplotlib.pyplot as plt
    import numpy as np
    from matplotlib.ticker import FormatStrFormatter

    fig, ax = plt.subplots()
    counts, bins, patches = ax.hist(data, facecolor='blue', edgecolor='gray')
    ax.set_title(title)
    # Set the ticks to be at the edges of the bins.
    ax.set_xticks(bins)
    # Set the xaxis's tick labels to be formatted with 1 decimal place...
    for tick in ax.get_xticklabels():
        tick.set_rotation(45)
    # Label the raw counts and the percentages below the x-axis...
    bin_centers = 0.5 * np.diff(bins) + bins[:-1]
    for count, x in zip(counts, bin_centers):
        # Label the raw counts
        ax.annotate(str(count), xy=(x, 0), xycoords=('data', 'axes fraction'),
            xytext=(0, -18), textcoords='offset points', va='top', ha='center')

        # Label the percentages
        percent = '%0.0f%%' % (100 * float(count) / counts.sum())
        ax.annotate(percent, xy=(x, 0), xycoords=('data', 'axes fraction'),
            xytext=(0, -32), textcoords='offset points', va='top', ha='center')


    # Give ourselves some more room at the bottom of the plot
    plt.subplots_adjust(bottom=0.15)
    plt.show()
```

Plotting (does not seem to work for annotations)

[1]:
```
#wfdb.plot_wfdb(record=file_data[0], plot_sym=True, time_units='samples',␣
 ↪title='Test', figsize=(10,4), ecg_grids='all')
```

Method for partitioning data into windows with a specific overlap Warning: overlaps other than 0.5 have not been tested. Might also behave unexpected when ‘shift’ is not an Integer

[6]:
```
def partition_data(data, window_size=3000, overlap=0.5, store_in_array=True,␣
 ↪align_right=True, verbose=True): #maybe allow non float overlap too
    samples, channels = data.shape
    if samples < window_size:
        print('too few samples (%d) to support window size of %d' % (samples,␣
 ↪window_size))
        return None
    if verbose: print('Input data has shape:', data.shape)
    shift = window_size*overlap
    offset = int(samples % shift)
    if align_right:
        used_data = data[offset:]
    else:
        used_data = data[:-offset]
    samples, _ = used_data.shape
    if verbose: print('The window of size %d will be shifted by %f. The total␣
 ↪data used is %d' % (window_size, shift, samples))
    partitioned = np.empty((int(samples/shift)-1, window_size, channels))
    if verbose: print('The partitioned data now has shape:', partitioned.shape)
    for i in range(len(partitioned)):
        index = int(i*shift)
        partitioned[i, :, :] = used_data[index:index+window_size, :]
    return partitioned
```

[ ]:

```
partition_data(d) #nur zum testen
```

[8]:

```
#testing
print(partition_data(np.repeat(np.arange(10)[np.newaxis, :].T, 5, axis=1),␣
 ↪window_size=4))
print(partition_data(np.repeat(np.arange(11)[np.newaxis, :].T, 5, axis=1),␣
 ↪window_size=4))
print(partition_data(np.repeat(np.arange(9)[np.newaxis, :].T, 5, axis=1),␣
 ↪window_size=4))
print(partition_data(np.repeat(np.arange(10)[np.newaxis, :].T, 5, axis=1),␣
 ↪window_size=10))
```

```
Input data has shape: (10, 5)
The window of size 4 will be shifted by 2.000000. The total data used is 10
The partitioned data now has shape: (4, 4, 5)
[[[0. 0. 0. 0. 0.]
  [1. 1. 1. 1. 1.]
  [2. 2. 2. 2. 2.]
  [3. 3. 3. 3. 3.]]

 [[2. 2. 2. 2. 2.]
  [3. 3. 3. 3. 3.]
  [4. 4. 4. 4. 4.]
  [5. 5. 5. 5. 5.]]

 [[4. 4. 4. 4. 4.]
  [5. 5. 5. 5. 5.]
  [6. 6. 6. 6. 6.]
  [7. 7. 7. 7. 7.]]

 [[6. 6. 6. 6. 6.]
  [7. 7. 7. 7. 7.]
  [8. 8. 8. 8. 8.]
  [9. 9. 9. 9. 9.]]]
Input data has shape: (11, 5)
```

```
The window of size 4 will be shifted by 2.000000. The total data used is 10
The partitioned data now has shape: (4, 4, 5)
[[[ 1.  1.  1.  1.  1.]
  [ 2.  2.  2.  2.  2.]
  [ 3.  3.  3.  3.  3.]
  [ 4.  4.  4.  4.  4.]]

 [[ 3.  3.  3.  3.  3.]
  [ 4.  4.  4.  4.  4.]
  [ 5.  5.  5.  5.  5.]
  [ 6.  6.  6.  6.  6.]]

 [[ 5.  5.  5.  5.  5.]
  [ 6.  6.  6.  6.  6.]
  [ 7.  7.  7.  7.  7.]
  [ 8.  8.  8.  8.  8.]]

 [[ 7.  7.  7.  7.  7.]
  [ 8.  8.  8.  8.  8.]
  [ 9.  9.  9.  9.  9.]
  [10. 10. 10. 10. 10.]]]
Input data has shape: (9, 5)
The window of size 4 will be shifted by 2.000000. The total data used is 8
The partitioned data now has shape: (3, 4, 5)
[[[1. 1. 1. 1. 1.]
  [2. 2. 2. 2. 2.]
  [3. 3. 3. 3. 3.]
  [4. 4. 4. 4. 4.]]

 [[3. 3. 3. 3. 3.]
  [4. 4. 4. 4. 4.]
  [5. 5. 5. 5. 5.]
  [6. 6. 6. 6. 6.]]

 [[5. 5. 5. 5. 5.]
  [6. 6. 6. 6. 6.]
  [7. 7. 7. 7. 7.]
  [8. 8. 8. 8. 8.]]]
Input data has shape: (10, 5)
The window of size 10 will be shifted by 5.000000. The total data used is 10
The partitioned data now has shape: (1, 10, 5)
[[[0. 0. 0. 0. 0.]
  [1. 1. 1. 1. 1.]
  [2. 2. 2. 2. 2.]
  [3. 3. 3. 3. 3.]
  [4. 4. 4. 4. 4.]
  [5. 5. 5. 5. 5.]
  [6. 6. 6. 6. 6.]
 [7. 7. 7. 7. 7.]
 [8. 8. 8. 8. 8.]
 [9. 9. 9. 9. 9.]]]
```

Generate a (hopefully correct) context task for any given partitioned data N : The number of samples generated for each current window. If this number is too high (No more windows to randomly sample from) numpy will throw an error.

observations : The number of windows you have to predict the future.

predictions: The number of windows that will be predicted.

Total amount of data output will be (*Number of windows* - *observations* - *predictions*) * N

[7]:
```
def generate_context_task(partitioned_data, N, observations=5, predictions=3,␣
 ↪verbose=True, shuffle_all=False): #maybe use shuffle?
    #generate N-1 negative samples and 1 positive sample:
    windows, samples, channels = partitioned_data.shape
    positive_samples_x = []
    positive_samples_y = []
    negative_samples_x = []
    negative_samples_y = []
    for i in range(0, windows - observations - predictions):
        i_cur = i+observations
        positive_samples_x += [partitioned_data[i:i_cur, :, :]]
        positive_samples_y += [partitioned_data[i_cur:i_cur+predictions]]
        possible_choices = list(range(i))+list(range(i_cur+predictions,␣
 ↪windows))get_file_list(BASE_DIR)
        for _ in range(N-1):
            choices = np.random.choice(possible_choices, predictions,␣
 ↪replace=False) #advanced use p param for different probabilities
            negative_samples_x += [partitioned_data[i:i_cur, :, :]]
            negative_samples_y += [partitioned_data[choices, :, :]]

    return positive_samples_x, positive_samples_y, negative_samples_x,␣
 ↪negative_samples_y
```

[10]:
```
#Testing
px, py, nx, ny = generate_context_task(partition_data(d), 10)
```

```
Input data has shape: (120012, 15)
The window of size 3000 will be shifted by 1500.000000. The total data used is
120000
The partitioned data now has shape: (79, 3000, 15)
```

[121]:
```
(len(px), len(py)), (len(nx), len(ny))
```

[121]:
```
((71, 71), (639, 639))
```

[12]:

```
px[0].shape, py[0].shape
```

[12]:

```
((5, 3000, 15), (3, 3000, 15))
```

[18]:

```
def convert_dat_to_h5(storage_path, dat_file_paths, window_size=3000, overlap=0.
↪5, align_right=True, verbose=True):
    if not os.path.exists(storage_path):
        os.makedirs(storage_path)
    for f in dat_file_paths:
        data = read_signal(os.path.join(BASE_DIR, f))
        print(f)
        partitioned = partition_data(data, window_size, overlap, True,␣
↪align_right, verbose)
        target = os.path.join(storage_path, f.replace('/', '-') +'.h5')
        with h5py.File(target, 'w') as wf:
            wf['windows'] = partitioned
            wf.flush()
            if verbose: print(target, 'file created and written. %d windows␣
↪saved.' % (len(partitioned)))
```

[ ]:

```
record_files = get_file_list(BASE_DIR)
convert_dat_to_h5('test', record_files)
```

[17]:

```
BASE_DIR = '/media/julian/Volume/data/ECG/ptb-diagnostic-ecg-database-1.0.0/'
record_files = get_file_list(BASE_DIR, relative=False)

file_data = []
success = 0
for f in record_files:
    header_record = read_header(f)
    print(header_record.comments)
    break
```

```
['age: 81', 'sex: female', 'ECG date: 01/10/1990', 'Diagnose:', 'Reason for
admission: Myocardial infarction', 'Acute infarction (localization): infero-
latera', 'Former infarction (localization): no', 'Additional diagnoses: Diabetes
mellitus', 'Smoker: no', 'Number of coronary vessels involved: 1', 'Infarction
date (acute): 29-Sep-90', 'Previous infarction (1) date: n/a', 'Previous
infarction (2) date: n/a', 'Hemodynamics:', 'Catheterization date: 16-Oct-90',
'Ventriculography: Akinesia inferior wall', 'Chest X-ray: Heart size upper limit
of norm', 'Peripheral blood Pressure (syst/diast):  140/80 mmHg', 'Pulmonary
artery pressure (at rest) (syst/diast): n/a', 'Pulmonary artery pressure (at
rest) (mean): n/a', 'Pulmonary capillary wedge pressure (at rest): n/a',
'Cardiac output (at rest): n/a', 'Cardiac index (at rest): n/a', 'Stroke volume
index (at rest): n/a', 'Pulmonary artery pressure (laod) (syst/diast): n/a',
'Pulmonary artery pressure (laod) (mean): n/a', 'Pulmonary capillary wedge
pressure (load): n/a', 'Cardiac output (load): n/a', 'Cardiac index (load):
```

```
n/a', 'Stroke volume index (load): n/a', 'Aorta (at rest) (syst/diast): 160/64
cmH2O', 'Aorta (at rest) mean: 106 cmH2O', 'Left ventricular enddiastolic
pressure: 11 cmH2O', 'Left coronary artery stenoses (RIVA): RIVA 70% proximal to
ramus diagonalis_2', 'Left coronary artery stenoses (RCX): No stenoses', 'Right
coronary artery stenoses (RCA): No stenoses', 'Echocardiography: n/a',
'Therapy:', 'Infarction date: 29-Sep-90', 'Catheterization date: 16-Oct-90',
'Admission date: 29-Sep-90', 'Medication pre admission: Isosorbit-Dinitrate
Digoxin Glibenclamide', 'Start lysis therapy (hh.mm): 19:45', 'Lytic agent:
Gamma-TPA', 'Dosage (lytic agent): 30 mg', 'Additional medication: Heparin
Isosorbit-Mononitrate ASA Diazepam', 'In hospital medication: ASA Isosorbit-
Mononitrate Ca-antagonist Amiloride+Chlorothiazide Glibenclamide Insulin',
'Medication after discharge: ASA Isosorbit-Mononitrate Amiloride+Chlorothiazide
Glibenclamide']
```

[44]:

```python
import datetime
import time
str(datetime.datetime.now().strftime("%d_%m_%y-%H"))
```

[44]:

```
'19_11_20-15'
```

[81]:

```python
def parse_comment_dict(wfdb_comment):
    comment_map = {}
    for c in wfdb_comment:
        e = c.lower().split(':')
        comment_map[e[0]] = e[1].strip()
    return comment_map
label_mappings = {}
def onehot_comment(wfdb_comment, terms: list, key_function_dict:dict = None):
    onehots = []
    if len(terms) > 0:
        terms_encoded = np.zeros(len(terms), dtype=bool)
        for i, term in enumerate(terms):
            for c in wfdb_comment:
                if term in c:
                    terms_encoded[i] = True
        onehots.append(terms_encoded)
    comment_dict = parse_comment_dict(wfdb_comment)
    for key, (func, n) in key_function_dict.items():
        if not key in label_mappings:
            label_mappings[key] = {}
        value = func(comment_dict[key])
        if not value in label_mappings[key]:
            label_mappings[key][value] = len(label_mappings[key])
        encoded_key = np.zeros(n, dtype=bool)
        encoded_key[label_mappings[key][value]] = value
        onehots.append(encoded_key)
    return np.concatenate(onehots)
```

```
def filter_comment(key, comment_string):
    c = comment_string
    if key == 'reason for admission':
        if 'cardiomyopathy' in c or 'heart failure' in c:
            return 'cardiomyopathy'
        elif 'n/a' in c or 'palpitation' in c:
            return 'miscellaneous'
        elif 'angina' in c:
            return 'angina'
    return comment_string
```

[82]:

```
from collections import defaultdict
default_key_function_dict = defaultdict(lambda y: (lambda x: True, 2))
default_key_function_dict['age'] = lambda x: int(x)>50 if x.isnumeric() else␣
 ↪False, 2 #age
default_key_function_dict['smoker'] = lambda x: x == 'yes', 2
default_key_function_dict['reason for admission'] = lambda x:␣
 ↪filter_comment('reason for admission', x), 10
```

[ ]:

```
BASE_DIR = '/media/julian/Volume/data/ECG/ptb-diagnostic-ecg-database-1.0.0/'
record_files = get_file_list(BASE_DIR, relative=False)

for f in record_files:
    wfdb_comment = read_header(f)
    print(onehot_comment(wfdb_comment, [], default_key_function_dict))
```

[85]:

```
label_mappings
```

[85]:

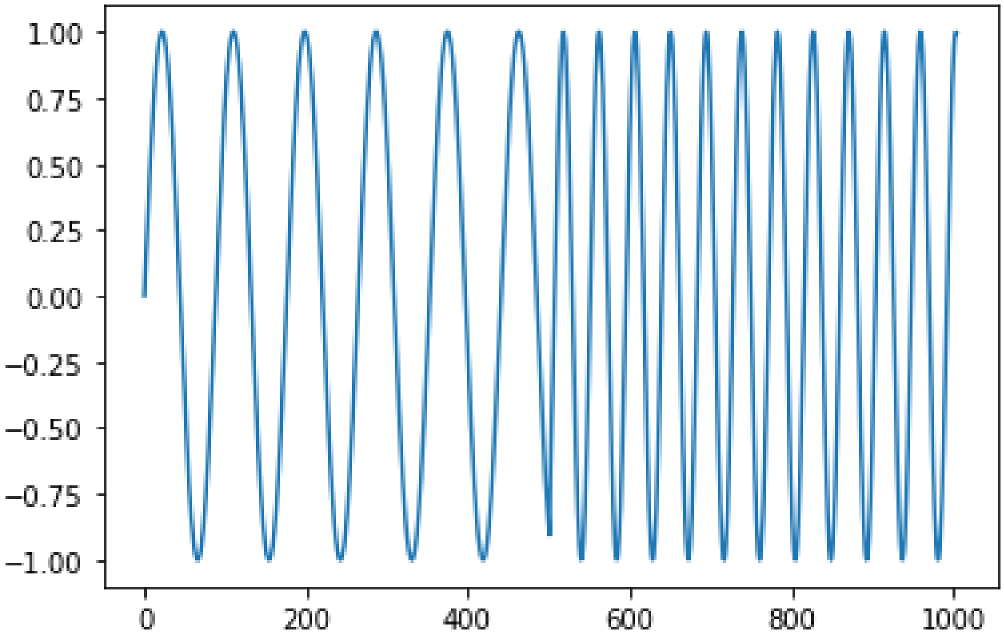
```
{'age': {True: 0, False: 1},
 'smoker': {False: 0, True: 1},
 'reason for admission': {'myocardial infarction': 0,
  'healthy control': 1,
  'valvular heart disease': 2,
  'dysrhythmia': 3,
  'cardiomyopathy': 4,
  'miscellaneous': 5,
  'angina': 6,
  'hypertrophy': 7,
  'bundle branch block': 8,
  'myocarditis': 9}}
```

[20]:

```
import numpy as np
def generate_sin_data(n, hz_low, hz_high):
    m_phase=0

    signal = np.empty(n)
    f = hz_low
    fs= 44100

    phaseInc = 2*np.pi*f/fs


    for i in range(int(n/2)):
        signal[i] = np.sin(m_phase)
        m_phase = m_phase + phaseInc

    m_phase = m_phase % 2*np.pi


    f=hz_high
    phaseInc = 2*np.pi*f/fs


    for i in range(int(n/2), n):
        signal[i] = np.sin(m_phase)
        m_phase = m_phase + phaseInc

    m_phase = m_phase % 2*np.pi
    return signal
```

[23]:

```
import matplotlib.pyplot as plt

plt.plot(generate_sin_data(1005, 500, 1000))
plt.show()
```



```
[19]: record_files = get_file_list(BASE_DIR, filter_function=lambda x: 'records500'␣
       ↪in x)
      print(len(record_files), 'files found')
```

```
21837 files found
```

```
[6]: import torch as t
```

```
[25]: tar = t.zeros((4,8))
      tar[0,1] = 1.
      tar
      y = tar
```

```
[26]: pred = torch.rand_like(tar)
      pred[:, 1:5] = 0.
      pred
      logits = pred
```

```
[24]: t.sum((pred != 0.0) | (tar != 0.0))
```

```
[24]: tensor(17)
```

```
[37]: t.sum(t.abs(y-pred) <= 0.000005)/(4*8)
```

```
[37]: tensor(0.4688)
```

[4]:

```python
import pandas as pd
import numpy as np
import wfdb
import ast

def load_raw_data(df, sampling_rate, path):
    if sampling_rate == 100:
        data = [wfdb.rdsamp(path+f) for f in df.filename_lr]
    else:
        data = [wfdb.rdsamp(path+f) for f in df.filename_hr]
    data = np.array([signal for signal, meta in data])
    return data

path = '/media/julian/Volume/data/ECG/
 ↪ptb-xl-a-large-publicly-available-electrocardiography-dataset-1.0.1/'
sampling_rate=500

# load and convert annotation data
Y = pd.read_csv(path+'ptbxl_database.csv', index_col='ecg_id')
Y.scp_codes = Y.scp_codes.apply(lambda x: ast.literal_eval(x))

# Load raw signal data
#X = load_raw_data(Y, sampling_rate, path)

# Load scp_statements.csv for diagnostic aggregation
agg_df = pd.read_csv(path+'scp_statements.csv', index_col=0)
agg_df = agg_df[agg_df.diagnostic == 1]

def aggregate_diagnostic(y_dic):
    tmp = []
    for key in y_dic.keys():
        if key in agg_df.index:
            tmp.append(agg_df.loc[key].diagnostic_class)
    return list(set(tmp))

# Apply diagnostic superclass
Y['diagnostic_superclass'] = Y.scp_codes.apply(aggregate_diagnostic)

# Split data into train and test
test_fold = 10
# Train
#X_train = X[np.where(Y.strat_fold != test_fold)]
y_train = Y[(Y.strat_fold != test_fold)].diagnostic_superclass
# Test
#X_test = X[np.where(Y.strat_fold == test_fold)]
y_test = Y[Y.strat_fold == test_fold].diagnostic_superclass
```

```
[5]: y_train
```

```
[5]: ecg_id
     1          [NORM]
     2          [NORM]
     3          [NORM]
     4          [NORM]
     5          [NORM]
                 …
     21833      [STTC]
     21834      [NORM]
     21835      [STTC]
     21836      [NORM]
     21837      [NORM]
     Name: diagnostic_superclass, Length: 19634, dtype: object
```