JOHNS HOPKINS UNIVERSITY

Track patient recovery in real-time by processing streaming data

BIOMEDICAL DATA DESIGN

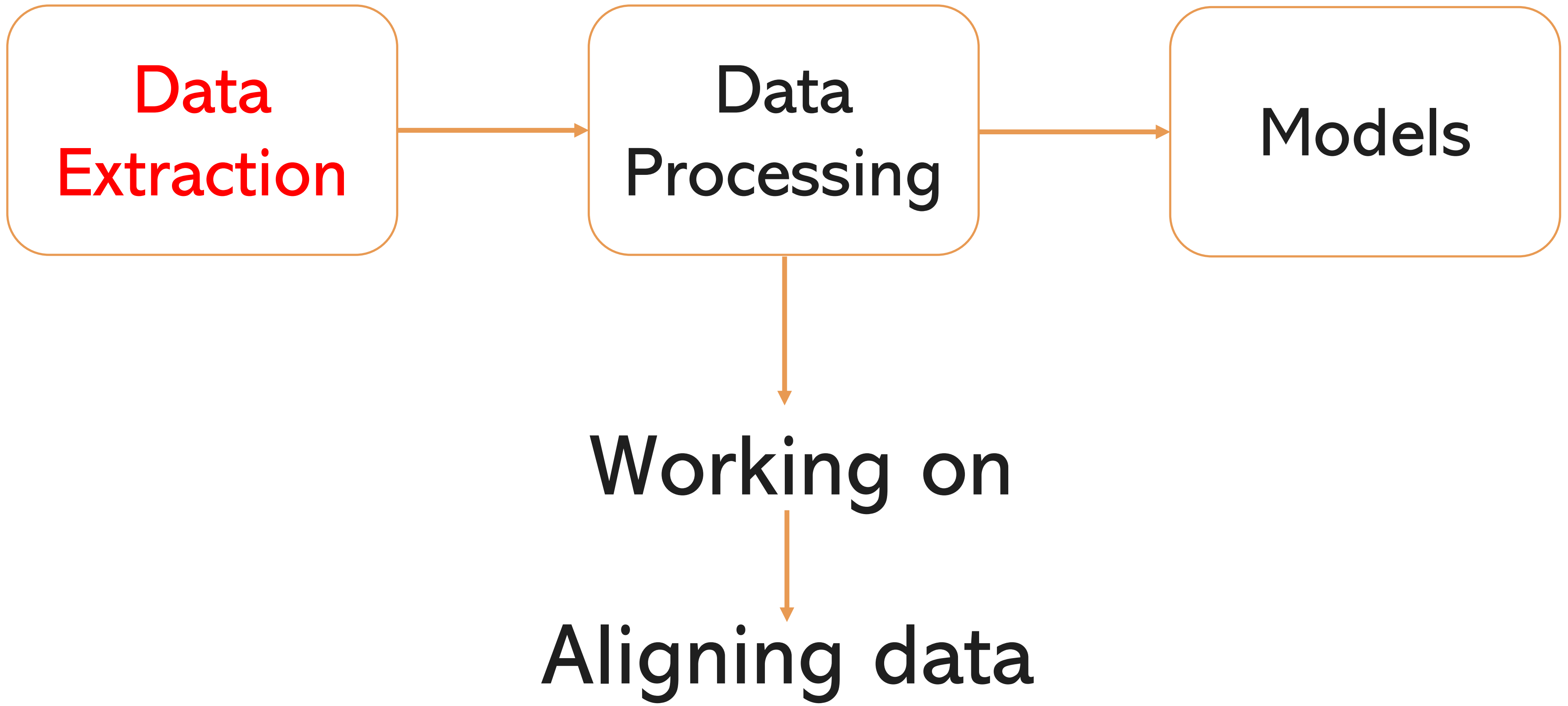
TA: Haoyin Xu

Group:Zhenyu Xiao
Haobin Zhou
Yimeng Xu
Emma Cardenas

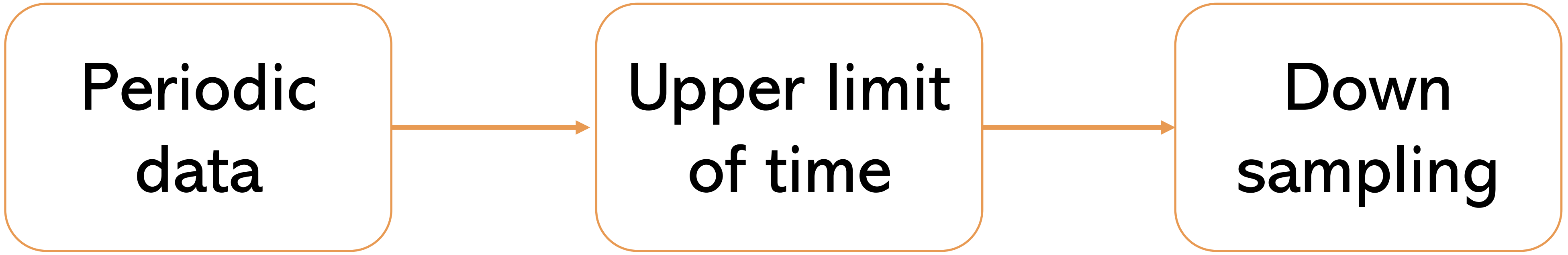
01
Aligning data

02

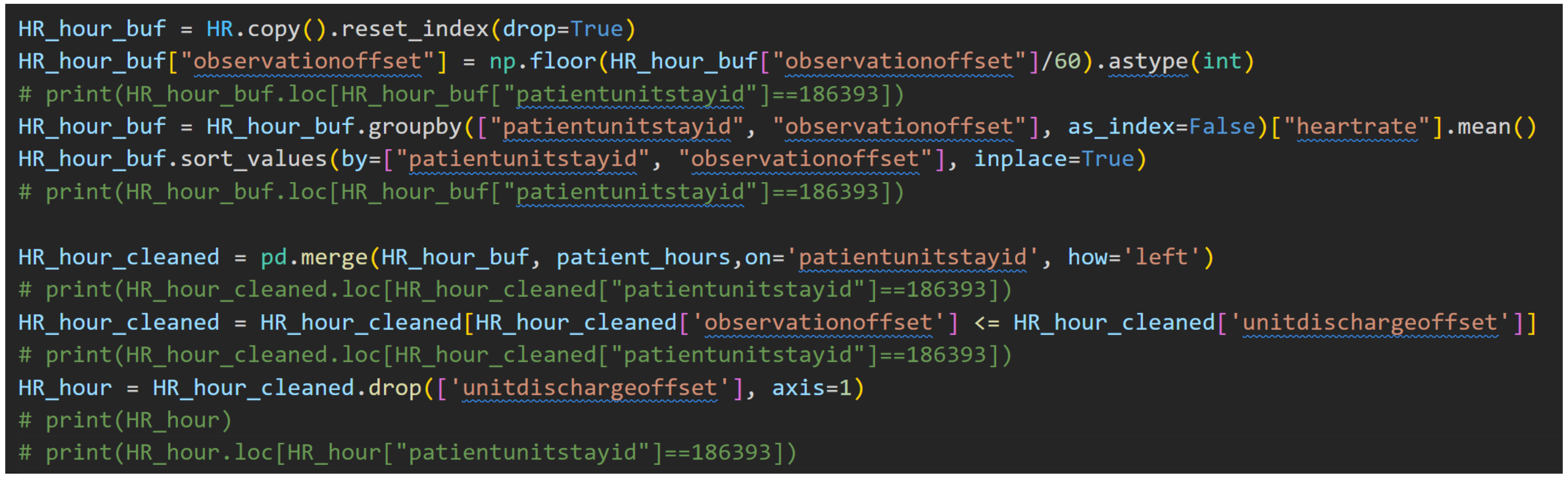
Our progress



02 ## Aligning data

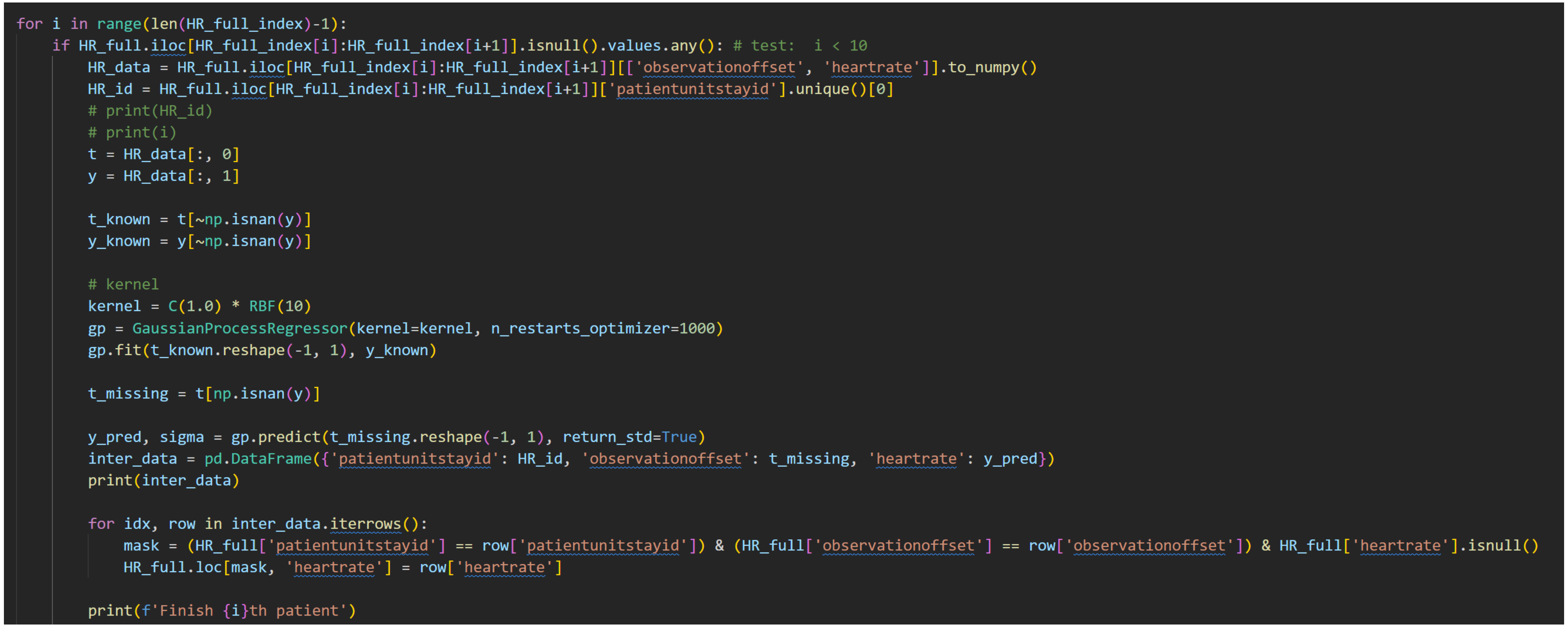


If need: One time per hour



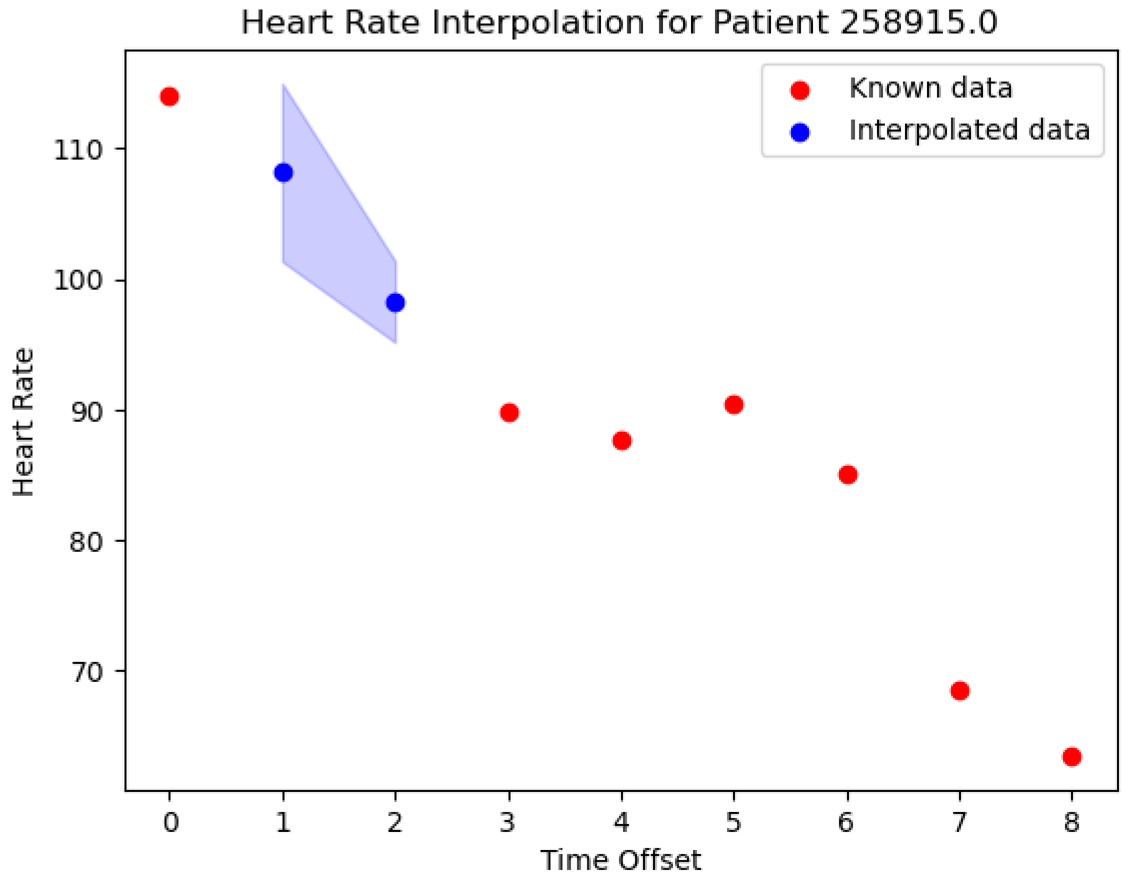
```
HR_hour_buf = HR.copy().reset_index(drop=True)
HR_hour_buf["observationoffset"] = np.floor(HR_hour_buf["observationoffset"]/60).astype(int)
# print(HR_hour_buf.loc[HR_hour_buf["patientunitstayid"]==186393])
HR_hour_buf = HR_hour_buf.groupby(["patientunitstayid", "observationoffset"], as_index=False)["heartrate"].mean()
HR_hour_buf.sort_values(by=["patientunitstayid", "observationoffset"], inplace=True)
# print(HR_hour_buf.loc[HR_hour_buf["patientunitstayid"]==186393])

HR_hour_cleaned = pd.merge(HR_hour_buf, patient_hours,on='patientunitstayid', how='left')
# print(HR_hour_cleaned.loc[HR_hour_cleaned["patientunitstayid"]==186393])
HR_hour_cleaned = HR_hour_cleaned[HR_hour_cleaned['observationoffset'] <= HR_hour_cleaned['unitdischargeoffset']]
# print(HR_hour_cleaned.loc[HR_hour_cleaned["patientunitstayid"]==186393])
HR_hour = HR_hour_cleaned.drop(['unitdischargeoffset'], axis=1)
# print(HR_hour)
# print(HR_hour.loc[HR_hour["patientunitstayid"]==186393])
```

02 Aligning data

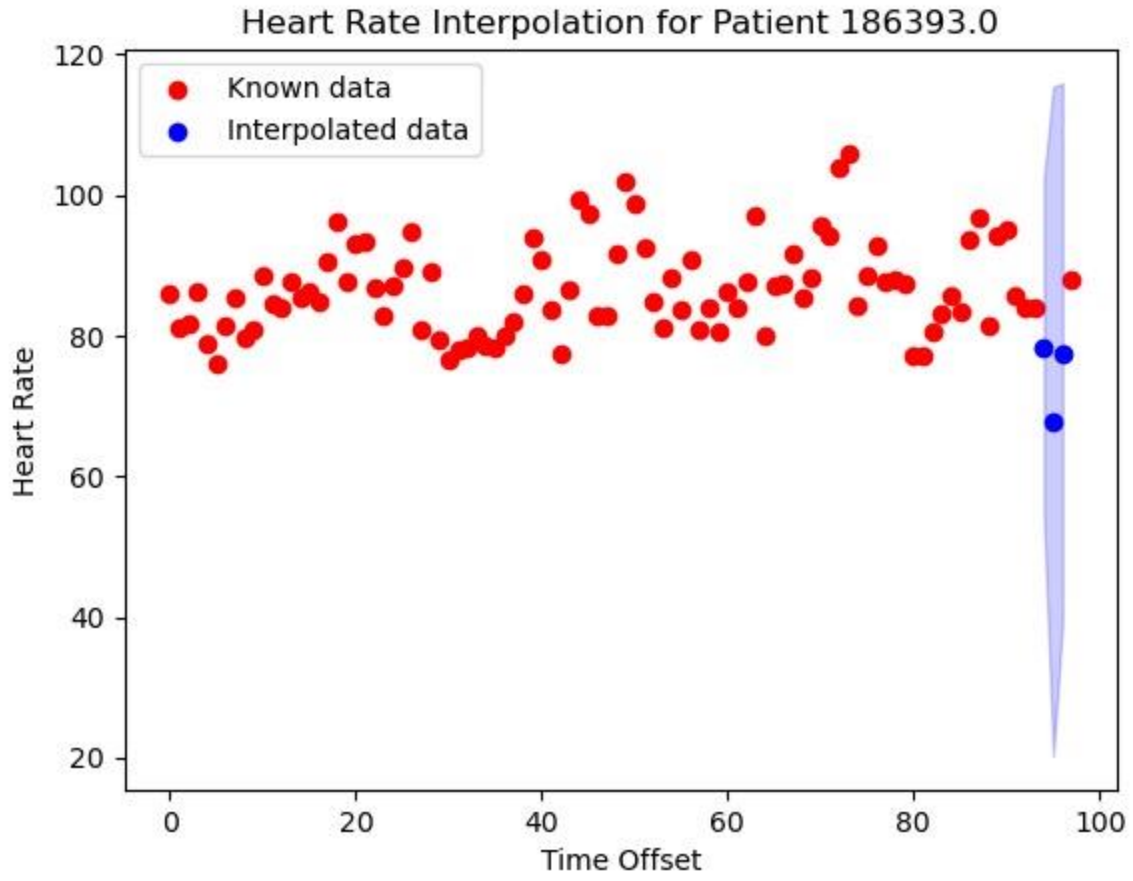


```python
for i in range(len(HR_full_index)-1):
    if HR_full.iloc[HR_full_index[i]:HR_full_index[i+1]].isnull().values.any(): # test:  i < 10
        HR_data = HR_full.iloc[HR_full_index[i]:HR_full_index[i+1]][['observationoffset', 'heartrate']].to_numpy()
        HR_id = HR_full.iloc[HR_full_index[i]:HR_full_index[i+1]]['patientunitstayid'].unique()[0]
        # print(HR_id)
        # print(i)
        t = HR_data[:, 0]
        y = HR_data[:, 1]

        t_known = t[~np.isnan(y)]
        y_known = y[~np.isnan(y)]

        # kernel
        kernel = C(1.0) * RBF(10)
        gp = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=1000)
        gp.fit(t_known.reshape(-1, 1), y_known)

        t_missing = t[np.isnan(y)]

        y_pred, sigma = gp.predict(t_missing.reshape(-1, 1), return_std=True)
        inter_data = pd.DataFrame({'patientunitstayid': HR_id, 'observationoffset': t_missing, 'heartrate': y_pred})
        print(inter_data)

        for idx, row in inter_data.iterrows():
            mask = (HR_full['patientunitstayid'] == row['patientunitstayid']) & (HR_full['observationoffset'] == row['observationoffset']) & HR_full['heartrate'].isnull()
            HR_full.loc[mask, 'heartrate'] = row['heartrate']

        print(f'Finish {i}th patient')
```

02 **Aligning data**

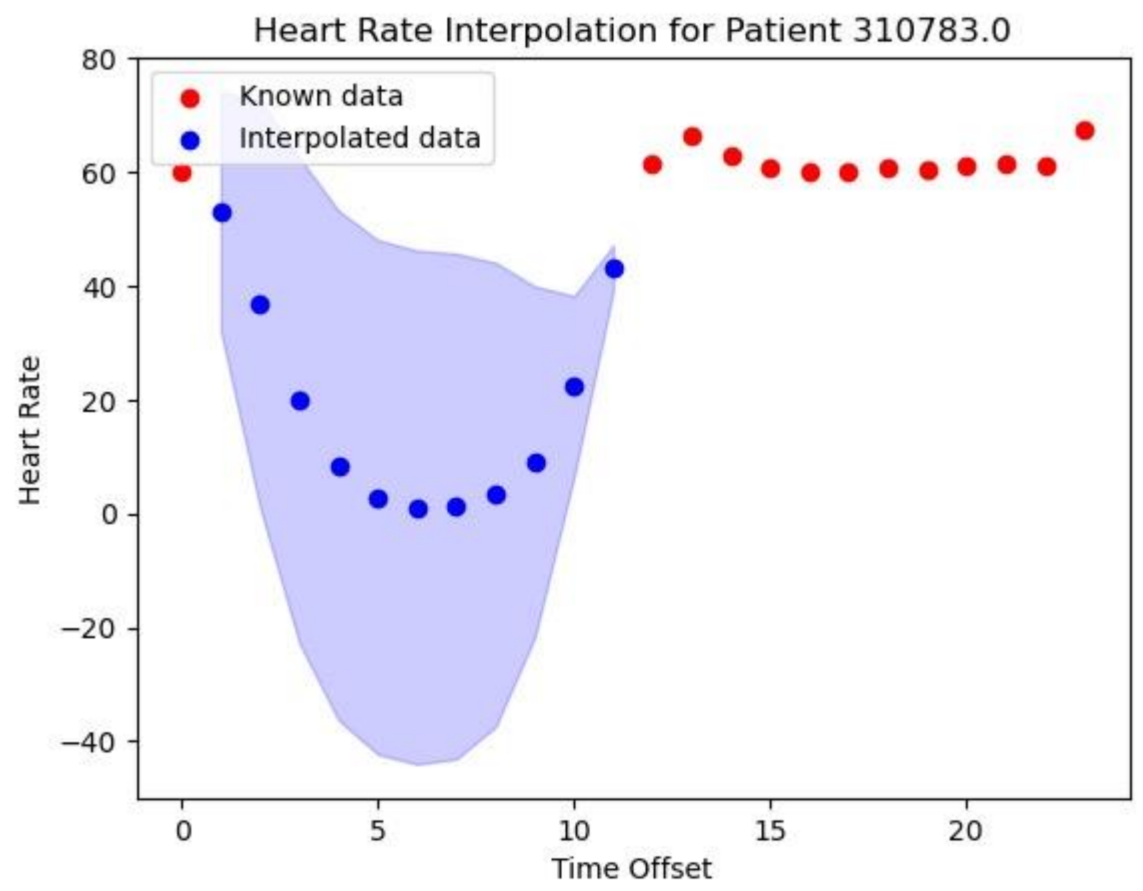


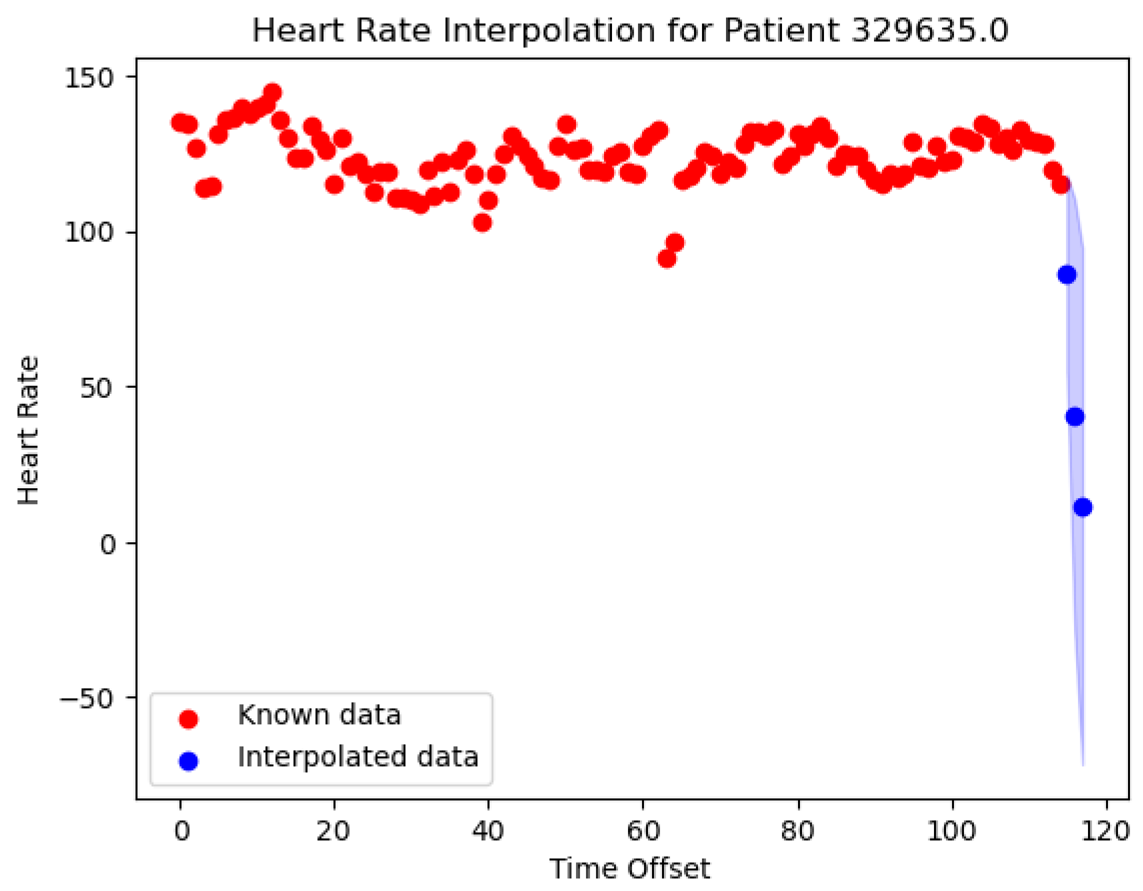
Reasonable



Large Confidence interval

02 Aligning data

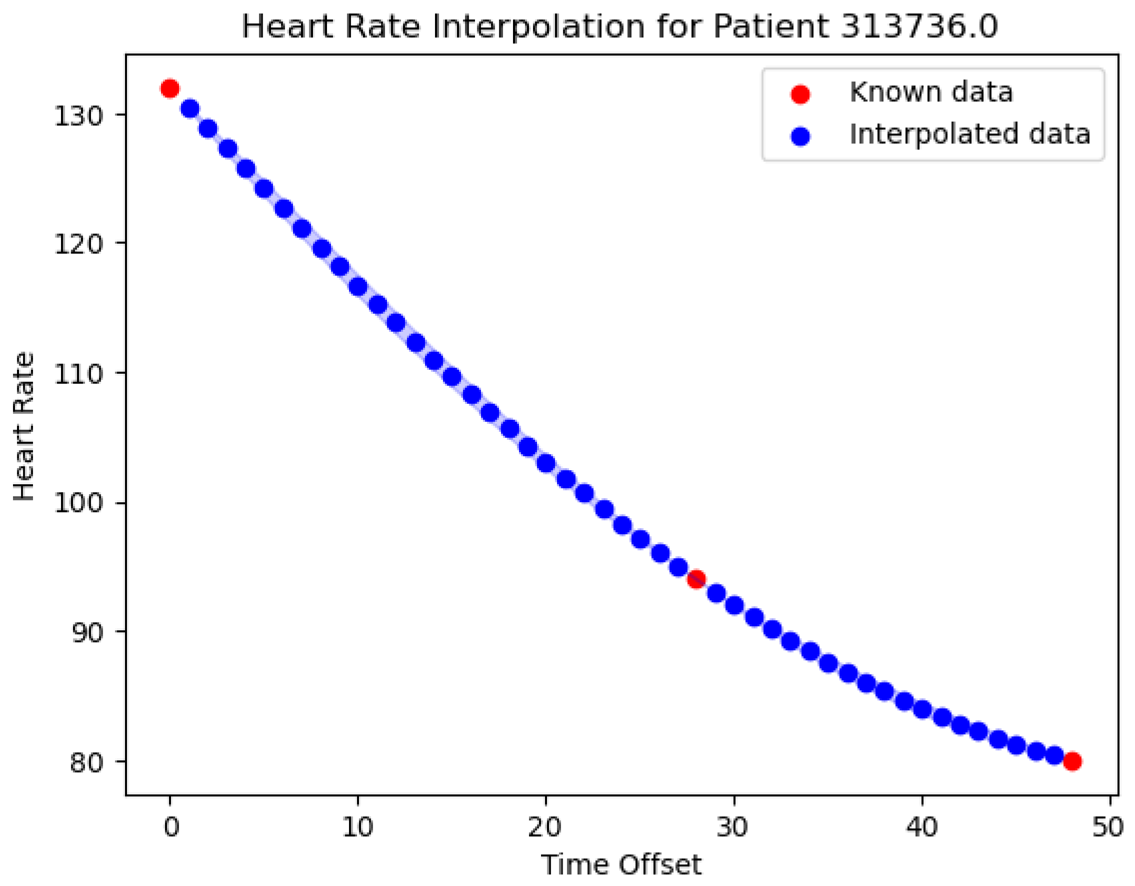


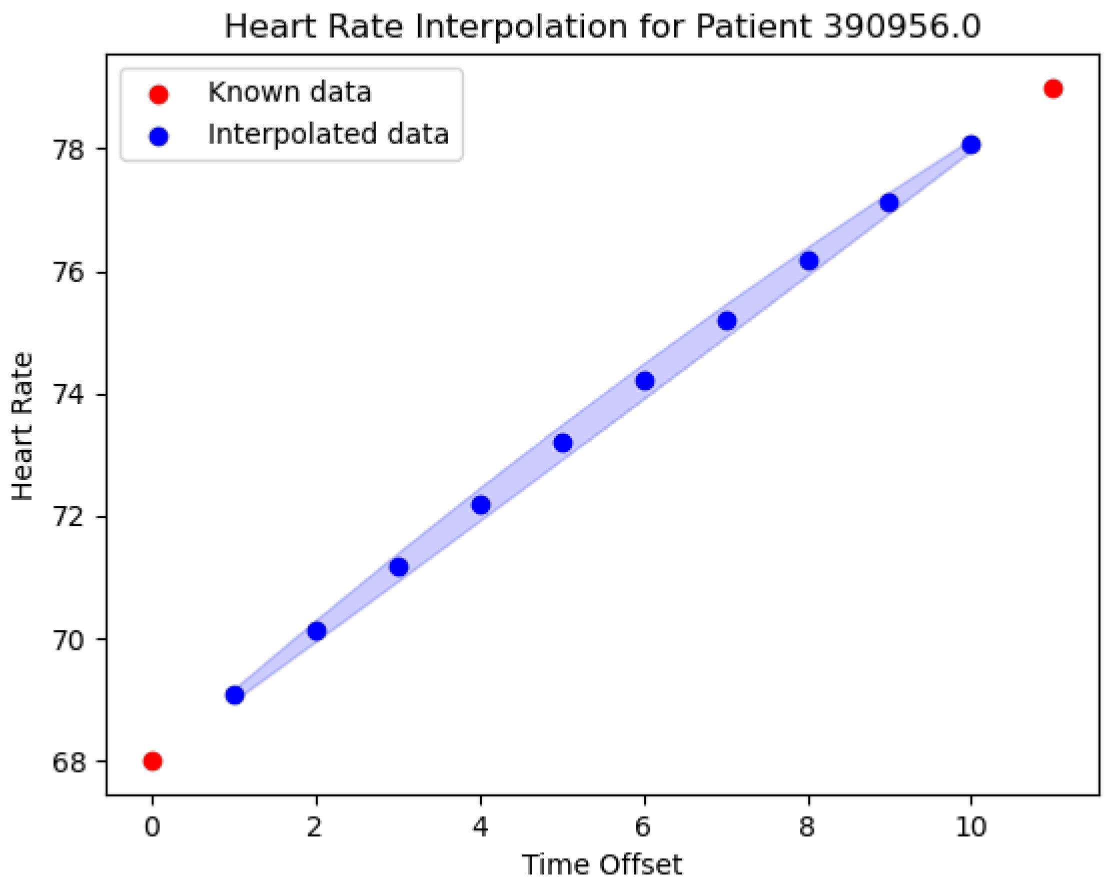


How to evaluate the results of different kernels?

02

Aligning data





Too sparse -- Drop

Thank you