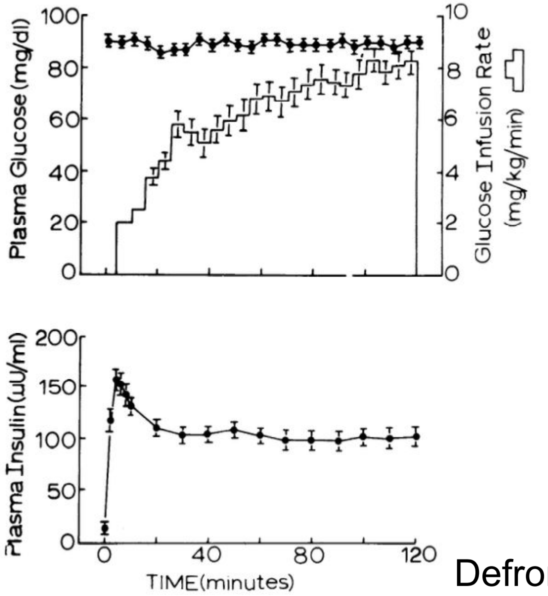
Algorithm for the Infusion Rate of Glucose During an Insulin Clamp

Jason Blohm, Nicholas Diehl, Joe Jeffrey, Sheng-Yau Lim
GlucoReg

Background: Hyperinsulinemic clamp

- Plasma insulin rate is raised and held at 120 μU/mL via constant insulin infusion
- Glucose is infused and blood glucose levels are measured; GIR adjusted
- Measures insulin sensitivity in patient
- Patients: Pre-diabetics, diabetics, those with endocrine & metabolic disorders



Defronzo et al., 1979

Refresher: Problem Statement

- In hyperinsulinemic clamp studies, Dr. Luther adjusts GIR on the fly based on his clinical judgment
- This can lead to inaccurate adjustments which can affect subject safety and data validity
- Some people claim that an algorithmic approach works, but no one Dr. Luther has talked to has been successful
- We will develop an algorithm that allows researchers to perform these studies in a more controlled manner

Refresher: Needs Assessment → Provider

Interface

1. Should be simple to understand
2. Should include inputs for all possible variables the physician may want to change: target glucose level, insulin clamp level, demographic data, time of experiment
3. Given patient demographics and history, should simulate the glucose level over time, prior to clinical testing

Refresher: Needs Assessment → Provider

Algorithm

1. Should calculate the amount of glucose uptake based on the constant insulin infusion rate the physician specifies
2. Depends on actual glucose infusion rate (not suggested)
3. Should output a recommended glucose infusion rate that accounts for the time delay in measuring glucose level from blood sample (t-1)

Refresher: Needs Assessment → Provider

Timing

1. Runtime -- should provide physician with proper glucose infusion rate (GIR) within 10 seconds of inputting the current glucose level
2. Should include an easy to navigate UI for immediate data entry

Refresher: Needs Assessment → Patient

Safety

1. Ensure that glucose levels do not exceed or drop below safe levels, as determined by the physician
2. Measurements need to be taken every 5 minutes to ensure glucose levels are where they should be. If not, the program should alert the physician (future iteration)
3. Must run smoothly so that no bugs interrupt the program

Refresher: Needs Assessment → System

Applicability and cost

1. Should be applicable to different physicians and different hospitals performing the same studies
2. Should be open source
3. Results from these studies should lower healthcare costs in the future

Progress: Multiple Regression Analysis

Linear regression model:
    GIR ~ 1 + Height + Weight + BSA + Age + Gender + Race

Estimated Coefficients:

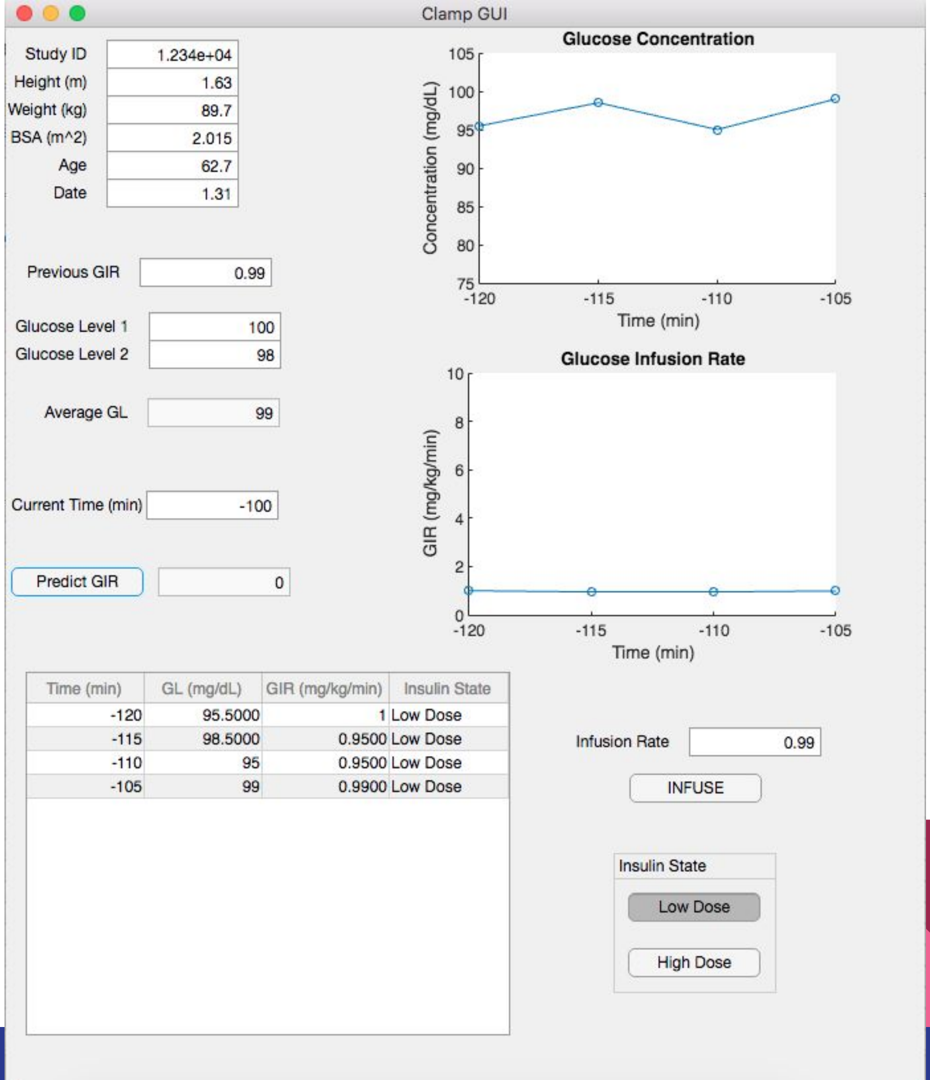
	Estimate	SE	tStat	pValue
(Intercept)	3.2236	2.011	1.603	0.11325
Height	15.35	5.7597	2.6651	0.0094679
Weight	0.19485	0.083456	2.3348	0.022309
BSA	-20.895	8.0997	-2.5797	0.0119
Age	-0.014226	0.0054871	-2.5927	0.011498
Gender	0.14254	0.16286	0.87521	0.38433
Race	-0.32696	0.22619	-1.4455	0.15261

Progress: First Iteration

- Relying on current algorithm for first iteration of algorithm
  - GIR = ((Gd-Gi)(10)((0.19)(body weight))/((Ginf)(15)) + (((SMi-2)(Gd)(FMi-1))/Gi)
    - Gd = desired plasma glucose concentration (mg/dL), 95
    - Gi = actual plasma glucose concentration (mg/dL)
    - Body weight in kg
    - Ginf = glucose concentration in infusate (mg/mL)
    - SMi = metabolic component: SMi = (SMi-2)(FMi)(FMi-1)
    - FMi = Gd/Gi
  - Will use our linear regression model to set the initial GIR

Progress: GUI

- Will take inputs and generate a predicted GIR
- Once closed, file will save to MATLAB workspace and an excel file



Code: GUI

```
time = cell2mat(app.UITable.Data(:,1));
gl = cell2mat(app.UITable.Data(:,2));
gir = cell2mat(app.UITable.Data(:,3));
%               x1 = linspace(-120,120,1000);

% Plot the GIR and GL

plot(app.glucoseinfus, time, gir,'-o');
plot(app.glucoseconc, time, gl,'-o');


    app.glucoseinfus.YLim = [0 10];
    app.glucoseconc.YLim = [75 105];

% Update the time
app.CurrentTime.Value = curtime + 5;

% Update the Infusion Rate
app.InfusionRate.Value = prevgir;
```

```
% Button pushed function: Button
function ButtonButtonPushed(app, event)

    % Update number of button pushes
    app.buttonpush.Value = app.buttonpush.Value +1;
     butpush = app.buttonpush.Value;

     % initialize variables
   studyid = app.StudyID.Value;
    height = app.Height.Value;
    bsa = app.BSA.Value;
    weight = app.Weight.Value;
    age = app.Age.Value;
    prevgir = app.PrevGIR.Value;
    gl_one = app.GlucoseLevel1.Value;
    gl_two = app.GlucoseLevel2.Value;
    avg_gl = mean([app.GlucoseLevel1.Value, app.GlucoseLevel2.Value]);
    app.AverageGL.Value = avg_gl;
    curtime = app.CurrentTime.Value;
    LD = app.LowDose.Value;


    % Calculate the future GIR
    if prevgir==0 && curgl==0
        app.PredictGIR.Value = 1+ 15.35*height + .19485*weight - 20.895*bsa - age*.014226;
    else
        % This is where we will put in the algorithm
        app.PredictGIR.Value = 0;
    end
    % Update table
    app.UITable.Data{butpush,1} = curtime;
    app.UITable.Data{butpush,2} = avg_gl;
    app.UITable.Data{butpush,3} = prevgir;
    if LD == 1
        app.UITable.Data{butpush,4} = 'Low Dose';
    else
        app.UITable.Data{butpush,4} = 'High Dose';
    end
```

Progress: Machine Learning

Met with Dr. Yevgeniy Vorobeychik

Two methods:

1. Inverse Reinforcement Problem
    a. Not Feasible to do in a semester
    b. Involves complex mathematics
2. **Support Vector Regression to build algorithm using scikit in Python**
    a. Open Source
    b. One of the best coding languages for Machine Learning

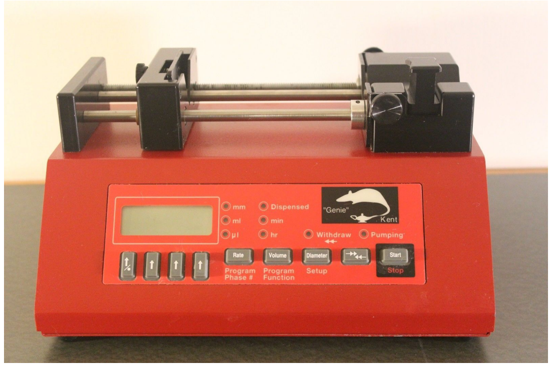
Our pump

CareFusion BD Alaris Pump Module with Guardrails MX software suite

*Because of proprietary and safety concerns, we do not plan on trying to interface with this pump

Pump POC

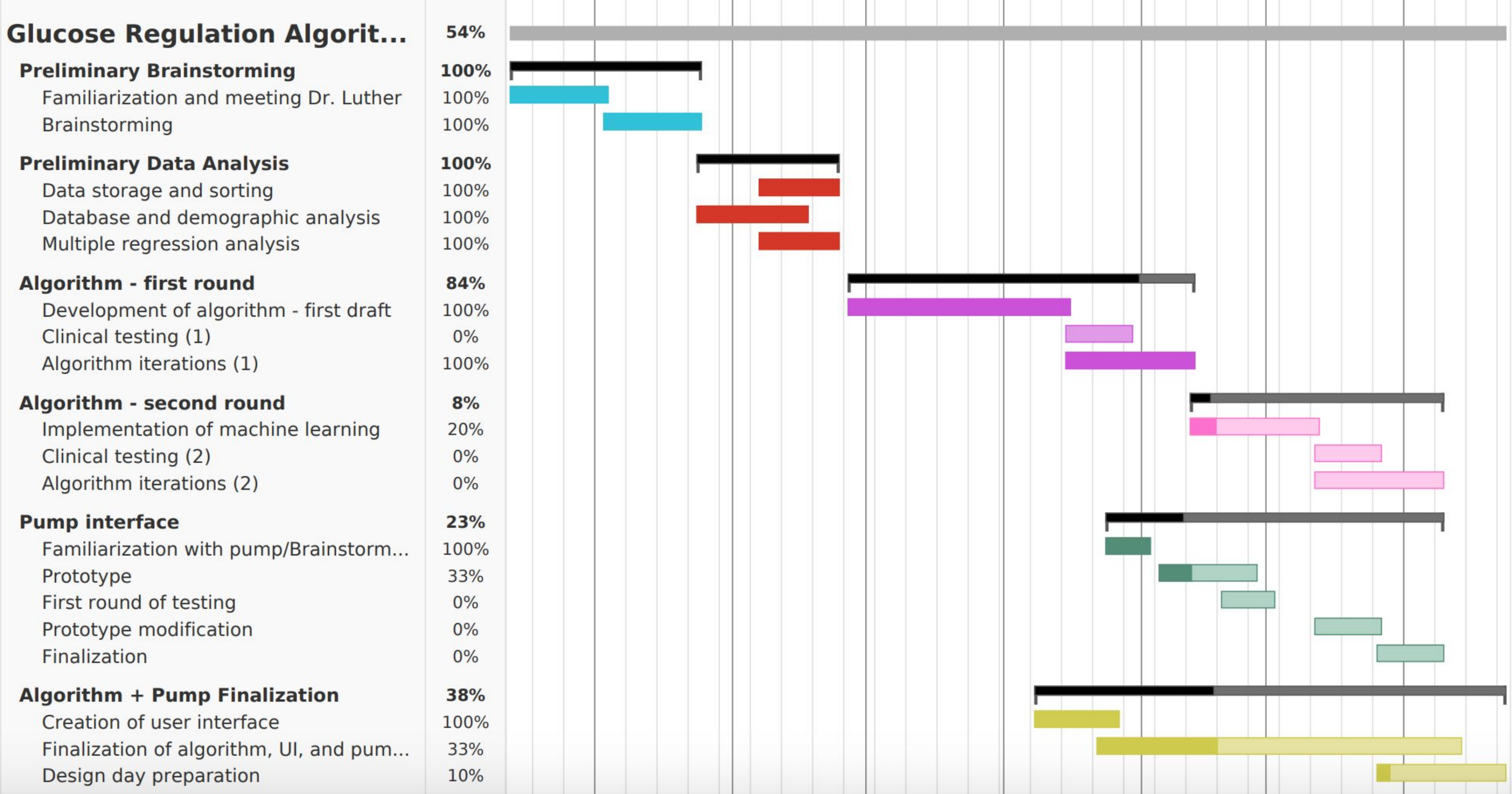
- Syringe Pump POC
  - Kent Scientific or Harvard Syringe Pump
  - Glucose solution in syringe
  - Communicate from MATLAB to pump via serial communication
  - Monitor physiological signals
  - Monitor pump controls



Pump POC Applicability

- Dr. Luther is on board with our POC idea and has mentioned another lab that does hyperinsulinemic clamp studies in mice
    - In future could apply the algorithm to mice studies and utilize the pump interface
    - Also possible to have a network of pumps for multiple doses, GIRs, etc.
- Could look into different languages for serial communication as well
    - Dr. Diedrich uses C++

Gantt Chart



9/17
10/17
11/17
12/17
1/18
2/18
3/18
4/18
11 17 24 1 8 15 22 29 5 12 19 26 3 10 17 24 31 7 14 21 28 4 11 18 25 4 11 18 25 1 8 15 22
Glucose Regulation Algorit... 54%
Preliminary Brainstorming 100%
Familiarization and meeting Dr. Luther 100%
Brainstorming 100%
Preliminary Data Analysis 100%
Data storage and sorting 100%
Database and demographic analysis 100%
Multiple regression analysis 100%
Algorithm - first round 84%
Development of algorithm - first draft 100%
Clinical testing (1) 0%
Algorithm iterations (1) 100%
Algorithm - second round 8%
Implementation of machine learning 20%
Clinical testing (2) 0%
Algorithm iterations (2) 0%
Pump interface 23%
Familiarization with pump/Brainstorm... 100%
Prototype 33%
First round of testing 0%
Prototype modification 0%
Finalization 0%
Algorithm + Pump Finalization 38%
Creation of user interface 100%
Finalization of algorithm, UI, and pum... 33%
Design day preparation 10%

BMES Grant Application

- Original BMES proposal had no mention of pump interface, so we are adding that in
- Beginning to modify proposal based on current progress with algorithm
- Will need a letter of support

Next Steps

- Finalize first iteration of algorithm and use in clinical trial (not directly on patient, but side by side)
- Obtain new sources of data
- Implement machine learning
- Begin working on pump POC
- Continue to work on BMES grant application
- Convert MATLAB code to Python

Potential Barriers

- Lack of knowledge of Python and C++
- Machine learning algorithm we would like to use is incredibly complex
- Lack of clinical opportunities

Questions?