Manage and Analyze Wearable Data for Your Research Using CyVerse

Shravan Aras, Ph.D., University of Arizona
Center for Biomedical Informatics & Biostatistics
Today’s webinar outline (Part 1)

- Current sensor landscape and its importance now
- SensorFabric and why we need it
- Integrating wearable device data into the study: CyVerse + MyDataHelps
- A step-by-step demonstration of how to do it
- Future steps
Focus on commercial wearables

Industrial Sensors

Clinical Sensors

Wearable Devices

IoT Sensors

Passive Sensors
Research Importance
30% increase in wearable use in last 5 years

• Wide range of affordability: $100 - $400
• More reliable and fashionable
• Increased ease of operations

Number of users in millions using wearable fitness devices in U.S.A.

Source: Statista
More than 7,000 publications in last 3 years

• **34% increase** in PubMed publications that involve wearable sensors

Publications indexed for “wearable sensors” on pubmed
More than 7,000 publications in last 3 years

• 34% increase in PubMed publications that use wearable sensors.

• **Over $58M NIH proposals** with wearable sensors funded in last 3 years

Publications indexed for “wearable sensors” on PubMed
More than 7,000 publications in last 3 years

• 34% increase in PubMed publications that use wearable sensors.
• Over $58M NIH proposals funded in last 3 years with wearable sensors.
• More data points for Idiographic (person specific) and nomothetic (within group)

Publications indexed for “wearable sensors” on PubMed
More than 7,000 publications in last 3 years

- 34% increase in PubMed publications that use wearable sensors.
- Over $58M NIH proposals funded in last 3 years with wearable sensors.
- More data points for idiographic (person specific) and nomothetic (within group).
- Inferred vs measured (e.g., physical activity, stress, sleep)
What are we trying to solve?
There are several pain points to address

• Difficulty in integrating multiple sensors at the same time

• Hurdles to easily query and analyze sensor data

• Tie it to institute resources already familiar to researchers

• How to address the new NIH data management policy
What is the NIH data management policy?

F.A.I.R.
What is the NIH data management policy?

• **F** – Findable (Example reDATA from UArizona library)

• **A** – Accessible (Through CyVerse Health or CyVerse)

• **I** – Interoperable (Standard storage format)

• **R** – Reproducible (example Jupyter notebooks)
We identified a few common components to solve this

- An easy way to connect to multiple sensors
- A way to make your research reproducible
- Accessible way to store sensor data and make it queriable “Whose laptop has that file? None of that”
- Make is very easy to share
And picked components to solve it

• An easy way to connect to multiple sensors - MyDataHelps

• A way to make your research reproducible – CyVerse DE

• Accessible way to store sensor data and make it queriable “Whose laptop has that file? None of that” – CyVerse Datastore

• Make is very easy to share – CyVerse Teams + Sharing
SensorFabric was born to facilitate all this!
What is SensorFabric?

• A set of libraries, infrastructure tools, and ton of cool code written here at UArizona that holds everything together.
What is SensorFabric library again?

• A set of libraries, infrastructure tools, and ton of cool code written here at UArizona that holds everything together.

• Opens and manages connections to various sources - MDH, AWS, etc.

• Hides all the nasty authentication, query executions and aggregation details
What is SensorFabric library again?

- A set of libraries, infrastructure tools, and tons of cool code written here at UArizona that holds everything together.
- Opens and manages connections to various sources - MDH, AWS, etc.
- Hides all the nasty authentication, query executions and aggregation details.
- Get your output as a simple pandas frame
- Supports for offline caching of indexed query results locally
- A ton of support functions and methods specific to wearables
What is SensorFabric library again?

• A set of libraries, infrastructure tools, and ton of cool code written here at UArizona that holds everything together.
• Opens and manages connections to various sources - MDH, AWS, etc.
• Hides all the nasty authentication, query executions and aggregation details.
• Get your output as a simple pandas frame
• Supports for offline caching of indexed query results locally
• A ton of support functions and methods specific to wearables

• Install now locally using `pip install sensorfabric`
• Opensource - https://github.com/UArizonaCB2/sensorfabric-py
Where do I start?
# Choosing a wearable

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Fitbit ($)</th>
<th>Apple Watch ($$)</th>
<th>Oura ($$$)</th>
<th>Garmin ($$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise Sessions</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td>Intraday HR</td>
<td>~ 5 seconds</td>
<td>~ 5 minutes</td>
<td>~ 5 minutes</td>
<td>~ 15 seconds</td>
</tr>
<tr>
<td>Resting HR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Duration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Phases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HRV</td>
<td>Day / Night (single)</td>
<td>Day / Night (mul)</td>
<td>Night (5 mins)</td>
<td>Night</td>
</tr>
<tr>
<td>IBI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Raw Acceleration</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Minute Temperature</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>spo2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Activity, Exercise, Sleep

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Fitbit ($)</th>
<th>Apple Watch ($$)</th>
<th>Oura ($$$)</th>
<th>Garmin ($$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise Sessions</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td>Intraday HR</td>
<td>~ 5 seconds</td>
<td>~ 5 minutes</td>
<td>~ 5 minutes</td>
<td>~ 15 seconds</td>
</tr>
<tr>
<td>Resting HR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Duration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Phases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HRV</td>
<td>Day / Night (single)</td>
<td>Day / Night (mul)</td>
<td>Night (5 mins)</td>
<td>Night</td>
</tr>
<tr>
<td>IBI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Raw Acceleration</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Minute Temperature</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>spo2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Everyone supports the basics*
## Intraday Heart Rate (IHR)

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Fitbit ($)</th>
<th>Apple Watch ($$)</th>
<th>Oura ($$$)</th>
<th>Garmin ($$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise Sessions</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td>Intraday HR</td>
<td>~ 5 seconds</td>
<td>~ 5 minutes</td>
<td>~ 5 minutes</td>
<td>~ 15 seconds</td>
</tr>
<tr>
<td>Resting HR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Duration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Phases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HRV</td>
<td>Day / Night (single)</td>
<td>Day / Night (mul)</td>
<td>Night (5 mins)</td>
<td>Night</td>
</tr>
<tr>
<td>IBI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Raw Acceleration</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Minute Temperature</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>spo2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Some devices offer something special

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Fitbit ($)</th>
<th>Apple Watch ($$)</th>
<th>Oura ($$$)</th>
<th>Garmin ($$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise Sessions</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td>Intraday HR</td>
<td>~ 5 seconds</td>
<td>~ 5 minutes</td>
<td>~ 5 minutes</td>
<td>~ 15 seconds</td>
</tr>
<tr>
<td>Resting HR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Duration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sleep Phases</td>
<td>Yes</td>
<td>Special support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HRV</td>
<td>Day / Night (single)</td>
<td>Day / Night (mul)</td>
<td>Night (5 mins)</td>
<td>Night</td>
</tr>
<tr>
<td>IBI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Raw Acceleration</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Minute Temperature</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>spo2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
More than just aggregate data...
Fitbit can give more than just summaries

Researchers can request access to Fitbit intraday data to unlock a wide range of high frequency metrics.

<table>
<thead>
<tr>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 activities-heart</td>
</tr>
<tr>
<td>2 activities-distance</td>
</tr>
<tr>
<td>3 activities-floors</td>
</tr>
<tr>
<td>4 activities-steps</td>
</tr>
<tr>
<td>5 activities-elevation</td>
</tr>
<tr>
<td>6 br</td>
</tr>
<tr>
<td>7 activities-calories</td>
</tr>
<tr>
<td>8 spo2</td>
</tr>
<tr>
<td>9 hrv</td>
</tr>
</tbody>
</table>

Intraday day HR values for a participant
Trying to measure stress? Garmin can help.

• A mixture of HR, HRV and activity
• Stress levels are provided as 3-minute averages of the real-time stress scores
• Values range from 1-100

<table>
<thead>
<tr>
<th>Range</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Not enough data</td>
</tr>
<tr>
<td>-2</td>
<td>Too much motion</td>
</tr>
<tr>
<td>1-25</td>
<td>Rest / Not Stressed</td>
</tr>
<tr>
<td>26-50</td>
<td>Low Stress</td>
</tr>
<tr>
<td>51-75</td>
<td>Medium Stress</td>
</tr>
<tr>
<td>76-100</td>
<td>High Stress</td>
</tr>
</tbody>
</table>

Number of seconds spent in each stress category
Get in-depth motion data with SensorKit

- **Multiple high frequency metrics**
  - Acceleration
  - Rotation Rate
  - Light Sensor

- **Acceleration & Rotation**
  - About 100Hz sampling rate
  - Does affect battery – almost half

```json
{
  "timestamp": 140941.628781,
  "acceleration": {
    "x": -0.30615234375,
    "y": 0.634033203125,
    "z": 0.739990234375
  },
  "startDate": "2023-08-28T06:43:57-0700",
  "identifier": 0
}
```

Does impact battery life
Oura is your companion for temperature

- Per minute temperature data, day + night
- IBI (Inter-beat Interval Values)
  - Validity indicating confidence
  - Be careful of duplicates!
What about BP, Glucose, Weight, etc.?

• Most BLE / BT smart devices connect Google Fit & Apple Health Kit
• Connecting them can be finicky → right permissions
• Syncing can be a challenge
• Good study material, instructions and monitoring can help!

<table>
<thead>
<tr>
<th>type</th>
<th>startdate</th>
<th>date</th>
<th>value</th>
<th>units</th>
<th>sourceidentifier</th>
<th>sourcename</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:28:25.000</td>
<td>2023-03-12 18:28:25.000</td>
<td>132</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:33:25.000</td>
<td>2023-03-12 18:33:25.000</td>
<td>130</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:38:25.000</td>
<td>2023-03-12 18:38:25.000</td>
<td>125</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:43:25.000</td>
<td>2023-03-12 18:43:25.000</td>
<td>112</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:48:25.000</td>
<td>2023-03-12 18:48:25.000</td>
<td>105</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
<tr>
<td>BloodGlucose</td>
<td>2023-03-12 18:53:25.000</td>
<td>2023-03-12 18:53:25.000</td>
<td>98</td>
<td>mg/dL</td>
<td>com.dexcom.G6</td>
<td>Dexcom G6</td>
<td>17375</td>
</tr>
</tbody>
</table>

Example data from Dexcom G6 CGM from one of our studies.
The Integration Magic
Big picture: CyVerse + MyDataHelps + AWS

CyVerse® → SensorFabric → AWS

SFTP

SensorFabric

GARMIN, fitbit, Apple Watch

MyDataHelps™ DESIGNER

Apple HealthKit, Google Fit

CGM Monitors, Smart Scales, Smart Scales

Oura Ring, 1080 Sprint

Rest API / Manual Ingestion

Direct Parquet
Focus: MyDataHelps

- CYVERSE®
  - AWS
    - Direct Parquet
    - Rest API / Manual Ingestion
  - SensorFabric
  - SFTP

- MyDataHelps™ DESIGNER
  - Apple HealthKit
  - Google Fit
  - GARMIN
  - fitbit
  - Apple Watch
  - CGM Monitors
  - Smart Scales
  - Smart Scales

- Oura Ring
- 1080 Sprint

- SensorFabric
- SFTP
Focus: How to connect with CyVerse

- SensorFabric
- SFTP
- MyDataHelps
- Direct Parquet
- Rest API / Manual Ingestion
- Apple HealthKit
- Google Fit
- CGM Monitors
- Smart Scales
- Smart Scales
- Oura Ring
- 1080 Sprint
- GARMIN
- fitbit
- Apple Watch
Focus: CyVerse + AWS but why?

CyVerse

AWS

SensorFabric

SFTP

Direct Parquet

Rest API / Manual Ingestion

Oura Ring

1080 Sprint

GARMIN

fitbit

Apple Watch

MyDataHelps™

Apple HealthKit

Google Fit

CGM Monitors

Smart Scales

SensorFabric

SFTP

Rest API / Manual Ingestion

Oura Ring

1080 Sprint
How we built our AWS stacks

Raw CSV

Sensor Data

Schema Inference / AWS GLUE

Apache Airflow / AWS Lambda

Add to S3 as Parquet

Real-Time

Real-time processing - Spark

Store in AWS S3

Add to S3 as Parquet

Athena for SQL

Athena for SQL

Athena for SQL

Athena for SQL

Python Library for SensorFabric, R Library for SQL, Power BI, Tableau ... connects to SQL
### Summarizing the Options

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| CyVerse -> MyDataHelps (using SensorFabric) | • No AWS setup needed  
• CyVerse Health will support PHI access  
• Least manual intervention | • Cannot create custom SQL tables / views  
• Can only access when study is active |
| CyVerse <- MyDataHelps (via sftp + duckdb) | • Data stored locally in CyVerse Data Store  
• Data persistent after study ends at MDH  
• Ingest data into duckdb  
• Easily share data | • Requires some amount of manual setup and coding  
• Sftp will not work with CyVerse Health  
• Will need schema generation for new sensors |
| CyVerse -> AWS (running Sensorfabric) | • Power of AWS Athena for large scale data  
• Can connect to sensor data not supported by MDH | • AWS setup and cost |
## Summarizing the Options

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyVerse -&gt; MyDataHelps (using SensorFabric)</td>
<td>• No AWS setup needed&lt;br&gt;• CyVerse Health will support PHI access&lt;br&gt;• Least manual intervention</td>
<td>• Cannot create custom SQL tables / views&lt;br&gt;• Can only access when study is active</td>
</tr>
<tr>
<td>CyVerse &lt;- MyDataHelps (via sftp + duckdb)</td>
<td>• Data stored locally in CyVerse Data Store&lt;br&gt;• Data persistent after study ends at MDH&lt;br&gt;• Ingest data into duckdb&lt;br&gt;• Easily share data</td>
<td>• Requires some amount of manual setup and coding&lt;br&gt;• Sftp will not work with CyVerse Health&lt;br&gt;• Will need schema generation for new sensors</td>
</tr>
<tr>
<td>CyVerse -&gt; AWS (running Sensorfabric)</td>
<td>• Power of AWS Athena for large scale data&lt;br&gt;• Can connect to sensor data not supported by MDH</td>
<td>• AWS setup and cost</td>
</tr>
</tbody>
</table>

**Today’s Focus**
Participant Journey
Download & Enroll from MyDataHelps App

- Enter 6 Letter Project Code
- OR
- Scan QR Code

Welcome

- You can share data from your Fitbit account if you have one. To begin, click or tap below to log in with your Fitbit credentials.

1. 2. 3. 4.

- Learn more about MDH at - https://mydatahelps.org/
Scan study QR code to enroll

• Learn more about MDH at - https://mydatahelps.org/
Join the study

• Learn more about MDH at - https://mydatahelps.org/
Hit the Connect Fitbit button

- Learn more about MDH at - https://mydatahelps.org/
Tada! You are all done

• Learn more about MDH at - https://mydatahelps.org/
Setting it up with MyDataHelps
Enable sensors in MDH Research Dashboard
You can setup sftp support for CyVerse
Choose what you want to export

Some exports can contain PHI. Only use when using CyVerse Health.
Generate your secrets in MDH service account

- Save your private key securely. It will be not shown again!
- You will need this for the CyVerse setup next.
Connecting it all with CyVerse
Setup the app in the Discovery Environment

- Log into your CyVerse account

- Search for the app named “sensorfabric-jupyter”

- Launch the app

- You can also build your own app using the tool “sensorfabric-jupyter"

Learn more at - https://learning.cyverse.org/
Various connection options

---

**Direct connect to MyDataHops.**
- Section 1 of 2

- MyDataHops account secret
- MyDataHops Account Name
- MyDataHops Project Name

**Direct connect to AWS.**
- Section 2 of 2

- Database
- Access Key
  - Enter the provided access key. Will be shared with you using VA Stache.
- Secret Key
  - Add the secret key provided. Will be shared with you using VA Stache.
- Region
  - us-east-1
### Connecting directly to MDH

#### Step 2: Analysis Parameters

<table>
<thead>
<tr>
<th>Section 1 of 2</th>
<th>Direct connect to MyDataHelps</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyDataHelps account secret</td>
<td></td>
</tr>
<tr>
<td>MyDataHelps Account Name</td>
<td></td>
</tr>
<tr>
<td>MyDataHelps Project Name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section 2 of 2</th>
<th>Direct connect to AWS</th>
</tr>
</thead>
</table>

54
Can also connect directly to AWS Athena
Configure your app to connect to MDH

• Enter the MDH Secret Key you got earlier from service account

---BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----
Configure your app to connect to MDH

• Enter the MDH Secret Key you got earlier from service account

• Shrink the key to a single line and add \n for newline

---BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----
Configure your app to connect to MDH

• Enter the MDH Secret Key you got earlier from service account
• Shrink the key to a single line and add \n for newline.
• Enter the MDH account name
Configure your app to connect to MDH

• Enter the MDH Secret Key you got earlier from service account
• Shrink the key to a single line and add \n for newline.
• Enter the MDH account name
• Enter the project name
Your “hello world” into sensors

Python & SQL is good to know.
Needle in the fabric

• A Needle() is how you initiate a data connection

```python
[2]: # Connect to MDH source.
    from sensorfabric.needle import Needle
    needle = Needle(method='mdh')
```

And you are connected to your source in MDH
Needle in the fabric

• A Needle() is how you initiate a data connection

```python
[2]: # Connect to MDH source.
    from sensorfabric.needle import Needle
    needle = Needle(method='mdh')

[3]: # Connect to AWS source.
    from sensorfabric.needle import Needle
    needle = Needle(method='aws')
```

And you are connected to your source in MDH

Just change the method to `aws` to connect to AWS
But what if I want to connect to both?

```
# Set the profileName parameter to separate out the authentication in the backend files.
needle_mdh = Needle(method='mdh', profileName='mdh-connect')
needle_aws = Needle(method='aws', profileName='aws-connect')
```
How do I exec SQL query?

- By call `execQuery(...)`
I like to create lambda functions

Less typing. Now we can just use q()

```python
[ ]: q = lambda x: needle.execQuery(x)
r1 = q('show tables')
r2 = q('describe garmindailysummary')
```
And what about offline caching?

- Even though querying is cheap, network transfer can be huge and take time
- Can be frustrating when rerunning to debug
- We have an option to create an query index offline and cache
And what about offline caching?

- Non-cached query takes about 9.5 seconds in total for 46k temperature data records
- Almost all of it is network + some delay in query queuing at Athena

```python
[6]:   needle = Needle(method='aws')

[17]:   %time
temp = needle.execute("\nselect from_unixtime(unixtimestamp) datetime, skintemp from temperature where pid = 1 and date(from_unixtime(unixtimestamp)) >= date('2021-08-06') and date(from_unixtime(unixtimestamp)) <= date('2021-09-06') and skintemp is not null order by from_unixtime(unixtimestamp) asc\n"")

CPU times: user 871 ms, sys: 23.8 ms, total: 895 ms
Wall time: 9.55 s

[20]:   print(temp.shape)
print((temp.memory_usage().sum()/1024)/1024)  # Memory consumed in MB

(46074, 2)
1.0545501708984375
```
With Caching enabled!

- Just set offlineCache=True and the library takes care of the rest!
Q: What’s next?   A: Part 2 of this webinar!
Thank you!

Contact me for any questions: shravanaras@arizona.edu

**Additional Links**

1. CareEvolution - [https://careevolution.com/](https://careevolution.com/)
2. Superset-Jupyter container GitHub - [https://github.com/nextgensh/sensorfabric-jupyterlab-cyverse.git](https://github.com/nextgensh/sensorfabric-jupyterlab-cyverse.git)
3. Sensorfabric GitHub - [https://github.com/UArizonaCB2/sensorfabric-py.git](https://github.com/UArizonaCB2/sensorfabric-py.git)