

---

# **loinc2hpo Documentation**

***Release 1.1.0***

**Peter Robinson, Aaron Zhang**

**Jun 08, 2020**



---

## Contents

---

<b>1</b>	<b>Introduction to LOINC</b>	<b>3</b>
1.1	Parts of LOINC entry . . . . .	3
1.2	Common LOINC codes . . . . .	4
<b>2</b>	<b>Introduction to FHIR</b>	<b>5</b>
2.1	Resources . . . . .	5
<b>3</b>	<b>Our Mission</b>	<b>11</b>
<b>4</b>	<b>Working logic</b>	<b>13</b>
4.1	Mapping LOINC to candidate HPO terms . . . . .	13
4.2	Retrieve Resources from EHR systems . . . . .	16
4.3	Convert Observations into HPO terms . . . . .	16
<b>5</b>	<b>Annotation Logic</b>	<b>17</b>
5.1	Qn . . . . .	17
5.2	Ord . . . . .	18
5.3	Nom . . . . .	18
5.4	Nar and other types . . . . .	19
<b>6</b>	<b>Install &amp; running</b>	<b>21</b>
<b>7</b>	<b>Configuration</b>	<b>23</b>
7.1	Mandatory Settings . . . . .	23
7.2	Optional Settings . . . . .	23
7.3	Change Settings . . . . .	24
<b>8</b>	<b>Tutorial</b>	<b>25</b>
8.1	Overview . . . . .	25
8.2	Annotation . . . . .	25
8.3	Converting Observations to HPO terms . . . . .	28
<b>9</b>	<b>Consumption Tutorial</b>	<b>33</b>
9.1	Overview . . . . .	33
9.2	Installation . . . . .	33
<b>10</b>	<b>Contact</b>	<b>37</b>



Loinc2hpo is a JavaFX app designed to convert FHIR observations into HPO terms. The app has three core functionalities:

- help mapping LOINC codes into candidate HPO terms
- help retrieve patient observations
- convert patient observations into HPO terms

Mapping LOINC to HPO terms requires significant efforts from biocurators. Loinc2hpo tries to simplify the process by automatically finding the best matching terms using Sparql query. With this app, collaborators can easily split LOINC codes of interest into smaller tasks and share their annotations.



# CHAPTER 1

## Introduction to LOINC

LOINC stands for Logical Observation Identifiers Names and Codes (LOINC). According to Regensrief, who developed and maintains LOINC:

“LOINC provides a set of universal names and ID codes for identifying laboratory and clinical test results. LOINC facilitates the exchange and pooling of results, such as blood hemoglobin, serum potassium, or vital signs, for clinical care, outcomes management, and research.” .

The LOINC website provides a very detailed explanation of LOINC. Here, we try to briefly illustrate the essence of LOINC and focus on the aspects that are relevant to this app.

In essence, LOINC is simply a big table with ~86,000 entries that define the names and IDs of laboratory tests. The following shows three examples of LOINC codes:

LOINC	Name	Component	Property	Time	Aspect	System	Scale	Method
10450-5	Glucose [Mass/volume] in Serum or Plasma --10 hours fasting	Glucose*post 10H CFst	MCnc	Pt		Ser/Plas	Qn	
777-3	Platelets [# /volume] in Blood by Automated count	Platelets	NCnc	Pt		Bld	Qn	Automated count
5769-5	Bacteria [# /area] in Urine sediment by Microscopy high power field	Bacteria	Naric	Pt		Urine sed	Qn	Microscopy.light.HPF

You can see that each LOINC entry simply represents a laboratory test. The first column LOINC contains a value with at most 7 numbers, separated by a -. This number is unique so that using it can uniquely identify an laboratory test. For example, 10450-5 represents a glucose test called “Glucose [Mass/volume] in Serum or Plasma”, while 777-3` represents a test on the count of platelets “Plates [# /volume] in Blood by Automated count”. Each LOINC entry have several other fields that define the test from different aspects.

### 1.1 Parts of LOINC entry

Component: defines the analyte in the test

Property: defines “kinds of quantities”. It can be divided into five several categories, mass, substance, catalytic activity, and number or counts. Each category is further divided into subclasses, for example MCnc or “mass concentration” is a subclass of “mass” property, while NCnc or “number of concentration (count/vol)” and Naric or “number aeric (number per area)” are subclasses of counts.

**Time:** defines whether a measurement was made at a moment, or aggregated from a series of physiologic states. The three examples are all **PT** (“points”), meaning that they are measurements at a single time point. As an example, a test on “daily urine amount” will be labeled as **24H** (“24 hours”).

**Aspect:** defines a modifier for a measurement over a duration. For example, “8H^max heart rate” means the “max” heart rate measured during an 8-hour period. Min, max, first, last, mean typically appear here.

**System:** can be considered as the specimen for the test, such as “serum”, “blood”, “urine”, “cerebrospinal fluid” etc.

**Scale** defines the scale of the measurement. Scale is the most important information for our application. The following table summarizes possible values of **scale**.

Table 12: Type of Scale (ref1)

Scale Type	Abbr.	Description
Quantitative	Qn	The result of the test is a numeric value that relates to a continuous numeric scale. Reported either as an integer, a ratio, a real number, or a range. The test result value may optionally contain a relational operator from the set {<=, <, >, >=}. Valid values for a quantitative test are of the form “7”, “-7”, “7.4”, “-7.4”, “7.8912”, “0.125”, “<10”, “<10.15”, “>12000”, 1-10, 1:256
Ordinal	Ord	Ordered categorical responses, e.g., 1+, 2+, 3+; positive, negative; reactive, indeterminate, nonreactive. (Previously named SQ)
Quantitative or Ordinal	OrdQn	Test can be reported as either Ord or Qn, e.g., an antimicrobial susceptibility that can be reported as resistant, intermediate, susceptible or as the mm diameter of the inhibition zone. (Previously named SQN) We discourage the use of OrdQn in other circumstances.
Nominal	Nom	Nominal or categorical responses that do not have a natural ordering. (e.g., names of bacteria, reported as answers, categories of appearance that do not have a natural ordering, such as, yellow, clear, bloody. (Previously named QL)
Narrative	Nar	Text narrative, such as the description of a microscopic part of a surgical papule test.
“Multi”	Multi	Many separate results structured as one text “glob”, and reported as one observation, with or without imbedded display formatting.
Document	Doc	A document that could be in many formats (XML, narrative, etc.)
Set	Set	Used for clinical attachments

ref1: LOINC USERS’ GUIDE, P32

**Qn**, **Ord** and **Nom** are the three most frequently used LOINC in real world, accounting for probably 99% cases, particularly **Qn**, which may account for 80% cases alone. **Qn** values may be continuous (e.g. serum sodium concentration) or discrete (e.g. titers, 1:16, 1:32). The most frequent **Ord** type are “yes/no” tests (e.g. presence or absence of a substance in the blood).

Aside from the three main types, **Nar** are reported as free texts.

**Method:** defines the method used for the measurement.

## 1.2 Common LOINC codes

The entire LOINC table has ~85,000 entries. Regenstrief also provides a subset of it containing commonly used top 2000 LOINC. According to Regenstrief, those top 2000 LOINC represent about 98% of all tests in real world applications (ref).



## Introduction to FHIR

Fast Healthcare Interoperability Resources (FHIR) is a set of standards created by HL7 that aims to improve the exchange and interoperability of healthcare information. Consider the following case:

The patient visits clinics, hospitals and receive medical tests in clinical labs, or visits drug stores. It would be nice if the physicians can retrieve his test results from other providers, or the drug store can automatically receive the prescriptions for the patient. The above is not possible without a standardization of information exchange. This is how FHIR come into play.

FHIR is also extremely helpful in reusing Electronic Health Record for scientific research. EHRs provide a wealth of medical information, but their lack of standardization presents a huge hurdle for their reusability. With FHIR, one can easily process patient record stored in different EHR systems.

### 2.1 Resources

FHIR uses RESTful API technology for data exchange. It takes all medical information as individual pieces of resources. For example, Administration-related resources are mainly about identifications (e.g. about patient, about practioner, about device ...). Diagnostics resources are those about observations, specimen etc. There are also resources about medications and health insurances. Resources are typically stored and exchanged in JSon format. There are unique identifiers in each resource that can cross reference to other another so that it is possible to find, say, all resources related to one patient.





Claim, Account, Coverage, Claim, EligibilityRequest, ExplanationOfBenefit, etc.

## Observation

Among all the resources, Observation is the most central one in EHR. Observation describes a medical test result for one patient. The following JSON snippet is an excerpt from a real world (ref: [hl7.org](http://hl7.org)).

Information about the ID of the resource.

```
{
  "resourceType": "Observation",
  "id": "f001",
  "text": {
    "status": "generated",
    "div": "<div xmlns='http://www.w3.org/1999/xhtml'><p><b>Generated Narrative_
    ↳with Details</b></p><p><b>id</b>: f001</p><p><b>identifier</b>: 6323 (OFFICIAL)
    ↳</p><p><b>status</b>: final</p><p><b>code</b>: Glucose [Moles\\volume] in Blood
    ↳<span>(Details : {LOINC code '15074-8' = 'Glucose [Moles\\volume] in Blood', given
    ↳as 'Glucose [Moles\\volume] in Blood'})</span></p><p><b>subject</b>: <a>P. van
    ↳de Heuvel</a></p><p><b>effective</b>: 02\\04\\2013 9:30:10 AM --&gt; (ongoing)</
    ↳p><p><b>issued</b>: 03\\04\\2013 3:30:10 PM</p><p><b>performer</b>: <a>A.
    ↳Langeveld</a></p><p><b>value</b>: 6.3 mmol\\l<span> (Details: UCUM code mmol\\L
    ↳= 'mmol\\L')</span></p><p><b>interpretation</b>: High <span>(Details : {http://
    ↳hl7.org/fhir/v2/0078 code 'H' = 'High', given as 'High'})</span></p><h3>
    ↳ReferenceRanges</h3><table><tr><td>-</td><td><b>Low</b></td><td><b>High</b></td></
    ↳tr><tr><td>*</td><td>3.1 mmol\\l<span> (Details: UCUM code mmol\\L = 'mmol\\L
    ↳')</span></td><td>6.2 mmol\\l<span> (Details: UCUM code mmol\\L = 'mmol\\L')</
    ↳span></td></tr></table></div>"
  },
  "identifier": [
    {
      "use": "official",
      "system": "http://www.bmc.nl/zorgportal/identifiers/observations",
      "value": "6323"
    }
  ],
  "status": "final",
```

Information about the test, identified with a LOINC code.

```
"code": {
  "coding": [
    {
      "system": "http://loinc.org",
      "code": "15074-8",
      "display": "Glucose [Moles\\volume] in Blood"
    }
  ]
},
```

Information about the subject/patient of this resource.

```
"subject": {
  "reference": "Patient/f001",
  "display": "P. van de Heuvel"
},
```

Some meta data about this test:

```
"effectivePeriod": {
  "start": "2013-04-02T09:30:10+01:00"
},
"issued": "2013-04-03T15:30:10+01:00",
"performer": [
  {
    "reference": "Practitioner/f005",
    "display": "A. Langeveld"
  }
],
```

Measured value and reference range:

```
"valueQuantity": {
  "value": 6.3,
  "unit": "mmol/l",
  "system": "http://unitsofmeasure.org",
  "code": "mmol/L"
},
"referenceRange": [
  {
    "low": {
      "value": 3.1,
      "unit": "mmol/l",
      "system": "http://unitsofmeasure.org",
      "code": "mmol/L"
    },
    "high": {
      "value": 6.2,
      "unit": "mmol/l",
      "system": "http://unitsofmeasure.org",
      "code": "mmol/L"
    }
  }
]
```

Interpretation from physicians:

```
"interpretation": {
  "coding": [
    {
      "system": "http://hl7.org/fhir/v2/0078",
      "code": "H",
      "display": "High"
    }
  ]
},
}
```

Patient

Patient manages all relevant information about the patient, such as name, address, sex, etc. The following snippet is an example in JSon (ref: [hl7.org](http://hl7.org)). The code is pretty self-explanatory.

```
{
  "resourceType": "Patient",
  "id": "f001",
  "text": {
    "status": "generated",
    "div": "<div xmlns='http://www.w3.org/1999/xhtml'><p><b>Generated Narrative_
↳with Details</b></p><p><b>id</b>: f001</p><p><b>identifier</b>: 738472983_
↳(USUAL), ?? (USUAL)</p><p><b>active</b>: true</p><p><b>name</b>: Pieter van de_
↳Heuvel </p><p><b>telecom</b>: ph: 0648352638 (MOBILE), p.heuvel@gmail.com (HOME)</
↳p><p><b>gender</b>: male</p><p><b>birthDate</b>: 17/11/1944</p><p><b>deceased
↳</b>: false</p><p><b>address</b>: Van Egmondkade 23 Amsterdam 1024 RJ NLD (HOME)
↳</p><p><b>maritalStatus</b>: Getrouwd <span>(Details : {http://hl7.org/fhir/
↳v3/MaritalStatus code 'M' = 'Married', given as 'Married'})</span></p><p><b>
↳multipleBirth</b>: true</p><h3>Contacts</h3><table><tr><td>-</td><td><b>
↳Relationship</b></td><td><b>Name</b></td><td><b>Telecom</b></td></tr><tr><td>
↳*</td><td>Emergency Contact <span>(Details : {http://hl7.org/fhir/v2/0131_
↳code 'C' = 'Emergency Contact')</span></td><td>Sarah Abels </td><td>ph:
↳0690383372 (MOBILE)</td></tr></table><h3>Communications</h3><table><tr><td>-</
↳td><td><b>Language</b></td><td><b>Preferred</b></td></tr><tr><td>*</td><td>
↳Nederlands <span>(Details : {urn:ietf:bcp:47 code 'nl' = 'Dutch', given as 'Dutch'})
↳</span></td><td>true</td></tr></table><p><b>managingOrganization</b>: <a>
↳Burgers University Medical Centre</a></p></div>"
  },
  "identifier": [
    {
      "use": "usual",
      "system": "urn:oid:2.16.840.1.113883.2.4.6.3",
      "value": "738472983"
    },
    {
      "use": "usual",
      "system": "urn:oid:2.16.840.1.113883.2.4.6.3"
    }
  ],
  "active": true,
  "name": [
    {
      "use": "usual",
      "family": "van de Heuvel",
      "given": [
        "Pieter"
      ],
      "suffix": [
        "MSc"
      ]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "0648352638",
      "use": "mobile"
    },
    {
      "system": "email",
```

(continues on next page)

(continued from previous page)

```

    "value": "p.heuvel@gmail.com",
    "use": "home"
  }
],
"gender": "male",
"birthDate": "1944-11-17",
"deceasedBoolean": false,
"address": [
  {
    "use": "home",
    "line": [
      "Van Egmondkade 23"
    ],
    "city": "Amsterdam",
    "postalCode": "1024 RJ",
    "country": "NLD"
  }
],
"maritalStatus": {
  "coding": [
    {
      "system": "http://hl7.org/fhir/v3/MaritalStatus",
      "code": "M",
      "display": "Married"
    }
  ],
  "text": "Getrouwd"
},
"multipleBirthBoolean": true,
"contact": [
  {
    "relationship": [
      {
        "coding": [
          {
            "system": "http://hl7.org/fhir/v2/0131",
            "code": "C"
          }
        ]
      }
    ],
    "name": {
      "use": "usual",
      "family": "Abels",
      "given": [
        "Sarah"
      ]
    },
    "telecom": [
      {
        "system": "phone",
        "value": "0690383372",
        "use": "mobile"
      }
    ]
  }
],

```

(continues on next page)

(continued from previous page)

```
"communication": [
  {
    "language": {
      "coding": [
        {
          "system": "urn:ietf:bcp:47",
          "code": "nl",
          "display": "Dutch"
        }
      ],
      "text": "Nederlands"
    },
    "preferred": true
  }
],
"managingOrganization": {
  "reference": "Organization\\/f001",
  "display": "Burgers University Medical Centre"
}
}
```

Patient is essential in cases when the interpretation of an observation, e.g. height or weight, depends on the sex, age or other relevant information of the patient. In this case, one can retrieve Patient with the link in the “subject” field of Observation.

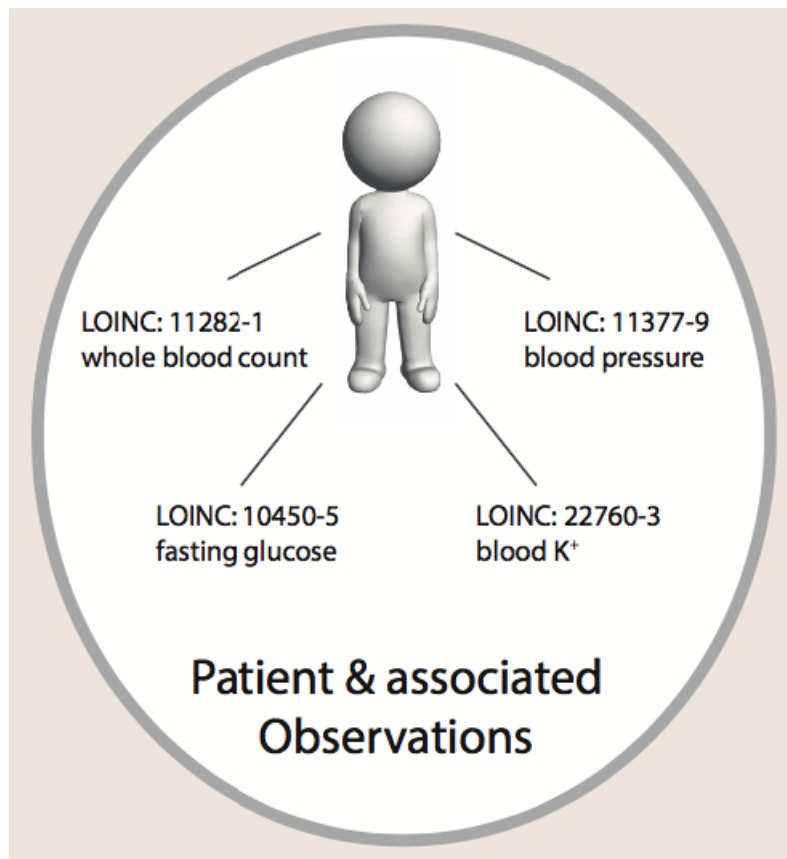
## CHAPTER 3

---

### Our Mission

---

Our goal with this app is to extract patient phenotypes in HPO terms from Electronic Health Records that complies with the FHIR standards: As illustrated below, each patient may have dozens or hundreds of associated medical observations. We hope our app will be able to analyze them and output a “phenopacket”, a list of HPO terms, that describes the patient.



To achieve our goal, we need to do the following:

- *Map LOINC codes to candidate HPO terms.* This step needs domain experts and will be done manually by our biocurators.
- *Retrieve patient observations.* We will use RESTful API technology to achieve this. The hapi-fhir library provides such functionalities so this can be done trivially. We will also collaborate with other institutions and hospitals to gain access to large patient cohorts.
- *Convert patient observations into phenopackets.* Our app will analyze the relevant fields from each observation and then output the corresponding HPO term that describes the patient. By analyzing all observations related to one patient, we will be able to create a phenopacket for this patient. We are still refining our app for this process.

Having a phenopacket for an individual is essential for disease prediction and patient clustering. We hope to extend our app so that it can be used by physicians to assist patient diagnosis; we also hope that the phenopackets can help academic researchers in study disease mechanisms.



The core mission of this tool is to convert medical laboratory observations from electronic health record into phenotypes coded in Human Phenotype Ontology terms. The app divides this goal into three subgoals:

- Map LOINC into candidate HPO terms. Each LOINC code corresponds to a set of HPO terms. For example, Loinc 10449-7 ("Glucose in Serum or Plasma -- 1 hour post meal") will be mapped to Hyperglycemia, hypoglycemia and Abnormality of blood glucose concentration.
- Retrieve resources from EHR systems. The app relies on EHR system that complies with FHIR standards. The most critical resources that are relevant to this app are Observation and Patient.
- Convert Observation into HPO terms. The app checks the outcome of an Observation and output the corresponding HPO term using the map generated in the first step.

In the following sections, we will discuss those goals in details.

## 4.1 Mapping LOINC to candidate HPO terms

LOINC observations have four main categories, quantitative(Qn), ordinal(Ord), nominal (Nom) and narrative(Nar). The majority of them are Qn (~50%) and Ord (~26%) and a smaller percentage are Nom and Nar type (<10%). The outcome of Qn and Ord type observations are typically represented with a code from the FHIR interpretation code valueset. Some examples are L for “low”, LL for “critically low”, or NEG for “negative”. The full valueset is included in Table 2. Therefore, if we map those interpretation codes to a candidate HPO term for each LOINC, we will be able to convert an observation into a HPO term.

However, the FHIR interpretation code valueset has 39 codes. It would be a massive task if we need to map 39 HPO terms for each LOINC code, let alone there could be more than one interpretation code system for different EHR systems. Fortunately, many interpretation codes are similar (e.g. FHIR L and LL both represent “low” but at different severity). So we picked 7 FHIR codes as our internal code to simplify the annotation job. Instead of annotating each each interpretation code, we will first map an interpretation code valueset into this internal FHIR subset (Table 1). Then we just need to annotate 7 values for each LOINC code (actually 6 because “U” does not need an annotation) instead of 39. Table 2 shows the map between FHIR interpretation code valueset to the FHIR subset. Similar maps can be built to deal with other interpretation code systems.

Table 1: Internal Code System

code system	FHIR
Code	Meaning
A	abnormal
L	low
N	normal
H	high
NEG	not present
POS	present
U	unknown

Table 2: FHIR interpretation code set Mapping to internal code system

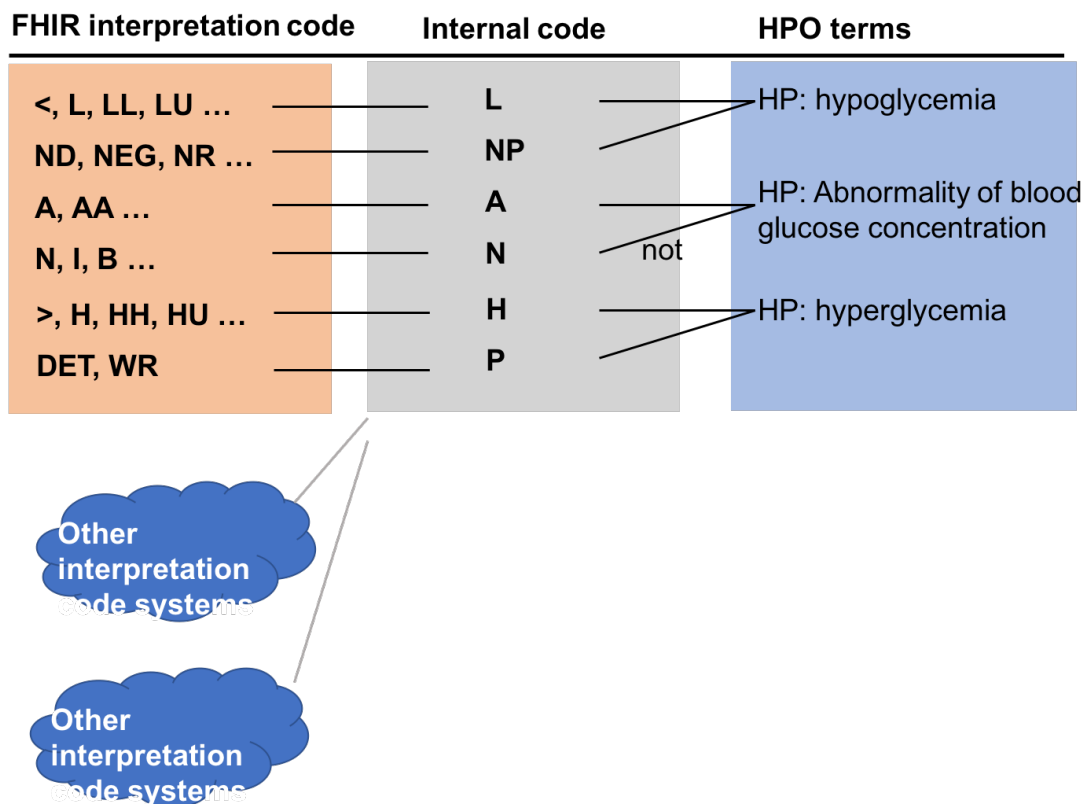
FHIR interpretation code value set	
system	<a href="http://hl7.org/fhir/v2/0078">http://hl7.org/fhir/v2/0078</a>
Code	Meaning
<	Off scale low
>	Off scale high
A	Abnormal
AA	Critically abnormal
AC	Anti-complementary substances present
B	Better
D	Significant change down
DET	Detected
H	High
HH	Critically high
HM	Hold for Medical Review
HU	Very high
I	Intermediate
IE	Insufficient evidence
IND	Indeterminate
L	Low
LL	Critically low
LU	Very low
MS	Moderately susceptible. Indicates for microbiology susceptibilities only.
N	Normal
ND	Not Detected
NEG	Negative
NR	Non-reactive
NS	Non-susceptible
null	No range defined or normal ranges don't apply
OBX	Interpretation qualifiers in separate OBX segments
POS	Positive
QCF	Quality Control Failure
R	Resistant
RR	Reactive
S	Susceptible
SDD	Susceptible-dose dependent
SYN-R	Synergy - resistant
SYN-S	Synergy - susceptible

Table 1 – continued from previous page

FHIR interpretation code value set	
system	<a href="http://hl7.org/fhir/v2/0078">http://hl7.org/fhir/v2/0078</a>
Code	Meaning
TOX	Cytotoxic substance present
U	Significant change up
VS	Very susceptible. Indicates for microbiology susceptibilities only.
W	Worse
WR	Weakly reactive

We can make the annotation task even easier. If we analyze our internal codes carefully, NEG (“not present”) and “POS”, resending an outcome that the measured value is only applicable to `Ord` type of LOINC that has “present” or “absent” as its outcome. On the other hand, “L”, “N”, “A”, “H” only apply to `Qn` type of LOINC. “A” and “N” will map to the same HPO term but have opposite negation value. What this means is that we only need to map 2 HPO terms to `Ord` LOINC with “presence”/“absence” outcome and 3 HPO terms to `Qn` LOINC.

The following graph summarize the annotation process and how the app convert a LOINC observation into HPO terms. We pick three HPO terms for each LOINC code, representing the desired phenotype to assign for the patient when the observation value is low, intermediate, or high. By default, those three terms will be mapped to the internal coding values, which further map to external code values used in real world.



- note:

`Ord` (non-“presence”/“absence” outcome), `Nom` and `Nar` observations can use other coding systems that are more difficult to handle. For example, `Loinc 600-7` or “Bacteria identified in Blood by Culture” may use a SNOMED concept to represent the finding that *Staphylococcus aureus* is detected:

```
"coding": [
{
  "system": "http://snomed.info/sct",
  "code": "3092008",
  "display": "Staphylococcus aureus"
}
]
```

To handle this type of outcomes, we allow annotating LOINC codes in the “advanced mode”. Under this mode, the user will assign a code from the coding system into a HPO term. In the above example, the user will say:

```
"system": "http://snomed.info/sct",
"code": "3092008"
```

map to HP: *Recurrent bacterial infections*. This workflow is actually what we used in the backend for annotating internal code values, but now user has to annotate in a more explicit way.

## 4.2 Retrieve Resources from EHR systems

We hope to allow at least two use cases with this app in real world. One is to allow patients to look at their own results, or allow physicians to look at a patient’s results. The second case is to allow academic researchers to retrieve a large cohort of patients and get a large data set containing patients’ phenotypes. The technology to handle both cases are very similar: both relies on REST api and a hapi-fhir Java library handles this task very nicely. We define a filter, either patient-specific or non-specific, and we retrieve patient resources from hospital EHR systems; then we can use a patients ID to retrieve all observations related to the person.

## 4.3 Convert Observations into HPO terms

We will describe how the app attempts to convert an observation into a HPO term in this section.

The app first tries to use the interpretation field in an observation. If there is one, the app checks whether we have a HPO term assigned for it. If there is one, it would be the desired output; but more likely there won’t be one. In the latter case, the app will try to convert the interpretation code into an internal code using the maps described in Table 2. If the app finds an internal code successfully, it will output the corresponding HPO term; if it fails or it cannot find a HPO term for the internal code, it will go to the next step.

If the first attempt fails, either because there is no interpretation field, or using the interpretation field did not find an HPO term, the app will check the value field. The first scenario is that it finds a coded value for the observation. In this case, the app will check whether there is a HPO term assigned for it and output the result if there is one. The second scenario, which is more likely in theory, is that there is a numeric value as the outcome of the observation, which means that the app has to make the last attempt to find the correct HPO term.

The central mission for the last attempt is to compare the measured value with the reference ranges, convert it into an internal code and then output the corresponding HPO term.

The key to successful annotation is to anticipate what kinds of outcome are for a LOINC code. Since the outcome is strongly associated with the `Scale` parameter of LOINC, we will describe three main types separately. For more detailed description of LOINC, refer to `Intro to LOINC`.

### 5.1 Qn

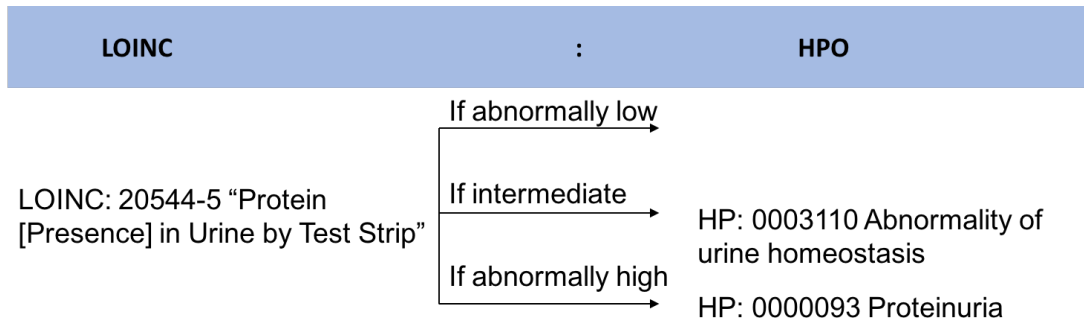
Qn or “quantitative” is the largest category of all LOINC codes, accounting for >80% in real world applications. Observations of this type have a numeric value as its outcome, such as the number of blood cells or the concentration of a substance. Annotating Qn is relatively straightforward: pick three HPO terms corresponding to the outcome of the observation when the measured value is low, intermediate or high. Here is an example for LOINC 2823-3 (“Potassium in Serum or Plasma”):

LOINC	:	HPO
LOINC: 10450-5 “Glucose [Mass/volume] in Serum of Plasma –10 hours fasting”	If abnormally low	HP: 0001943 Hypoglycemia
	If intermediate	NOT HP: 0011015 Abnormality of blood glucose concentration
	If abnormally high	HP: 0003074 Hyperglycemia

**Term Negation** As you can see from the above example, we assigned the negated form of “HP:0011015 Abnormality of blood glucose concentration” to the intermediate value. Intermediate value usually means “normal”, e.g. in this case, so by default the term that you choose for intermediate value is always negated. However, there are cases when this is not true, e.g. LOINC 9269-2 Glasgow coma score total, where the intermediate value is not interpreted as “no coma” but instead “mild coma”. So you will need to uncheck the “negate” checkbox in this case.

## 5.2 Ord

Ord or “ordinal” is the second largest category of LOINC codes, accounting for ~14%. Ordinal type of observation has a ordered set of values as its outcome. For example, “Presence, Absence”, “1, 2, 3”, but the values have no linear relationship to one another. Since ordinal results are comparable to each other, many observations also use the an interpretation code to indicate whether the result is too low, normal or too high. “Presence” or “positive” can be considered as a value that is “too high”, while “absence” or “negative” usually indicates that the measured value is not “too high”. Therefore, one simple way to annotate this type is to pick two HPO terms, one for “too high” and the other for “intermediate”:



In some occasions it might become necessary to annotate each value of the outcome. The procedure becomes very similar to how we annotate nominal type LOINC (see below).

## 5.3 Nom

Nom or “nominal” accounts for a small percentage of LOINC codes used in real world (~5%). Nominal observations also have a set of values as its outcomes but those values do not have an order. Therefore, one has to annotate each possible value for nominal observations (and some ordinal observations). We use LOINC 600-7 “Bacteria identified in Blood by Culture” as an example. The outcome could be any kind of bacteria that can infect the blood. As an example, we annotate the following finding of *Staphylococcus aureus* in blood to HP: 0002726 “Recurrent *Staphylococcus aureus* infections”, the best available HP term currently.:

```
"coding": [
{
  "system": "http://snomed.info/sct",
  "code": "3092008",
  "display": "Staphylococcus aureus"
}
]
```

However, it is probably not realistic to annotate every possible bacteria to an HPO term. For one thing, it is a huge task; for the other thing, it requires a term exists in HPO that matches the bacteria, which is not the case. We can summarize our finding by saying that the identification of any bacteria in the blood indicates bacteremia. Therefore, we convert the problem of mapping N outcomes to mapping 2 outcomes (“presence” or “absence” of bacteria in blood). Physicians indeed interpretate 600-7 observations in this manner. Even if such observations do not have such interpretations, we can always create one automatically.

## 5.4 Nar and other types

Other types of LOINC codes are much more heterogeneous, making their interpretation much more challenging. Since they only account for 1% of real world applications, we will not consider those LOINC codes for now. But in future, we may attempt to use those those resources with natural language processing, image analysis etc.





## CHAPTER 6

---

### Install & running

---

Java library to map LOINC-encoded test results to Human Phenotype Ontology

To build and run the GUI, use the following command.:

```
$ mvn clean package
$ java -jar loinc2hpogui/target/Loinc2HpoGui-jar-with-dependencies.jar
```

If you downloaded the jar releases directly, use the following command.:

```
//replace {$PATH} with the path to the jar file
$ java -jar {$PATH}/Loinc2HpoGui-jar-with-dependencies.jar
```



The app requires the following configurations the first time you run it.

### 7.1 Mandatory Settings

- Download the Loinc Core Table from [loinc.org](https://loinc.org). Follow instructions from the loinc.org website. You need to register before you can download the document. The current version is *Loinc Version 2.63* (2017/12/15 release).
- Configure the path to the Loinc core table. From the menu bar, click **“Configuration”** - **“Set path to Loinc Core Table file”** and point to the LoincTableCore.csv file.
- Download HPO file. From the menu bar, click **“Configuration”** - **“Download HPO file”**. The files (HPO in .obo and .owl formats) will be automatically downloaded.
- Set the path to auto-saved data. First clone the repository for loinc2hpoAnnotation from Github (<https://github.com/TheJacksonLaboratory/loinc2hpoAnnotation>)[{}<https://github.com/TheJacksonLaboratory/loinc2hpoAnnotation>]. Then from the menu bar, click **“Configuration”** - **“Set path to Auto-saved Data”**, point to the loinc2hpoAnnotation folder.

**Restart the app to apply all settings**

### 7.2 Optional Settings

The following setting are strongly recommended. Not specifying them will not affect the operation of the app.

- Set biocurator ID. From the menu bar, click **“Configuration”** - **“Set biocurator ID”**, specify your biocurator ID. If you are not assigned one, create one for yourself with the following format: organization name first followed by :, then your name/id.
- Once you are done, click **“Configuration”** - **“Show settings”** to view all your settings. The first two settings should **NOT** be null in order for the app to work correctly.

- Use customized hpo. If you prefer to use your own versions of hpo, click **“Configuration”** - **“Change hpo.owl”** to set the path to your hpo.owl file; use the button below to set the path to the hpo.obo file. An important note: inconsistencies of hpo.owl and hpo.obo files will lead to errors, so make sure your hpo OWL and OBO files are serialized from the same HPO. For details, refer to <https://github.com/TheJacksonLaboratory/loinc2hpoAnnotation/blob/master/README.md#annotating-with-newly-created-hpo-terms>

## 7.3 Change Settings

The settings will be saved to a local file so that you do not need to repeat the above steps every time you run the app. This applies even after you upgrade to a newer version of the app. Should you decide to change the settings, follow the above steps accordingly to overwrite the original settings.

## 8.1 Overview

The app has three tabs.

`Annotate`: contains functions for annotating LOINC codes (mapping LOINC codes to HPO terms);

`Loinc2HpoAnnotations`: displays the annotations created from the `Annotate` tab;

`Loinc2HpoConversion`: retrieves patient observations and convert to HPO terms.

## 8.2 Annotation

Before you start annotation, make sure that you have configured the app properly (see `Configuration`).

There are two modes of annotation, the basic mode and the advanced mode. For basic mode annotation, you simply pick 3 (or less) that correspond to the LOINC code when the observation value is abnormally low, normal, or abnormally high, and the app will map them to the correct codes; for advanced mode annotation, you have to specify the code and the corresponding HPO term for each LOINC code.

### 8.2.1 Quick Start

Follow the steps to start the curation process.

- Import LOINC codes. If you have configured the app properly, you should be able to see there are contents in the LOINC table. You can always click **“Initialize Loinc Table”** on the left upper corner to import Loinc codes from the Loinc Core Table file. Try using the “Search” function to select some Loinc codes, e.g. try searching for “10449-7” and then “glucose” (You will get one result for “10449-7” and many results for “glucose”).
- Import HPO. Click “Initialize HPO appTempData” on the left upper corner to import all HPO terms to the app. If you configured the app properly, this will automatically run when you start the app.

- After completing the above steps, you should be able to start annotating LOINC codes! - Go to the Loinc Table in the bottom half of the tab, and choose the LOINC code that you want to annotate. When you click the “**Auto Query**” button or double click on the LOINC code, the app will automatically find candidate HPO terms for you, listed from the most likely term to the least likely.
  - Choose the appropriate term and drag it to one of the three textfields. (Tip: you can drag another term to overwrite the current one; try to avoid manually typing to avoid spelling error).
  - After you are done with the current annotation, you can click “**Create annotation**” button to record your annotations. You are done with one LOINC code! Now if you go to the *Loinc2HpoAnnotations* tab, you will find the annotation in the table.

### A few helpful features

- Filter the Loinc table. You can choose a subset of LOINC code to annotate. To do this, create a .txt file with one Loinc code in each line (see the following example.) Save the file and click the *Filter* button. Point to the .txt file and you should get a much shorter list of Loinc codes in the table.:

```
15074-8
779-9
600-7
```

Alternatively, you can use the long common names of Loinc codes like below:

```
Potassium [Moles/volume] in Serum or Plasma
Hemoglobin [Mass/volume] in Blood
Erythrocyte distribution width [Ratio] by Automated count
```

You can switch to previously filtered lists by right-clicking the Loinc Table and choosing relevant buttons.

- Flag an annotation. If you are not certain about your annotation and wants to come back later, you can check the *Flag* checkbox as a reminder.
- Leave a note for your annotation. You can leave a note on your annotation whenever necessary, e.g. why annotate this way, why it needs revisit, etc..
- Find parent and child for an HPO term. When you double-click on a candidate HPO term, the app will find its parent(s) and child(ren) and display it in the right box. Sometimes this can be helpful in determining whether a HPO term should be chosen or not. (Tip: it can save time, too. e.g. If you double-click on “Abnormality of blood glucose concentration”, you will find “Hyperglycemia” and “Hypoglycemia” as its two children. Drag the children to fill the correct textfields, without needing to go through the long list of candidate terms looking for “Hyperglycemia” and “Hypoglycemia”!)
- Manually search for candidate HPO terms. If *Auto Query* does not give you the HPO terms that you need, try using the *Manual Query* button with comma-separated keys. Tip: *try synonyms*; words without comma will be taken as one key and the app will try to find a exact match to it.
- Group LOINC. Sometimes you may not be able to annotate a LOINC code because you are waiting for new HPO terms (see below: *Suggest new HPO terms*) or you simply do not know how to annotate it. You can group LOINC into a list so that your collaborators can help. Right click on the LOINC code, choose *Group/ungroup* and select a list. By default there will be two groups, “require\_new\_HPO\_terms” and “unable\_to\_annotate”. You can create more groups but the app does not display their colors very well yet (working on it!).

### Review/Edit your annotations

You can review all your annotations (sort of...continue to *Advanced Mode curation* to see why) under the *Loinc2HpoAnnotations* tab. Right-click a LOINC annotation and choose “Review” to see all annotations. Focus on the left two tables for the current being and we will deal with right table later.

Alternatively, you can review annotations in the *Annotate* tab. Select the Loinc code that you want to review and then click the *All annotations* button.

In both cases, you can edit LOINC annotation by click the *Edit* button. You may go to *Loinc2HpoAnnotations* tab, right-click a Loinc code and choose “Edit” directly. The app will bring you back to the annotation mode so that you can overwrite your annotations with new values. After you are done, click “Save” to save your new data.

## 8.2.2 Advanced Mode curation

Consider this Loinc test: *Loinc 600-7 or Bacteria identified in Blood by Culture*. The name of the Loinc code suggests two types of outcomes:

1. Whether there are bacteria identified in the blood or not (alternatively, we say whether the patient is “positive” or “negative” for bacterial infection). In this case, we choose “Recurrent bacterial infections” for positive and the same term in negated form for negative result and annotate in the Basic Mode as described above.
2. What type of bacteria is identified in the patient’s blood. Here is from a real-world example.

```
"coding": [
{
  "system": "http://snomed.info/sct",
  "code": "3092008",
  "display": "Staphylococcus aureus"
}
]
```

We can guess that the above lines indicate that the patient has *S. aureus* infection in his/her blood. In this case, our Basic Mode does not work well anymore because it only handles values that are too high, too low and intermediate. This is when Advanced Mode comes into play. To allow our app recognize this result, we need to assign a HPO term for **Snomed** code 3092008. To do this,

- Select Loinc 600-7 by using the “Search” function.
- Annotate Loinc 600-7 at the Basic Mode as described in last section. You may also skip this step to next one.
- Annotate at Advanced Mode. Click “advanced>>>” button and you will see three new textfields for *system*, *code*, and *hpo term*.
- Type in “*http://snomed.info/sct*” into *system*, “3092008” into *code*. (Note: the information in *system* and *code* is sufficient to encode a piece of information, *display* is only used for display purposes so we do not need it)
- Now we have to choose a HPO term. As an example, we double-click on “*Recurrent bacterial infections*” and drag one of its children “*Recurrent staphylococcal infections*” to the *hpo term* field. Click the + button to add this annotation.
- Repeat the above two steps if you have more codes to add. After we are done, click *Create annotation* button to complete.
- Now if you review your annotations for 600-7, you can see annotation data in the left bottom table. (This is why we said the table in *Loinc2HpoAnnotations* does not show all the annotations information—because it does not show data that were created for **Advanced Mode curation**)

Note: Pay attention to the strict proprietary right of Snomed codes. It may not be allowed to map to other codes.

**Term negation** Term negation means that you cannot find a HPO term that matches your need, but the opposite of a HPO term does. For example, if a patient’s blood glucose concentration is normal, we say that the inverse of “Abnormality of blood glucose concentration” best describes his/her phenotype.

Note: In the Basic Mode, the “**negate**” button only controls the term in the center textfield. The default value is *true* for Basic Mode, *false* for the Advanced Mode.

### 8.2.3 Suggest new HPO terms

Sometimes you may not be able to find an appropriate HPO term for a LOINC code. You can send a request to the authors of HPO directly from the app to ask for new terms.

- Create a new term for a Loinc code. Select a Loinc code and then click **“Suggest New HPO term”**. Provide the proposed term and your comment, type in your GitHub username and GitHub password, choose a label that best describes your request, e.g. *LOINC*, and click **“Create GitHub issue”**.
- Create a new child term for a Loinc code. If a current HPO term is close to what you need but you need a new child beneath it, you can select both the Loinc code and the candidate HPO term, right-click, select **“Suggest child term”**, fill in relevant information and submit.

Note:

1. If you do not have a GitHub account, you need to create one following their instructions ([GitHub website](<https://github.com>)).
2. The app currently does not support authentication with two-factor verifications [learn more](<https://github.com/blog/1614-two-factor-authentication>). If you enabled that feature on your account, you may encounter issues during submission.

### 8.2.4 Save & Export data

To save data, you can click *File - Save Session*. This will save your annotations into loinc2hpoAnnotation/Data/TSVSingle/annotations.csv. All those files are located at the folder that you specified for auto-saved data.

## 8.3 Converting Observations to HPO terms

The third tab “Loinc2HpoConversion” is responsible for converting patient observations into corresponding HPO terms using the annotation map generated above. Here we will demonstrate a few different ways to do this, which is largely dependent on the source of patient observations.

### 8.3.1 Convert locally stored observations to HPO terms

Assuming that you have some json serialization of patient observations, you can click *Local* to select the files (multi selection is supported). The app will import the files into the left text region. Then you can click the *convert* button to get the HPO term.

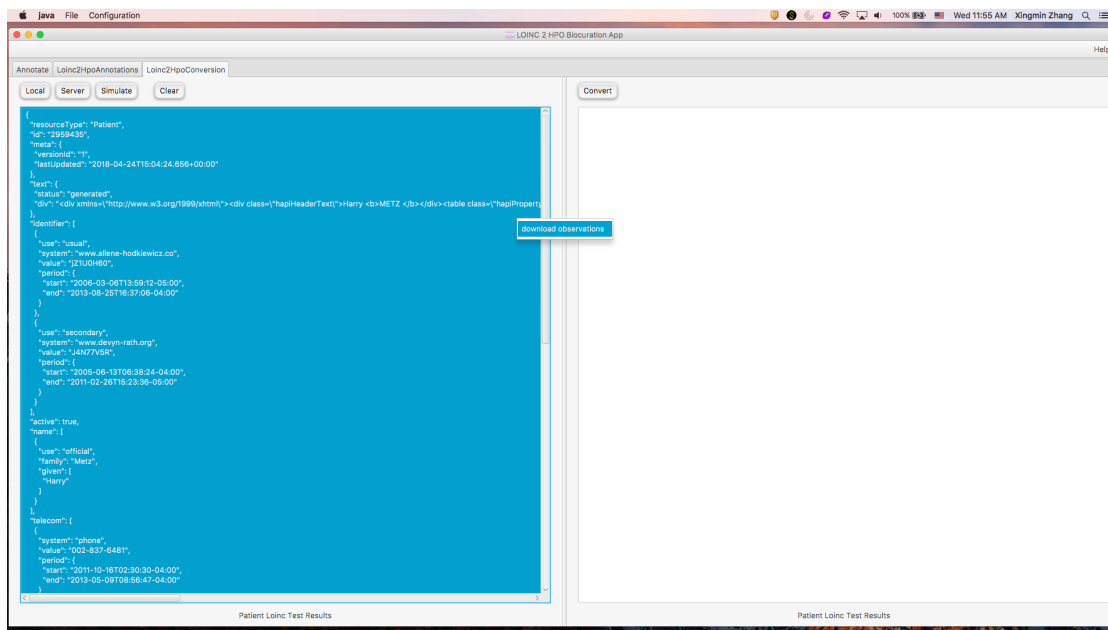
### 8.3.2 Download observations from FHIR server and convert to HPO terms

In real world, patient information and their observations are probably saved in a hospital FHIR server. In order to get the observations for a patient, you have to retrieve the patient first using his or her identification. To do this, click the *Server* button, type in the base url for the FHIR server, and provide relevant information to search for the patient.

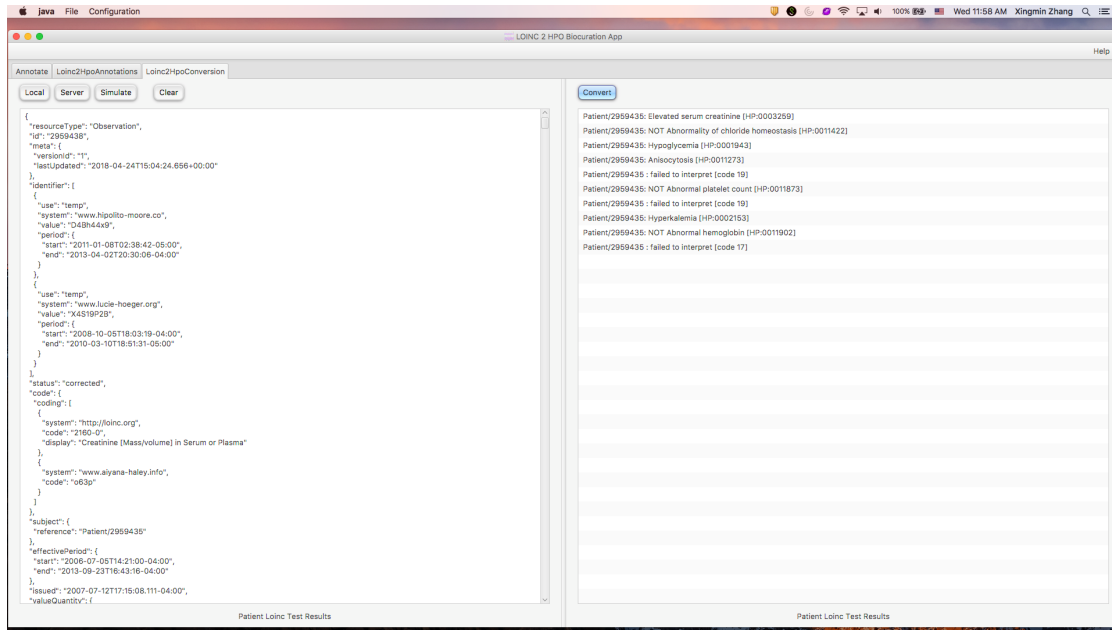
To demonstrate the process, we will search for a patient named “Harry Metz” with a phone number “002-837-6481” and postal code “79442-0781” from a test FHIR server (hapi-fhir test server: <http://fhirtest.uhn.ca>):



After clicking *confirm*, the patient is found on the server and loaded into the left text region.



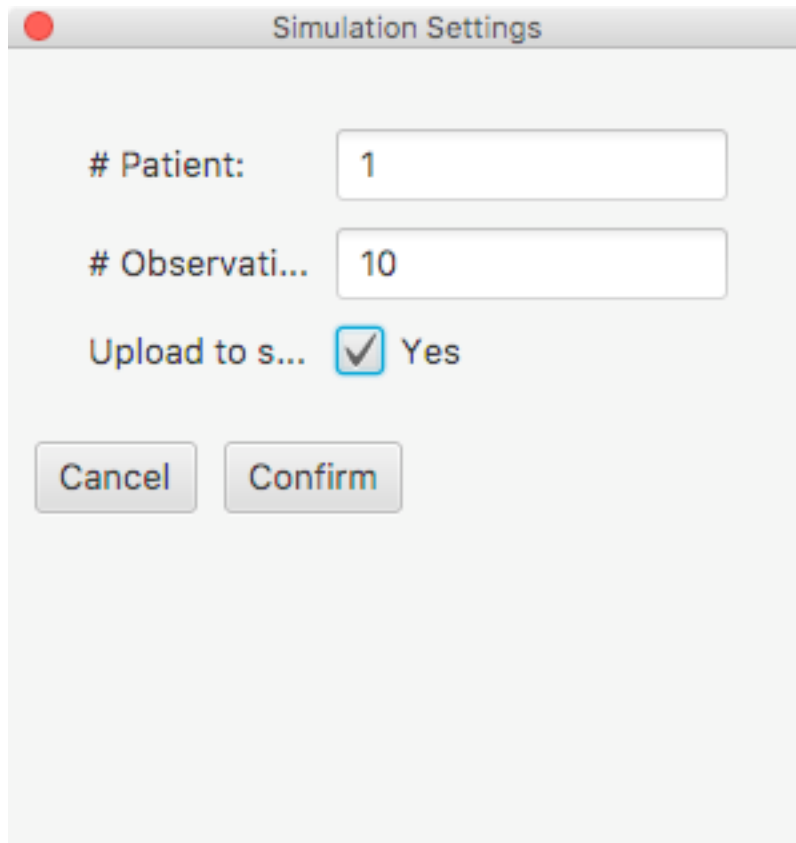
Next, you can get all observations related to this patient by right clicking on the patient record and then clicking “download observations”. The observations will be loaded to the left text region. Last, you can click *Convert* button to get a list of HPO terms, one for each observation, on the right text region.



Note that you may not be able to find such a patient if you try it yourself. This is because the hapi-fhir test server regularly purges their system so all data will be lost. We actually faked such a patient and uploaded to the test server right before we wrote the tutorial. Next, we are going to show you how we did this.

### 8.3.3 Simulate patients and observations

For demonstration purposes, you can create simulated patients and their observations by clicking the *Simulate* button. You will see a popup window that asks you how many patients you want to simulate. Currently, we choose to create 10 observation for each patient, 5 *Qn* type, 4 *Ord* type (“presence”/“absence” outcome) and 1 *Nom* type. Check the “Upload” checkbox and click *Confirm*. The app will generate fake patients and observations and then upload to the hapi-fhir test server. If it is successful, the patient information will populate the left text region. You can select a patient and request his/her observations from the server. Since the server has the patient, you are now able to search for it using his/her identifications.



A screenshot of a 'Simulation Settings' dialog box. The dialog has a title bar with a red close button and the text 'Simulation Settings'. Inside, there are three settings: '# Patient:' with a text box containing '1', '# Observati...' with a text box containing '10', and 'Upload to s...' with a checked checkbox and the text 'Yes'. At the bottom are two buttons: 'Cancel' and 'Confirm'.

# Patient:	1
# Observati...	10
Upload to s...	<input checked="" type="checkbox"/> Yes

Cancel Confirm

Note that the hapi-fhir server is not always stable. You can avoid uploading to server by not checking the *Upload* choice. If you do that, patient observations will directly populate the left text region.



## 9.1 Overview

In this tutorial, we are going to demonstrate how to use the core library (loinc2hpo-core) to transform LOINC-coded lab tests into HPO terms. Although our original paper focused on transforming FHIR-encoded lab tests into HPO term, the loinc2hpo tool can be used for lab tests encoded in any standards, such as i2b2, OMOP, PCORNET, **as long as** they are identified with LOINC codes and that you can interpret the results (see below). This tutorial expects you know Java and maven.

## 9.2 Installation

Because the library is not yet on maven central, you have to install the library before you can use it in your project.

Cd into the *loinc2hpo/* directory, then run

```
> mvn install
```

Then you can put the JAR file into your project classpath. If you are using maven to manage dependencies, put the following section into your pom file.

```
<dependency>
  <groupId>org.monarchinitiative</groupId>
  <artifactId>loinc2hpo-core</artifactId>
  <version>1.1.7-SNAPSHOT</version>
</dependency>
```

### 9.2.1 Quick Start

Download the annotation file from the [loinc2hpoAnnotation Github repository](#).

Instantiate a Loinc2Hpo object

```
Loinc2Hpo loinc2Hpo = new Loinc2Hpo(path_to_annotation_file);
```

For a given lab test, assume that you can extract the LOINC id and are able to interpret the result as L (lower than normal), N (normal), H (higher than normal), NEG (negative) or POS (positive), then call the following method:

```
public HpoTerm4TestOutcome query(LoincId loincId, String system, String id) throws_  
↳ LoincCodeNotAnnotatedException, AnnotationNotFoundException {  
    ...  
}
```

For example, suppose the test is LOINC 2823-3 Potassium in Serum or plasma, the result is L, then your code is:

```
HpoTerm4TestOutcome phenotype = loinc2hpo.query(new LoincId("2823-3"), "FHIR", "L");  
//is phenotype negated: false in this case (it would be true if the result is normal)  
System.out.println(phenotype.isNegated);  
//HPO term id: HP:0002900  
System.out.println(phenotype.getId().getValue());
```

## 9.2.2 Step-by-step tutorial

We are going to break down the transformation process into two steps.

I. In step one, interpret lab test result.

Most lab tests that you see are reported as numeric values. In order to use this library to map the result into an HPO term, it has to be compared with a reference range in order to assign an interpretation code, *L* for *lower than normal*, *N* for *normal* and *H* for *higher than normal*. It is a trivial task if your data comes with applicable reference range, but it can also be quite tricky. For one thing, it is possible that your dataset does not have reference ranges. For the other, it is possible that the reference ranges are too general and do not apply to every lab record, for example, you know the reference ranges in the general population but a lab test is on an infant. We want to point out that the library does not provide a solution to solve either issues, but we can give some suggestions based on our and other people's experience. For example, the most straightforward way is to ask the data provider (namely, clinical labs) to provide you their references if they are lacking in your dataset, and ideally to get the applicable reference range for each lab record. If that is not feasible, we saw people treating the entire population as “normal”, and define arbitrary thresholds as the reference ranges (for example, 2.5% and 97.5% percentiles as lower and upper normal range). We also noticed that in some datasets, a lab test is flagged if the result is abnormal. In this case, one can deduce the normal range from the unflagged subset.

For an individual lab test, extract the LOINC id (let's assign it to variable *loincId*). Assume you have determined an applicable reference range [*lowest\_normal\_value*, *highest\_normal\_value*], we can assign an interpretation code to the result:

```
String code_system = "FHIR";  
String code_id = null;  
if (result < lowest_normal_value) {  
    code_id = "L";  
} else if (result > highest_normal_value) {  
    code_id = "H";  
} else {  
    code_id = "N";  
}
```

Note that the variable *code\_system* specifies the namespace of the interpretation code. Internally, we used a subset of FHIR interpretation codes for our annotation file. Therefore, this value is always “FHIR” (see Exception).

If the LOINC term is expected to report a binary value, absence or presence, the interpretation code should be “NEG” or “POS”, respectively. We noticed real world data is quite messy in reporting this type of lab tests. Some datasets uses free texts in the result field with phrases like “positive”, “POSITIVE”, “negative”, “abnormal”. If this is the case, using syntactic matching might suffice. In other datasets, the original numerical values are reported. In this case, one still have to determine the applicable reference range and manually transform them into binary results.

```
//syntactic matching
if (result.toLowerCase().contains("pos") {
    code_id = "POS";
} else if (result.toLowerCase().contains("neg") {
    code_id = "NEG";
} else {
    //handle this case properly
    System.out.println("unknown result");
}

//comparing with reference ranges
if (result > threshold) {
    code_id = "POS";
} else {
    code_id = "NEG";
}
```

**Exception** A small subset of our annotations used SNOMED-CT concepts, in which case the namespace is “snomed-ct” and the code is using SNOMED-CT concept id. These lab tests are called nominal in the LOINC term. For example, if you want to convert the result of lab test LOINC 5778-6 Color of Urine into HPO term, you need to query with the SNOMED-CT concepts:

```
code_system = "snomed-ct"
code_id = "449071000124107";
```

II. After you have successfully interpreted the result with a code, the remaining task is to call the query method.

You need to download the annotation file from our [loinc2hpoAnnotation Github repository](#). Instantiate the Loinc2Hpo class with the path to the annotation file:

```
Loinc2Hpo loinc2Hpo = new Loinc2Hpo(path_to_annotation_file);
```

Then call the query method to return the phenotype term.

```
HpoTerm4TestOutcome phenotype = loinc2hpo.query(new LoincId(loincId), code_system,
↪code_id);
//print out the result
System.out.println(phenotype.isNegated());
System.out.println(phenotype.getId().getValue());
```

You will get a *HpoTerm4TestOutcome* instance as the result, which basically contains two pieces of information, an HPO term, and a boolean value indicating whether the term should be negated. The negation value is false if the result is abnormal, and true if the result is normal. For example, the resulting HPO is Hypokalemia HP:0002900 for blood potassium measurement that is too low, and NOT Abnormal blood potassium concentration HP:0011042 if the measurement is within reference range. We have to use the negation because HPO does not encode normal phenotypes.





## CHAPTER 10

---

### Contact

---

**Prof. Peter Robinson**

[Peter.Robinson@jax.org](mailto:Peter.Robinson@jax.org)

**Dr. Xingmin(“Aaron”) Zhang** [Aaron.Zhang@jax.org](mailto:Aaron.Zhang@jax.org)

The Jackson Laboratory for Genomic Medicine

Farmington, CT USA



## CHAPTER 11

---

GitHub repo

---

The source code of Loinc2hpo can be found at GitHub: <https://github.com/monarch-initiative/loinc2hpo>