OncoTime

Domain Specific Languages

COMP 520: Compiler Design (4 credits)
Professor Laurie Hendren
hendren@cs.mcgill.ca

OncoTime

Designing Domain-Specific Languages (DSLs)

- Specialize language for domain-specific task.
- Avoid the urge to include all of your favourite language constructs.
- Make the language easy to use for the domain specialist.
- Decide on the right data and control abstractions.
- Try to reduce excess notation - are type declarations necessary?
- Ensure that you will be able to generate “efficient-enough" code for each construct.
- Decide whether a DSL or a specialized library is a better solution.
- Decide on the target language - aim for portability.
- Consider interfacing with a known host language in order to leverage existing libraries and for developing DSL libraries.

OncoTime is an experimental DSL for a real problem

- The real problem is designing a DSL which allows for easily analyzing, verifying and visualizing patient treatment paths in radiation oncology.
- OncoTime design motivated by real problems currently being solved “by hand coding" in the HIG group.
- Since the language is experimental, groups will be allowed to make small modifications and additions to the OncoTIme language. All such modifications/additions should be documented and justified.
- Since the language is addressing a real problem, the projects may be prototypes of a system that is eventually used.

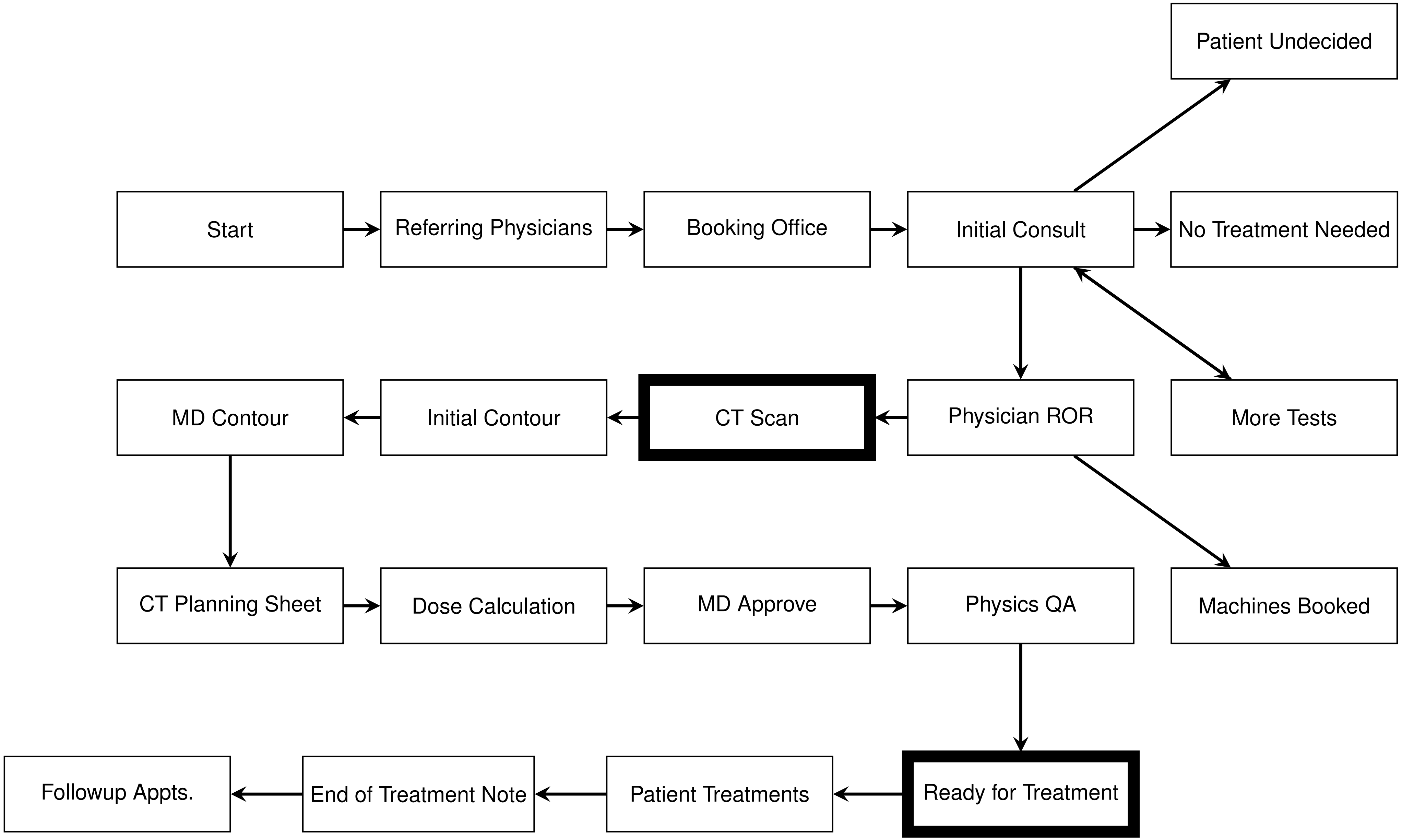
Radiation Oncology at the MUHC

- Radiation Oncology is the use of radiation to treat cancer and other diseases.
- About 1 in 3 people in the western world will develop cancer in their lifetime, and about half of those will require radiation. `http://ranzcr.edu.au/about/radiation-oncology`.
- Radiation Oncology in MUHC currently centered at MGH, but soon to move to the new cancer centre at the Glen site.
- Last year they treated 2,500 patients for a total of 40,000 treatments.
- The entire operation of the radiation oncology department is computerized, with a database containing complete information about treatments, documents, appointments and so on.
- For our project we have access to an anonymized version of the key parts of the database, so we will be able to use the OncoTime compilers to compile OncoTime programs which compute real, and potentially useful information.

A typical patient pathway through radiation oncology treatment

For a video and link to the patient information brochure, please visit `http://muhc.ca/cancer/radiation-oncology`. This should give you some idea of the process the patient goes through.

`http://acfro.com/wp-content/uploads/2012/07/image003.jpg`


Patient Undecided
Start
Referring Physicians
Booking Office
Initial Consult
No Treatment Needed
MD Contour
Initial Contour
CT Scan
Physician ROR
More Tests
CT Planning Sheet
Dose Calculation
MD Approve
Physics QA
Machines Booked
Followup Appts.
End of Treatment Note
Patient Treatments
Ready for Treatment

Patient Treatments

- Patient treatments are given every weekday, often for many weeks. For example, 5 weeks (25 treatments) is common for breast cancer.
- Each week the patient also has an appointment with his/her radiation oncologist, and may have other appointments with other medical professionals, such as nutritionists.
- For each appointment there are several times. The patient scans their medicare card when they arrive, this is the *arrival time*. Then when the patient is moved to the consultation room this gives us the *actual consult start time*. Each appointment also has a *scheduled start time*. One could imagine that there are many interesting questions to ask about waiting times for patients.

Time Series View of the Data

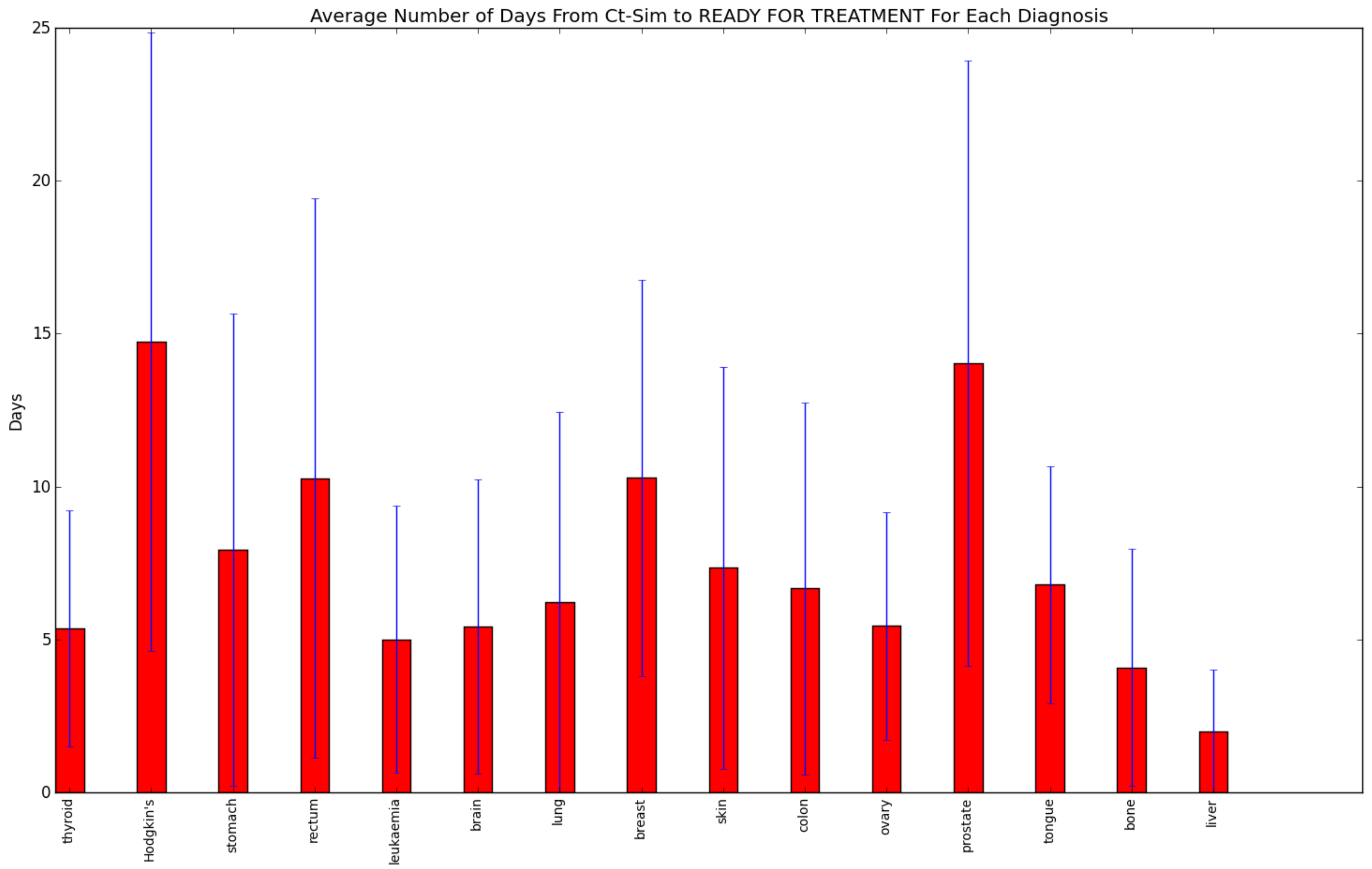
- You will note that the entire process is described as a series of events, ordered by time of the event.
- From the patient perspective, this series of events is also how they see the process.
- We would like to compute over the time series to ask questions such as:
  - What are all the events, ordered by time, for patient P?
  - How many (Which) consults were performed on breast cancer patients in January?
  - How many (Which) consults led to a ROR (RadOnc Requistion)?
  - What is the expected delay between **CT Scan** and **Ready for Treatment**?
  - How long has patient P been waiting for previous consultations?
  - Do Dr. D's patients wait longer than average for planning and/or appointments?
  - Do patients wait longer on Monday mornings?
  - How many (Which) patients check-in more than 1 hour early?
  - Does checking in early result in being seen early?
  - How many patients who have finished treatments have the **End of Treatment** note finished within two weeks?

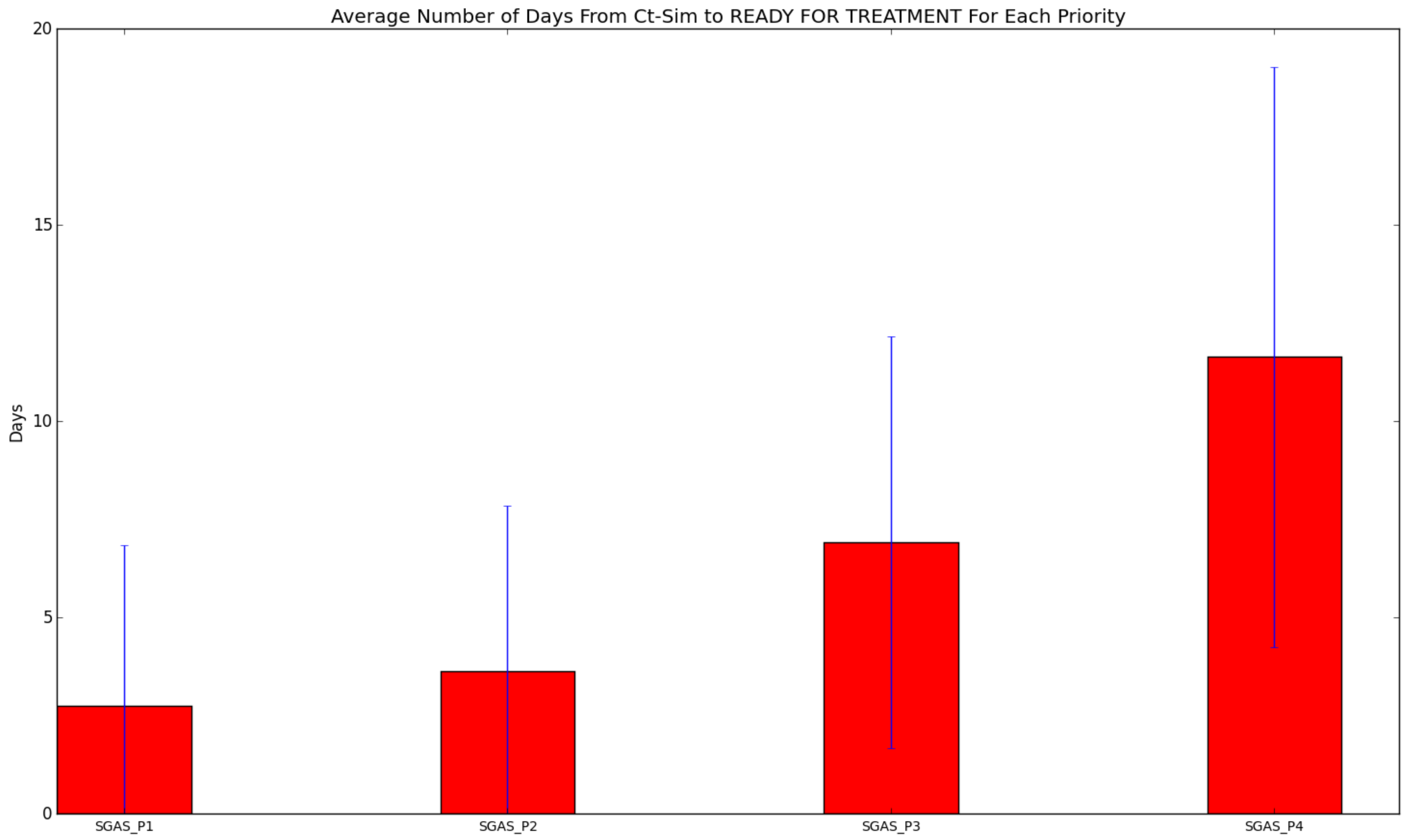
Filtering Data

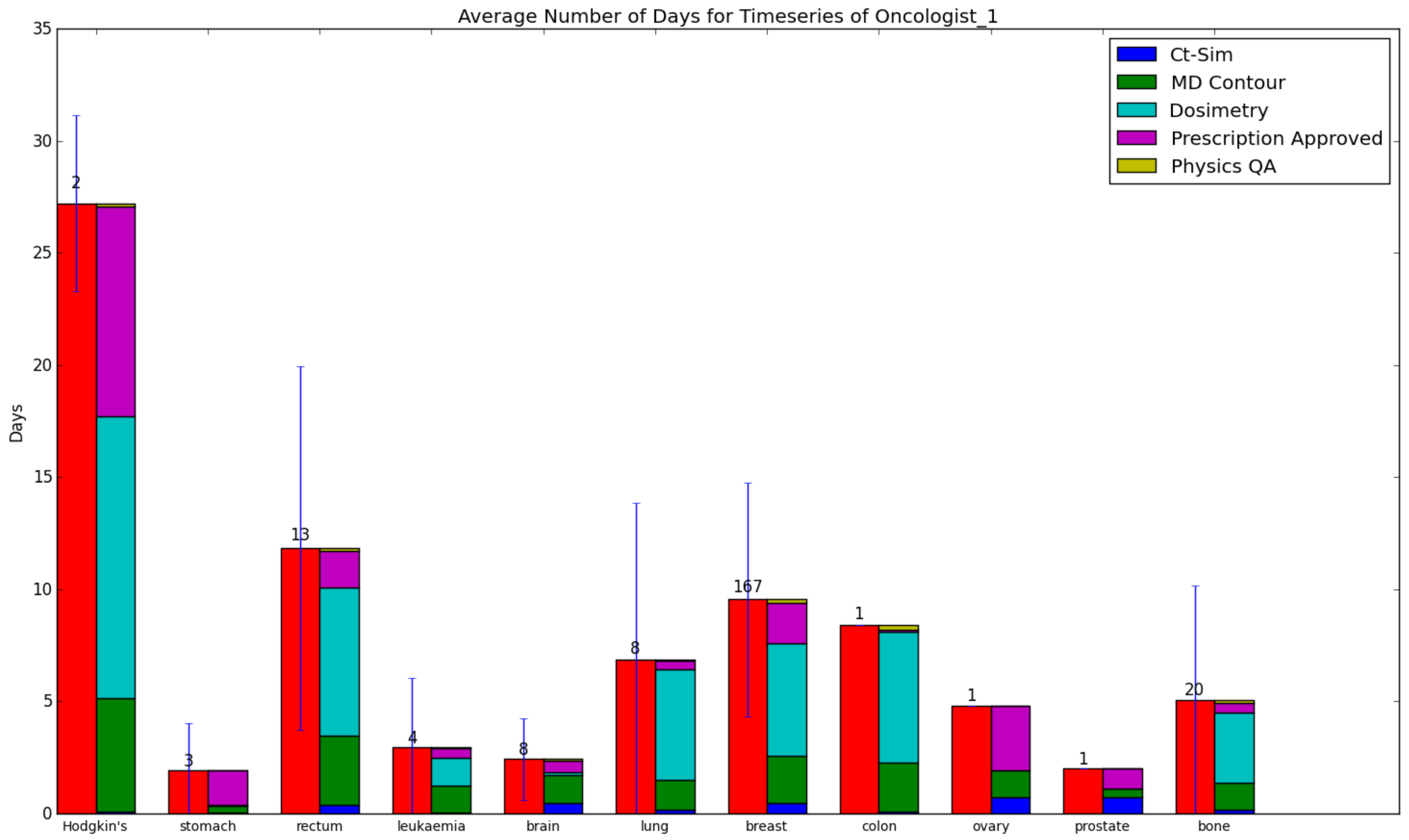
- Many of the questions we want to ask are relevant only to part of the data, for example:
  - We may be interested only in certain time periods: between given dates, only Mondays, only the mornings.
  - We may only want to look at events relevant to patients with a specific diagnosis (i.e. breast cancer, prostate cancer, lung cancer, ...).
  - We may only want to look at some of the patients: only men, only patients born between 1990 and now, only patients from a specific postal code, only patient $\boldsymbol{P}$, ...

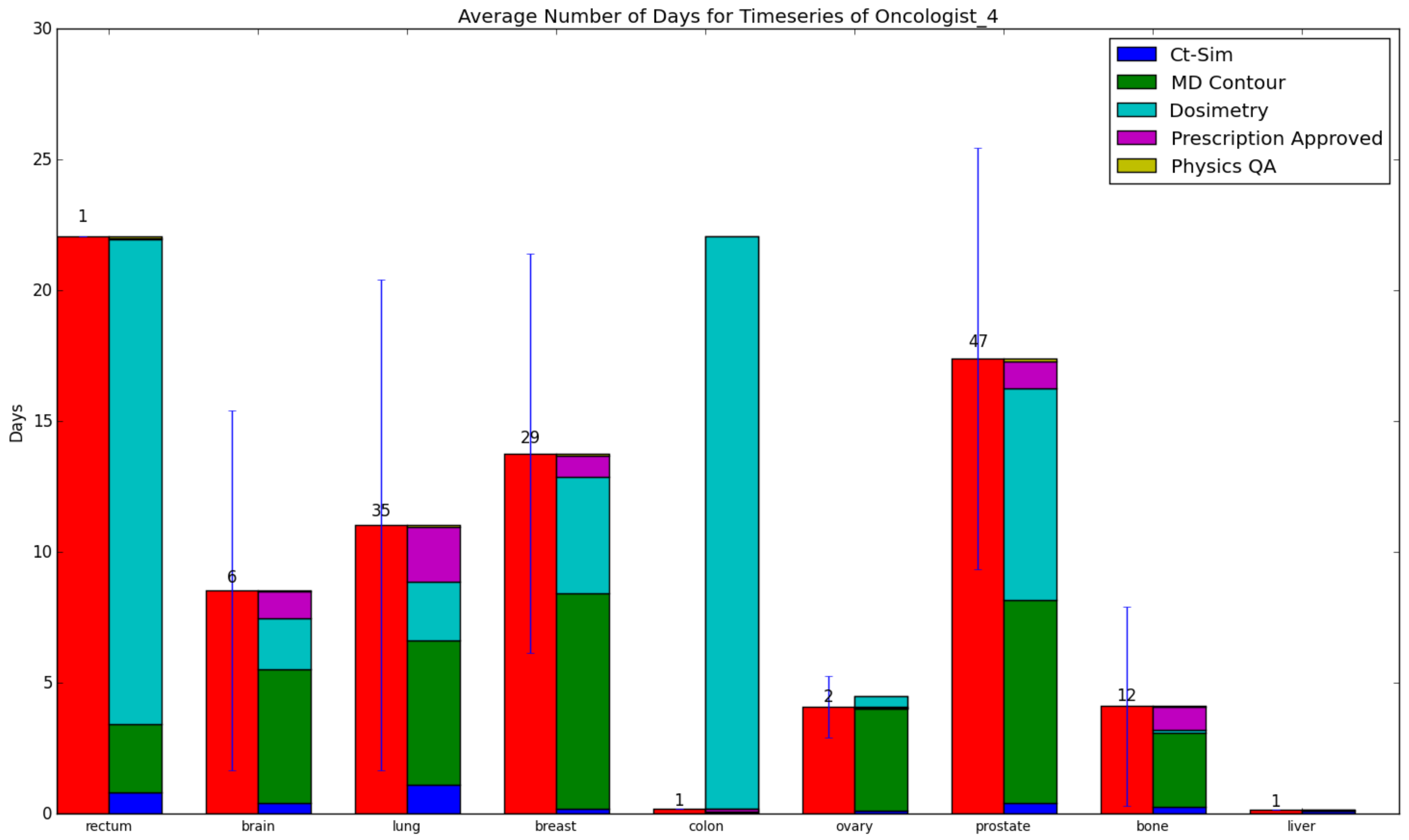
Waiting Time for Planning

- Look at the time for planning for patients who had six essential events, in the correct order, **CT-SIM**, **Ready for MD Contour**, **Ready for Dosimetry**, **Prescription Approved**, **Ready for Physics QA**, and **Ready for Treatment**.
- Visualize the data in many different ways.
- Eventually develop a predictor, which can fairly accurately determine the expected number of days before the patient can begin treatment, based on many factors.


Average Number of Days From Ct-Sim to READY FOR TREATMENT For Each Diagnosis
Days
0
5
10
15
20
25
thyroid
Hodgkin's
stomach
rectum
leukaemia
brain
lung
breast
skin
colon
ovary
prostate
tongue
bone
liver

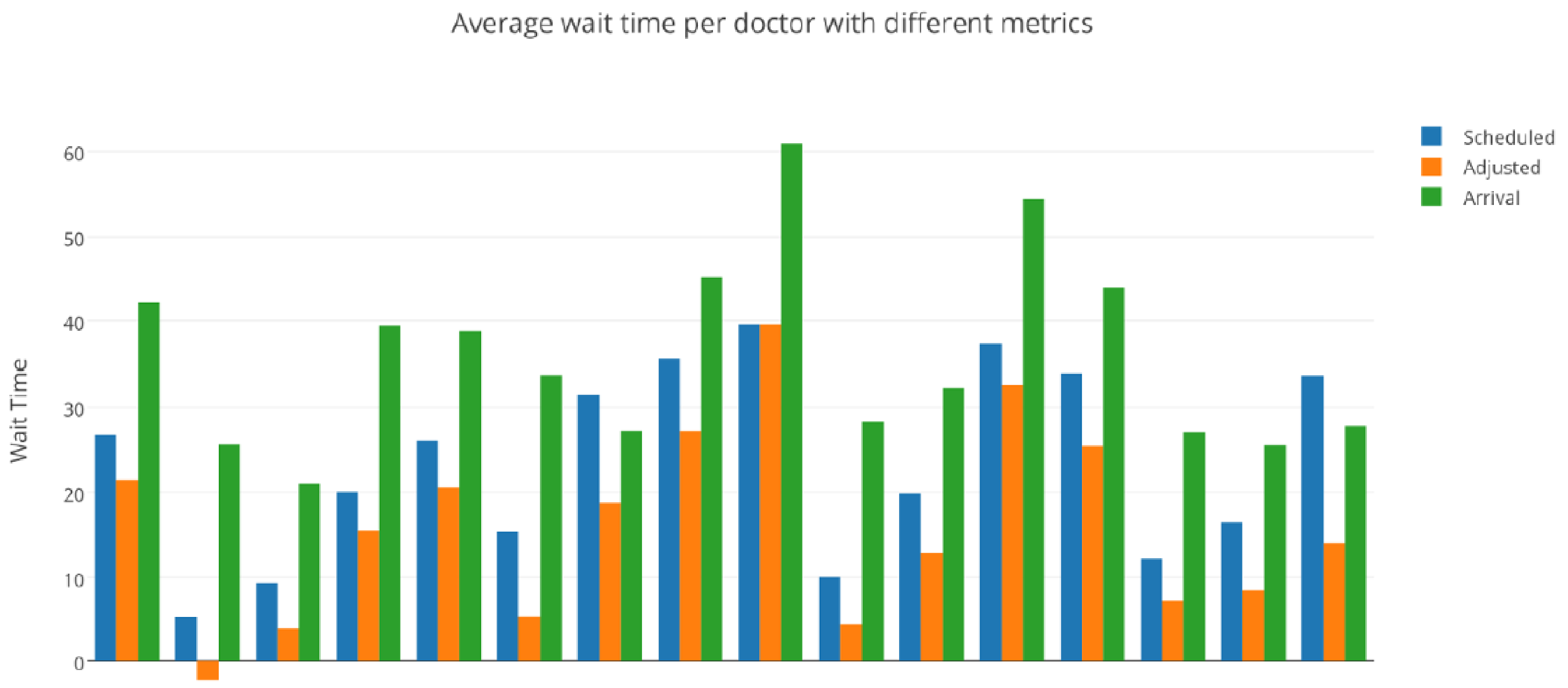

Average Number of Days From Ct-Sim to READY FOR TREATMENT For Each Priority
Days
0
5
10
15
20
SGAS_P1
SGAS_P2
SGAS_P3
SGAS_P4

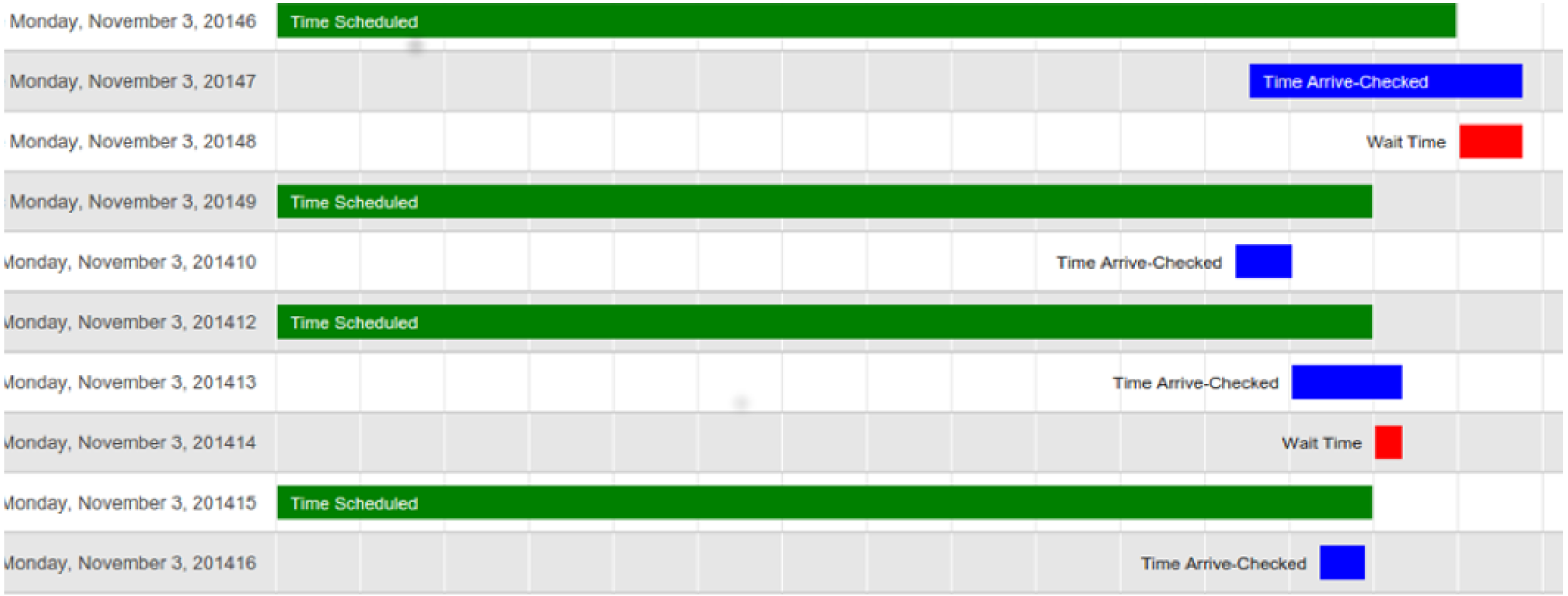

Average Number of Days for Timeseries of Oncologist_1
Days
0
5
10
15
20
25
30
35
Ct-Sim
MD Contour
Dosimetry
Prescription Approved
Physics QA
Hodgkin's
stomach
rectum
leukaemia
brain
lung
breast
colon
ovary
prostate
bone
2
3
13
4
8
8
167
1
1
1
20


Average Number of Days for Timeseries of Oncologist_4
Days
Ct-Sim
MD Contour
Dosimetry
Prescription Approved
Physics QA
rectum
brain
lung
breast
colon
ovary
prostate
bone
liver
1
6
35
29
1
2
47
12
1
0
5
10
15
20
25
30

Waiting time for Consults

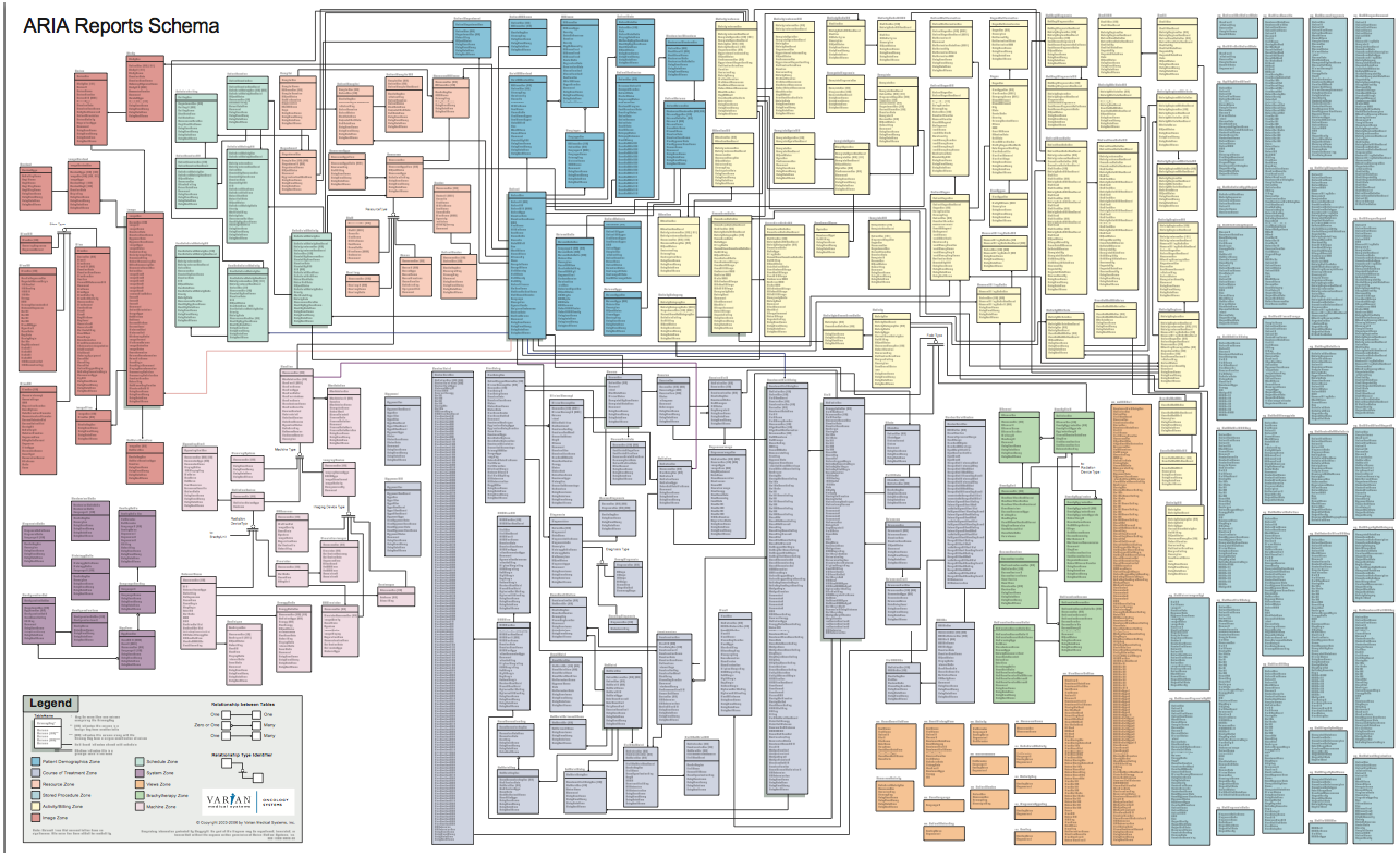
- Patients also come for many different types of appointments, including the initial consult, weekly consultations during treatments, and follow-up consults.
- It is interesting to study the waiting time for these patients, in order to better predict the expected waiting time, and to identify undesirable situations, such as patients arriving much too early, doctors scheduling too many patients during some time periods, and patients who are habitually late.

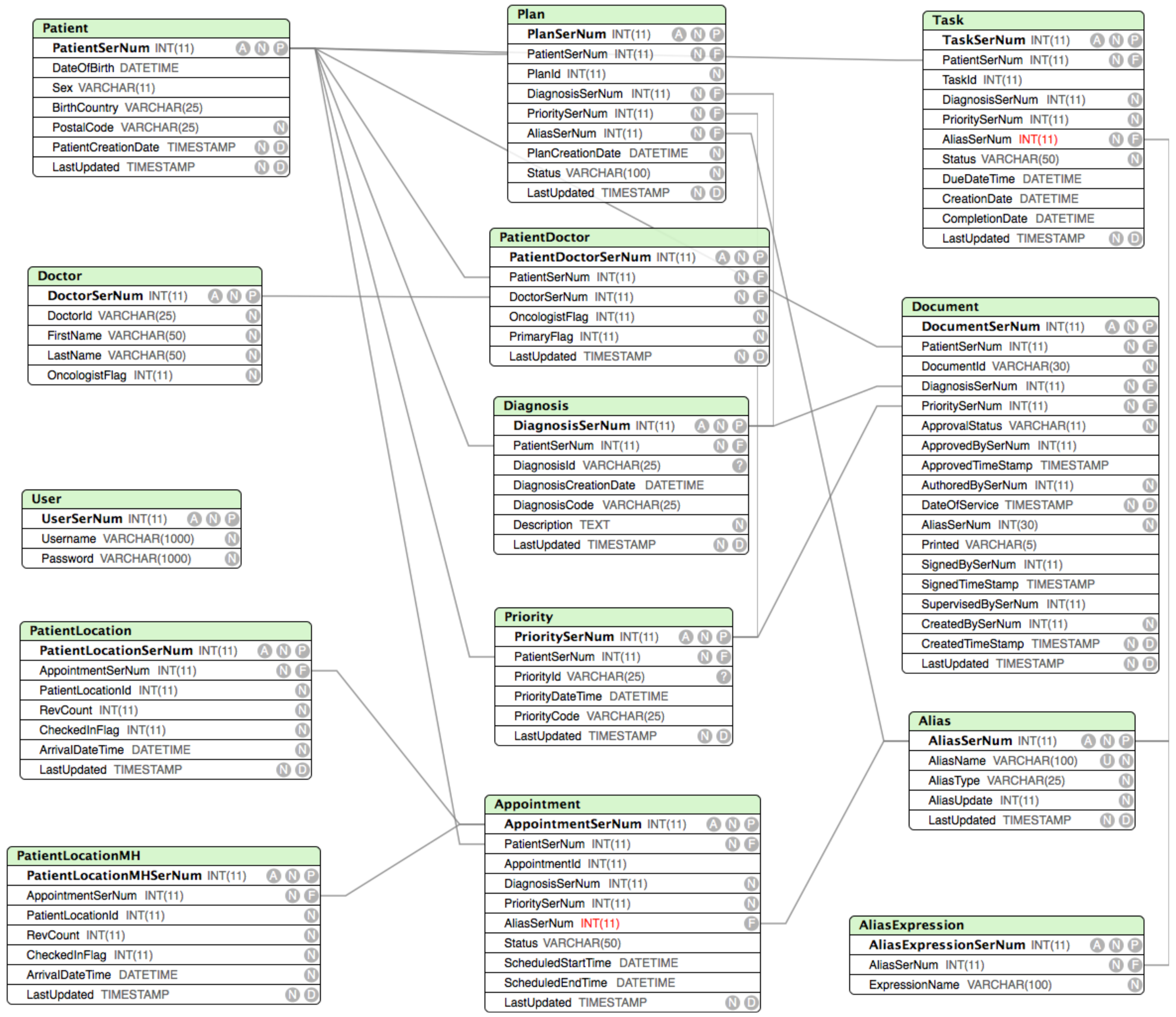

Average wait time per doctor with different metrics
Wait Time
60
50
40
30
20
10
0
Scheduled
Adjusted
Arrival

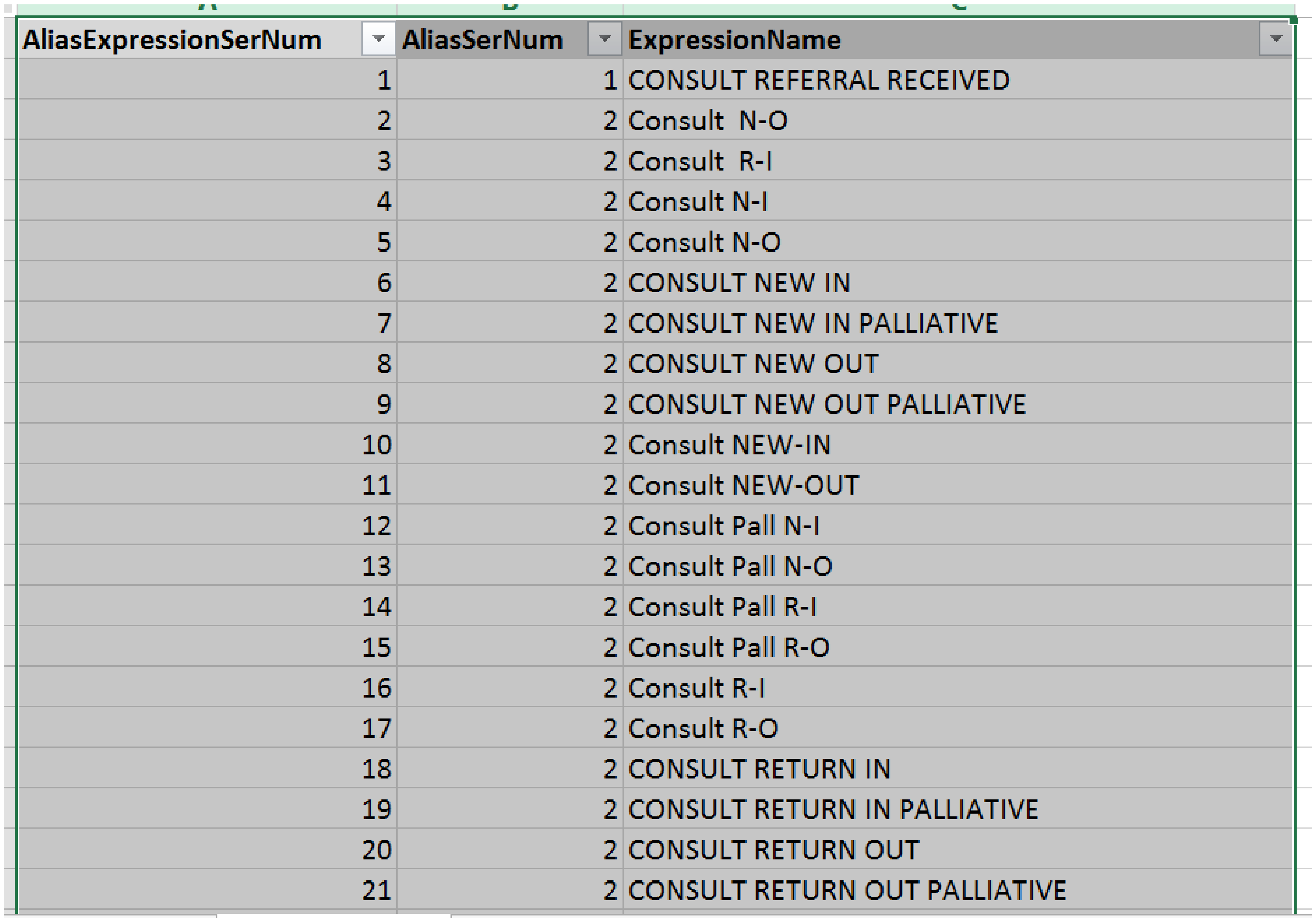

Monday, November 3, 20146
Time Scheduled
Monday, November 3, 20147
Time Arrive-Checked
Monday, November 3, 20148
Wait Time
Monday, November 3, 20149
Time Scheduled
Monday, November 3, 201410
Time Arrive-Checked
Monday, November 3, 201412
Time Scheduled
Monday, November 3, 201413
Time Arrive-Checked
Monday, November 3, 201414
Wait Time
Monday, November 3, 201415
Time Scheduled
Monday, November 3, 201416
Time Arrive-Checked

How is the data stored?

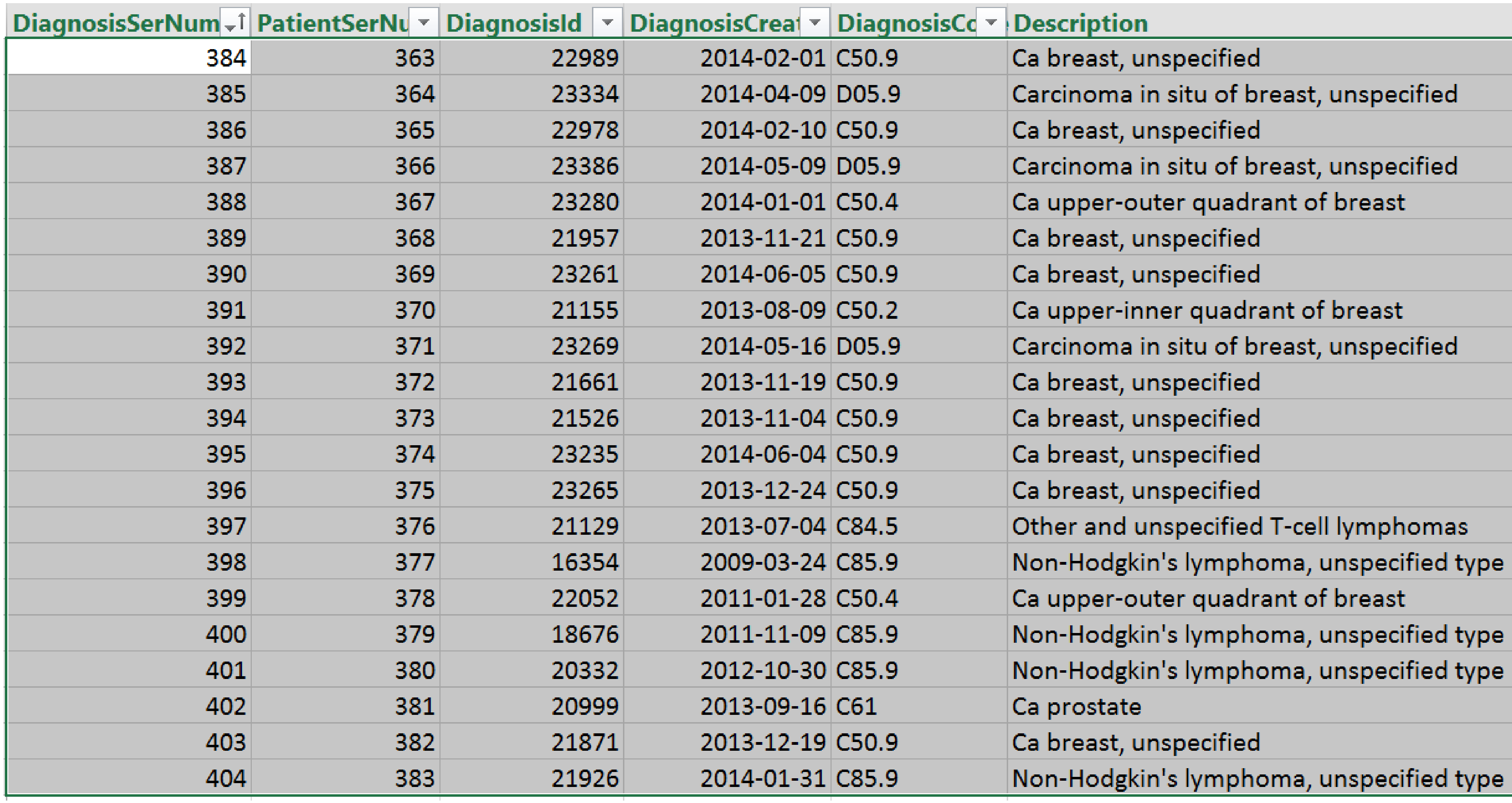
- The production database, Aria, is very large and complex.
- We are going to use a much simpler MySQL database.
- Events are encoded implicitly in the MySQL database.
- Previous studies have combined SQL queries with some post-processing.
- There are many ways of encoding similar values, and we need some way of grouping them (these groups were called aliases in a related project, `http://scitation.aip.org/content/aapm/journal/medphys/41/8/10.1118/1.4894911`).


ARIA Reports Schema
Legend
Relationship between Tables
One
One
Zero or One
Many
One
Many
Relationship Type Identifier
Patient Demographics Zone
Course of Treatment Zone
Resource Zone
Stored Procedure Zone
Activity/Billing Zone
Image Zone
Schedule Zone
System Zone
Views Zone
Brachytherapy Zone
Machine Zone
VARIAN medical systems
ONCOLOGY SYSTEMS
© Copyright 2003-2008 by Varian Medical Systems, Inc.


Patient
PatientSerNum INT(11) A N P
DateOfBirth DATETIME
Sex VARCHAR(11)
BirthCountry VARCHAR(25)
PostalCode VARCHAR(25) N
PatientCreationDate TIMESTAMP N D
LastUpdated TIMESTAMP N D
Doctor
DoctorSerNum INT(11) A N P
DoctorId VARCHAR(25) N
FirstName VARCHAR(50) N
LastName VARCHAR(50) N
OncologistFlag INT(11) N
User
UserSerNum INT(11) A N P
Username VARCHAR(1000) N
Password VARCHAR(1000) N
PatientLocation
PatientLocationSerNum INT(11) A N P
AppointmentSerNum INT(11) N F
PatientLocationId INT(11) N
RevCount INT(11) N
CheckedInFlag INT(11) N
ArrivalDateTime DATETIME N
LastUpdated TIMESTAMP N D
PatientLocationMH
PatientLocationMHSerNum INT(11) A N P
AppointmentSerNum INT(11) N F
PatientLocationId INT(11) N
RevCount INT(11) N
CheckedInFlag INT(11) N
ArrivalDateTime DATETIME N
LastUpdated TIMESTAMP N D
Plan
PlanSerNum INT(11) A N P
PatientSerNum INT(11) N F
PlanId INT(11) N
DiagnosisSerNum INT(11) N F
PrioritySerNum INT(11) N F
AliasSerNum INT(11) N F
PlanCreationDate DATETIME N
Status VARCHAR(100) N
LastUpdated TIMESTAMP N D
PatientDoctor
PatientDoctorSerNum INT(11) A N P
PatientSerNum INT(11) N F
DoctorSerNum INT(11) N F
OncologistFlag INT(11) N
PrimaryFlag INT(11) N
LastUpdated TIMESTAMP N D
Diagnosis
DiagnosisSerNum INT(11) A N P
PatientSerNum INT(11) N F
DiagnosisId VARCHAR(25) ?
DiagnosisCreationDate DATETIME
DiagnosisCode VARCHAR(25)
Description TEXT N
LastUpdated TIMESTAMP N D
Priority
PrioritySerNum INT(11) A N P
PatientSerNum INT(11) N F
PriorityId VARCHAR(25) ?
PriorityDateTime DATETIME
PriorityCode VARCHAR(25)
LastUpdated TIMESTAMP N D
Appointment
AppointmentSerNum INT(11) A N P
PatientSerNum INT(11) N F
AppointmentId INT(11)
DiagnosisSerNum INT(11) N
PrioritySerNum INT(11) N
AliasSerNum INT(11) F
Status VARCHAR(50)
ScheduledStartTime DATETIME
ScheduledEndTime DATETIME
LastUpdated TIMESTAMP N D
Task
TaskSerNum INT(11) A N P
PatientSerNum INT(11) N F
TaskId INT(11)
DiagnosisSerNum INT(11) N
PrioritySerNum INT(11) N
AliasSerNum INT(11) N F
Status VARCHAR(50) N
DueDateTime DATETIME
CreationDate DATETIME
CompletionDate DATETIME
LastUpdated TIMESTAMP N D
Document
DocumentSerNum INT(11) A N P
PatientSerNum INT(11) N F
DocumentId VARCHAR(30) N
DiagnosisSerNum INT(11) N F
PrioritySerNum INT(11) N F
ApprovalStatus VARCHAR(11) N
ApprovedBySerNum INT(11)
ApprovedTimeStamp TIMESTAMP
AuthoredBySerNum INT(11) N
DateOfService TIMESTAMP N D
AliasSerNum INT(30) N
Printed VARCHAR(5)
SignedBySerNum INT(11)
SignedTimeStamp TIMESTAMP
SupervisedBySerNum INT(11)
CreatedBySerNum INT(11) N
CreatedTimeStamp TIMESTAMP N D
LastUpdated TIMESTAMP N D
Alias
AliasSerNum INT(11) A N P
AliasName VARCHAR(100) U N
AliasType VARCHAR(25) N
AliasUpdate INT(11) N
LastUpdated TIMESTAMP N D
AliasExpression
AliasExpressionSerNum INT(11) A N P
AliasSerNum INT(11) N F
ExpressionName VARCHAR(100) N



AliasExpressionSerNum	AliasSerNum	ExpressionName
1	1	CONSULT REFERRAL RECEIVED
2	2	Consult N-O
3	2	Consult R-I
4	2	Consult N-I
5	2	Consult N-O
6	2	CONSULT NEW IN
7	2	CONSULT NEW IN PALLIATIVE
8	2	CONSULT NEW OUT
9	2	CONSULT NEW OUT PALLIATIVE
10	2	Consult NEW-IN
11	2	Consult NEW-OUT
12	2	Consult Pall N-I
13	2	Consult Pall N-O
14	2	Consult Pall R-I
15	2	Consult Pall R-O
16	2	Consult R-I
17	2	Consult R-O
18	2	CONSULT RETURN IN
19	2	CONSULT RETURN IN PALLIATIVE
20	2	CONSULT RETURN OUT
21	2	CONSULT RETURN OUT PALLIATIVE



DiagnosisSerNum	PatientSerNu	DiagnosisId	DiagnosisCreat	DiagnosisCo	Description
384	363	22989	2014-02-01	C50.9	Ca breast, unspecified
385	364	23334	2014-04-09	D05.9	Carcinoma in situ of breast, unspecified
386	365	22978	2014-02-10	C50.9	Ca breast, unspecified
387	366	23386	2014-05-09	D05.9	Carcinoma in situ of breast, unspecified
388	367	23280	2014-01-01	C50.4	Ca upper-outer quadrant of breast
389	368	21957	2013-11-21	C50.9	Ca breast, unspecified
390	369	23261	2014-06-05	C50.9	Ca breast, unspecified
391	370	21155	2013-08-09	C50.2	Ca upper-inner quadrant of breast
392	371	23269	2014-05-16	D05.9	Carcinoma in situ of breast, unspecified
393	372	21661	2013-11-19	C50.9	Ca breast, unspecified
394	373	21526	2013-11-04	C50.9	Ca breast, unspecified
395	374	23235	2014-06-04	C50.9	Ca breast, unspecified
396	375	23265	2013-12-24	C50.9	Ca breast, unspecified
397	376	21129	2013-07-04	C84.5	Other and unspecified T-cell lymphomas
398	377	16354	2009-03-24	C85.9	Non-Hodgkin's lymphoma, unspecified type
399	378	22052	2011-01-28	C50.4	Ca upper-outer quadrant of breast
400	379	18676	2011-11-09	C85.9	Non-Hodgkin's lymphoma, unspecified type
401	380	20332	2012-10-30	C85.9	Non-Hodgkin's lymphoma, unspecified type
402	381	20999	2013-09-16	C61	Ca prostate
403	382	21871	2013-12-19	C50.9	Ca breast, unspecified
404	383	21926	2014-01-31	C85.9	Non-Hodgkin's lymphoma, unspecified type

How to design OncoTime?

- Want to support an abstraction of time series of events, because this corresponds to our conceptual model of the process.
- Want a way of easily creating groups of related values.
- Need to get the correct data of out the MySQL database, but we do not want to require the user to build an SQL query.
- Once the data is expressed as a time series, we want some control structures to compute with those time series.
- Want to be able to interface to the host generated language (python?) in order to use existing libraries and to easily program specialized functions.

Program Structure

```
script Foo() //should reside in file Foo.onc, should first letter be a cap
/**
Documentation comment. This comment should be returned by the command "hel
foo"
*/

// ------- USE CLAUSES ---------------------------

// ------- GROUP DEFINITIONS ---------------------

// ------- FILTERS -------------------------------

// ------- COMPUTATIONS --------------------------
{}
```

- Header, documentation and computation block required, others optional.
- Sections must come in this order.

```
script FilterTest()
/**
 * This file demos filters ...
 */
// The filter is applied over the whole database, resulting in the data
// which will be considered in the computations part

population is
       Id: 1000, 2000, 3000
       Sex: M
       Birthyear: 1967
       PostalCode: K2G6K8
       Diagnosis: breast
doctors are
      Id: 2123, 1231, 1415
period is
      Dates: 2013-01-01, 2015-01-01
events are
      Events: ct_sim_booked
{ foreach Patient p
   print p
}
```

Any order and repetition allowed?

- the four sections, population, doctors, period and events may come in any order, and may be omitted
- if a section is omitted, then the '*' value is implied for all the fields (i.e. no filtering)
- if a section is repeated, then any field that is redefined becomes the active filter.
- inside the population and period sections the fields can be listed in any order
- if a field is omitted, then the '*' value is implied for all those fields
- if a field is repeated, then the last definition is the one taken (should redefinition be a warning?)

What does a group file look like? - myGroups.grp

```
/**
 * This file defines a set of data that can easily be changed to
 * make it specific to particular queries or doctor's interests.
 */

group Id myId = {1001, 2000}
group Id extraId = {400, 500}
group Id myPatientIds = {100, 200, 300, 500, <extraId>}
group Sex myGenders = {M, F}
group Birthyear myBirthyears = {1993, 2014, 1960}
group Diagnosis myDiagnosis = {breast, prostate, liver}
group Days myWorkDays = {Mon, Tues, Wed, Thurs, Fri}
group Years my_years = {2014, 2015, 2016}
```

Other types are: Postalcode, Date, Hour, Event

You can include some predefined groups, and also define some more

```
// --- uses
use myGroups.grp, ExtraGroups.grp

// ------- GROUP DEFINITIONS --------------------
// define a new group - string by default, other group names with <>
group Diagnosis diaglist = {breast, prostate}

// here is an example of using an existing group
group Diagnosis diaglist2 = {<diaglist>, liver}

// should you allow for some regular expressions for group values?
```

Database views to Event Streams

- The use, groups and filter sections set up all the information needed to determine what information needs to be extracted from the database.
- The compiler will generate SQL queries that will result in the information needed to generate an internal representation of an event stream, and it will create an efficient representation of the event trace.
- What is a reasonable representation? Want to not use excess space, but also want links for quickly traversing all events, all events for a patient, and perhaps other specializations.
- The computation section of the program computes over the event trace, and also assumes a table of patient info and doctor info for all patients and doctors that occur in the events.

What are events? Before treatment planning group:

```
consult_referral_received(time, patientid)
initial_consult_booked(time, patientid, doctorid) ... room?
initial_consult_completed(time, patientid, doctorid)
CT_sim_booked(time, patientid, doctorid) ... room?
ready_for_CT_sim(time, patientid, doctorid)
```

Treatment planning group:

```
CT_sim_completed(time, patientID, doctorID)
ready_for_initial_contour(time, patientID, doctorID) ... dosometristid?
ready_for_MD_contour(time, patientID, doctorID)
ready_for_dose_calculation(time, patiendID, doctorID)
prescription_approved_by_MD(time, patientID, doctorID)
ready_for_physics_QA(time, patientID, doctorID) ... physicistid?
ready_for_treatment(time, patientID, doctorID)
machine_rooms_booked(time, patientID, doctorID) ... machines?
patient_contacted(time, patientID, doctorID)
```

vikramPatientData.grp

```
/**
 * Contains all the information I might want to use across OncoTime progra
 */

group Id patientGroupOne = {1 to 250, 300 to 400, 1001}
group Birthyear patientBirthyearRange = {1950 to 1970}
```

vikramPatientData.grp

```
script DailyPatientHistory()
/**
 * Generates my patient Timelines.
 */

// ---- Change data in this group file ----
use vikramPatientData.grp

// ---- Filters ----
population is
      Id: <patientGroupOne>
      Birthyear: <patientBirthyearRange>
      Sex: M, F

// ---- Computations ----
{ foreach Patient p
    print p
}
```

Example 2 from Reference Compiler

```
script Barcharts()
/**
 * Generates barcharts.
 */

// ---- Change data in this group file ----
use vikramPatientData.grp

// ---- Filters ----
population is
      Id: <patientGroupOne>
      Birthyear: <patientBirthyearRange>
      Sex: M, F

{
      table t = count Patients by Birthyear
      table s = count Patients by Diagnosis
      table v = count Doctors by Id

      print t
      print s
      print v
}
```

Example 3 from Reference Compiler

```
script PatientDoctorDiagnosis()
/**
 * Prints all the informations for male patients who came in between
 * 2010 and 2015.
 * All months except for December.
 */

group Months myWorkMonths = {01, 02, 03 to 10}

population is
      Sex: M
      Id: 1 to 2500
      Birthyear: 1950 to 1970

period is
      Months: <myWorkMonths>, 11
      Years : 2010 to 2015

{ foreach Patient p
    { print p
      foreach Doctor d
        { print d
          foreach Diagnosis diag
            { print diag
```

```
                }
            }
        }
    }
```

Example 4 from Reference Compiler

```
script PatientSequences()
/**
 * Generates my patient Timelines.
 */

// ---- Change data in this group file ----
use vikramPatientData.grp

// ---- Filters ----
population is
       Id: <patientGroupOne>
       Birthyear: <patientBirthyearRange>
       Sex: M, F

{ list s = sequences like
             [ct_sim_booked ->
              ct_sim_completed | patient_arrives ->
              end]

  foreach member i in s
   print i
}
```

Example 5 from Reference Compiler

```
script ParameterExpansion(Sex s, Birthyear b1, Birthyear b2, Events e)
/**
 * Parameter example.
 */

group Birthyear myBirthyears = {<b1>, 1993, 2012, <b2>}
population is
      Id: 13456, 134, 2455
      Sex: M, F
      Birthyear: <myBirthyears>
      Diagnosis: breast, prostate, <e> // can be strings or groups
      PostalCode: H4X
period is
      Years: 2012, 2012, 2014
      Months: 01, 02, 03
      Days: Mon, Tues, Wed, Thurs, Fri, Sat, Sun
events are
      Events: <e>

{ foreach Doctor d
   print d
}
```

Some possible computations - may not match reference compiler

```
foreach patient p
 print p

// optional where clause ??
foreach patient p // [where gender is female, postalcode is K]
 print timeline of p // all events, ordered by time, for a patient
                     // can support a variety of nice visualizations

foreach diagnosis d // [where diagnosis is breast, prostate]
 { print d
   foreach patient p
    print ID, Gender, Age, Diagnosis of p // field names with caps
 }

foreach doctor d
 print timeline of d
```

Tables - basically association tables between keys and values

```
table x = count patients by Diagnosis // table of pairs (Diagnosis : n)

// What are good high-level operators to provide for tables?

foreach element i of x
 print x[i] // prints pairs

print x.length

// plot
barchart x // [to filename]?
```

Sequences - one of the key abstractions

```
// each p1 is either an identifier or a group
  foreach sequence s like []
    // [patient_arrives -> patient_seen]
    // [patient_arrives(p) -> patient_seen(p)]
    // [e1 -> e2 | e3 -> e4]
    // [e1 -> {e2, e3} -> e4]
    // [e1 -> {e2, e3}* -> e4]
    // [e1 -(not e2)-> end]
   print s

  // lists are lists of sequences
  list S = sequences like []
  foreach member s in S
   print s
```

Using the host language to compute

```
// call out to native code
table z = native(func,x,y) // where x and y are tables, sequences, or
                           // lists of sequences
sequence z = native(func,x,y)
list l = native(func,x,y)
```

- Need to define an interface between your internal representation and the representation expected by the native call.
- Perhaps easier if you generate code in the same language as the native code (Python?)

What are the challenges?

- Write the grammar. Maybe leave quite a bit of checking to a weeding phase, which is aided by an external structure.
- Determine what static checks including type checks are required.
- Design the code generation for the MySQL queries.
- Design the events representation, and store results of queries in that representation.
- Generate the code generation for computation code as computations over the events.
- You can simplify the language if it gets too complex, and we can add new language features if things go easily. We will have a core set of features that you must implement.