Biological Data Analysis using InterMine

Rachel Lyne
Julie Sullivan
Josh Heimbach

October 27th 2017

Workshop Schedule

- Introduction
- Walk through interface with hands-on exercises
- Analysis workflows
- Web services
- Questions and detailed use-case

www.flymine.org
www.humanmine.org

What is InterMine?

- Data Warehouse
- Open Source
- Integration of Biological Data
- Gos Micklem; Dept of Genetics

What is InterMine?

- Webapp – user interface
  - Data browsing and exploration
  - Analysis tools
  - Advanced search interfaces
- API
  - Perl, Python……JSON, XML

Who Uses InterMine?

EXISTING MINES

A number of different data warehouses powered by InterMine already exist. These include:

FlyMine - *Drosophila* genomics
modMine - fly and worm modENCODE data
MouseMine - at MGI
RatMine - at RGD
WormMine - at WormBase
YeastMine - at SGD
ZebrafishMine - at ZFIN
INDIGOmine - microbes
ThaleMine - Araport Project with data for Arabidopsis thaliana
ChickpeaMine - Desi & Kabul chickpea
TargetMine - drug target discovery
MitoMiner - proteomic data for mitochondria
HumanMine - human
FlyTF.org - *Drosophila* transcription factors
PhytoMine - plants
MedicMine - *Medicago truncatula*
BovineMine - *Bos Taurus*
HymenopteraMine - Bees, Ants & Wasps
SoyMine - Soybase soy bean data
CHOMine - *Cricetulus griseus* and CHO cells
BeanMine - LegFed chado bean data
LegumeMine - String bean, Soy, and Peanut
PeanutMine - Peanut chado/GFF data
Shaare - Gene candidate prioritisation
PlanMine - Planarian flatworms
Wheat3BMine - Wheat chromosome 3B
GrapeMine - Grapevine
RepetDB - repetitive DNA elements
XenMine - Xenopus
TetraMine - *Tetrahymena thermophila*

Data

GENES

INTERACTIONS

ALLELES

PROTEINS

FUNCTION

SEQUENCE VARIANTS

PROTEIN DOMAINS

EXPRESSION

GENE ONTOLOGY

PHENOTYPES

DISEASE

PATHWAYS

ORTHOLOGUES

GWAS

REGULATION

Why use InterMine?

- Query across several data sources at once
- No need to deal with several data formats
- Identifier resolution system
- Collate information about items and sets
- Common platform to many organisms and type of data
- Information without visiting several sites

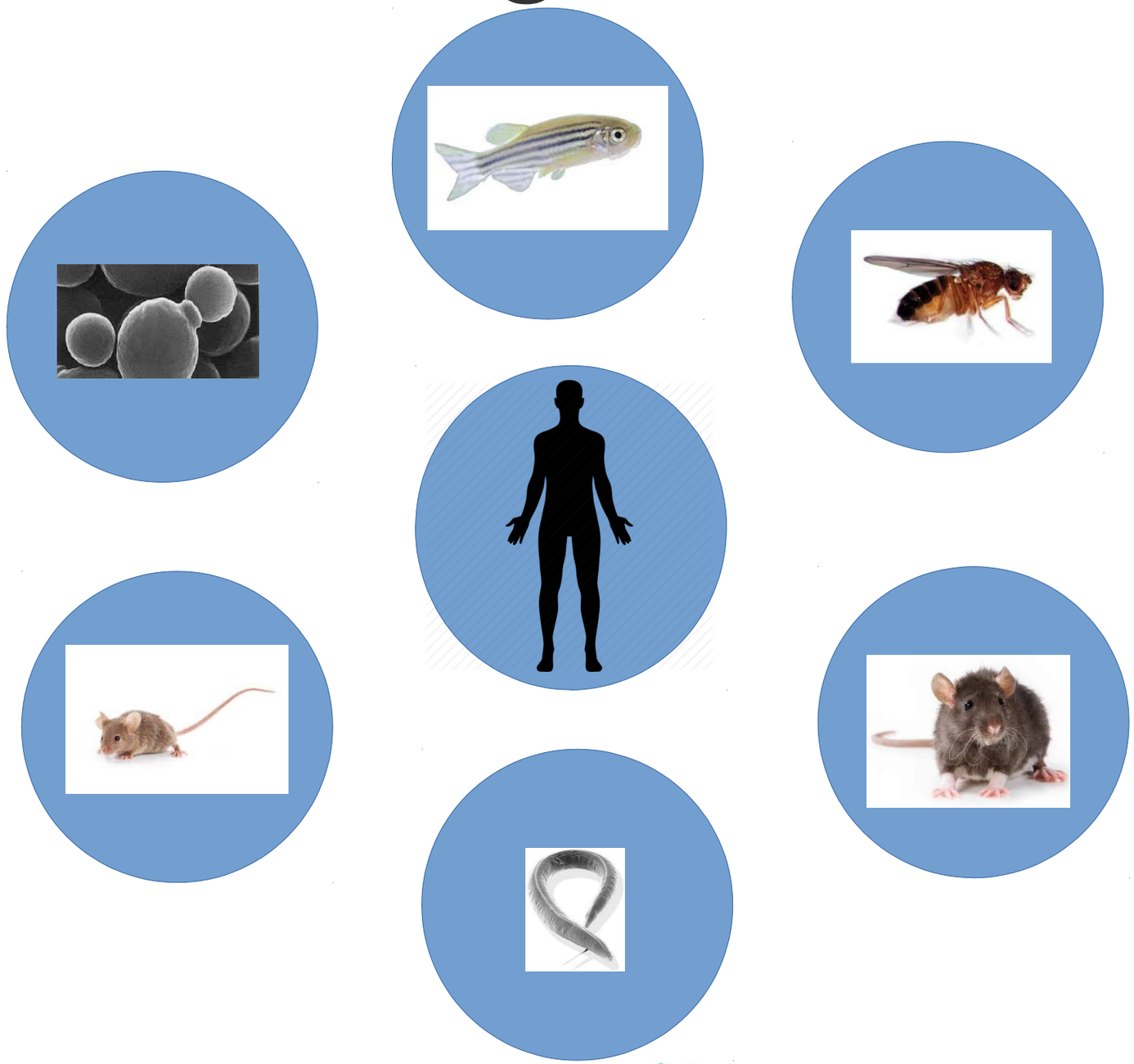
InterMine Data Integration

Your own InterMine:

- Sophisticated build system
- Set integration keys
  (define equivalence between datasets)
- ID resolution system
  (old datasets, different identifier systems)
- Core library of data loaders
- Custom data loaders - own data

Tutorial: www.intermine.org

Model Organism Data



MGI Mouse Genome Informatics

RGD

WormBase

ZFIN

FlyBase

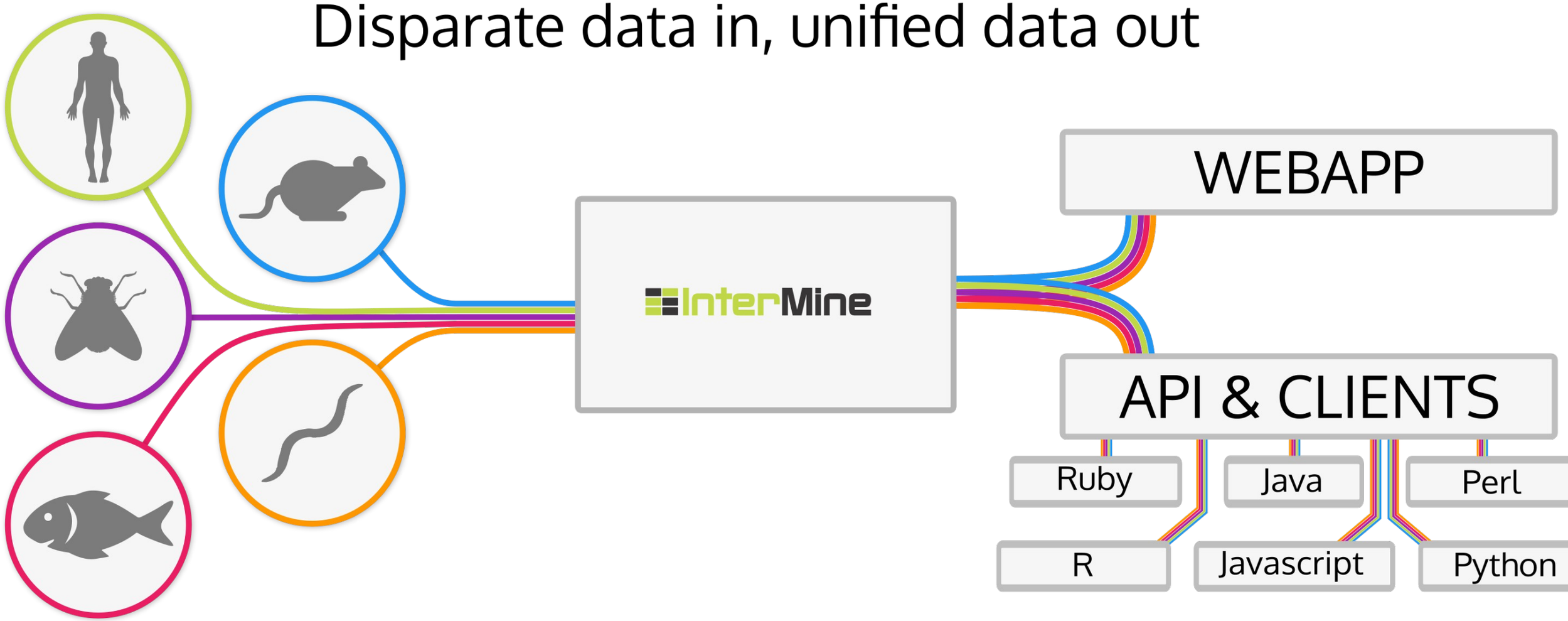
SGD

UNIVERSITY OF CAMBRIDGE

InterMine

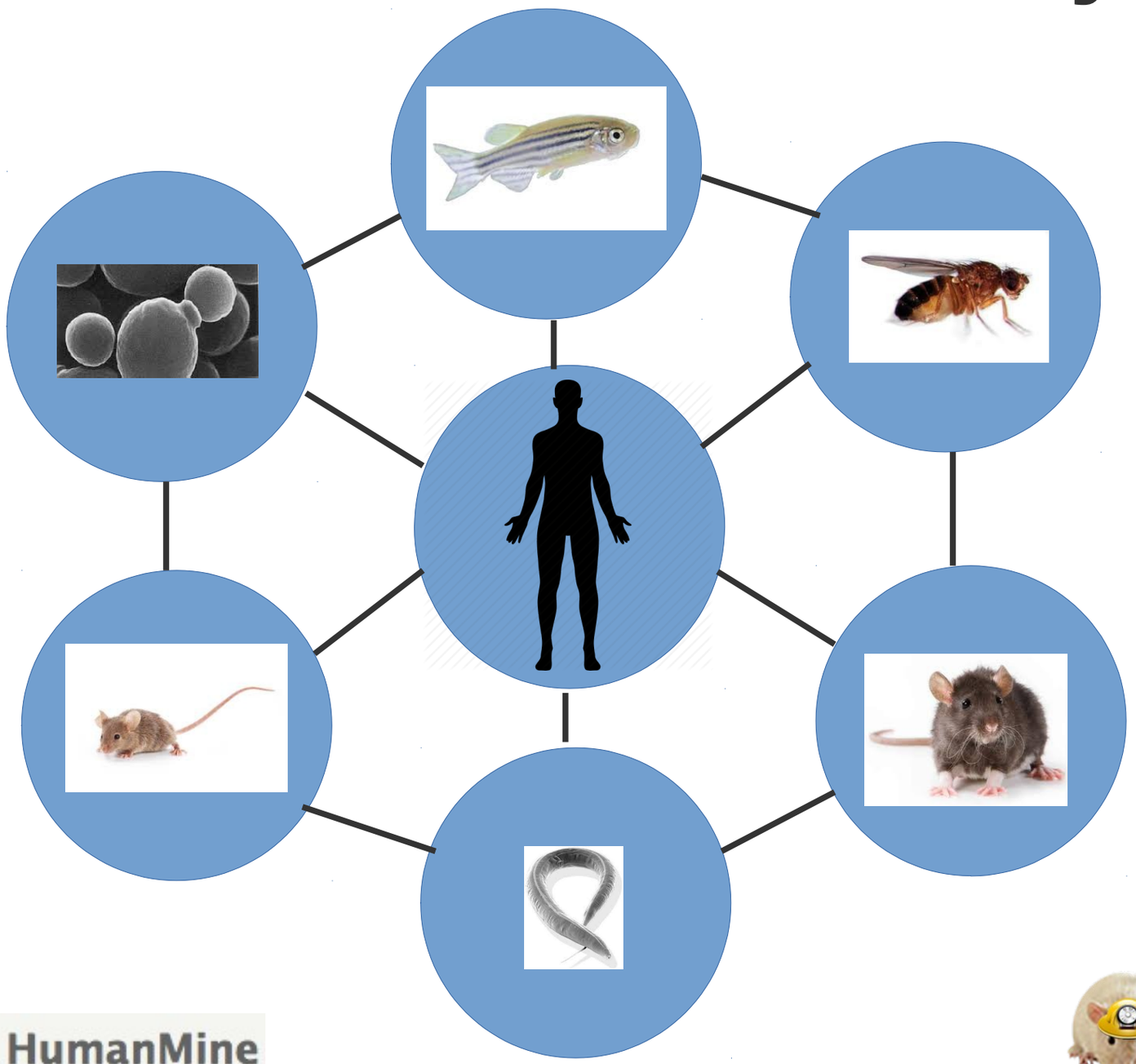
The InterMOD Project

Disparate data in, unified data out



Model organism images Designed by Freepik and distributed by Flaticon

The InterMOD Project



HumanMine

RatMine

ZebrafishMine

SGD YeastMine

MouseMine

WormMine

FlyMine

UNIVERSITY OF CAMBRIDGE

InterMine

InterMine Accounts

InterMine is free to use without creating an account.

Creating an account allows you to save lists and searches permanently and share lists with your colleagues.

At the moment you have to make a separate account for Each InterMine database

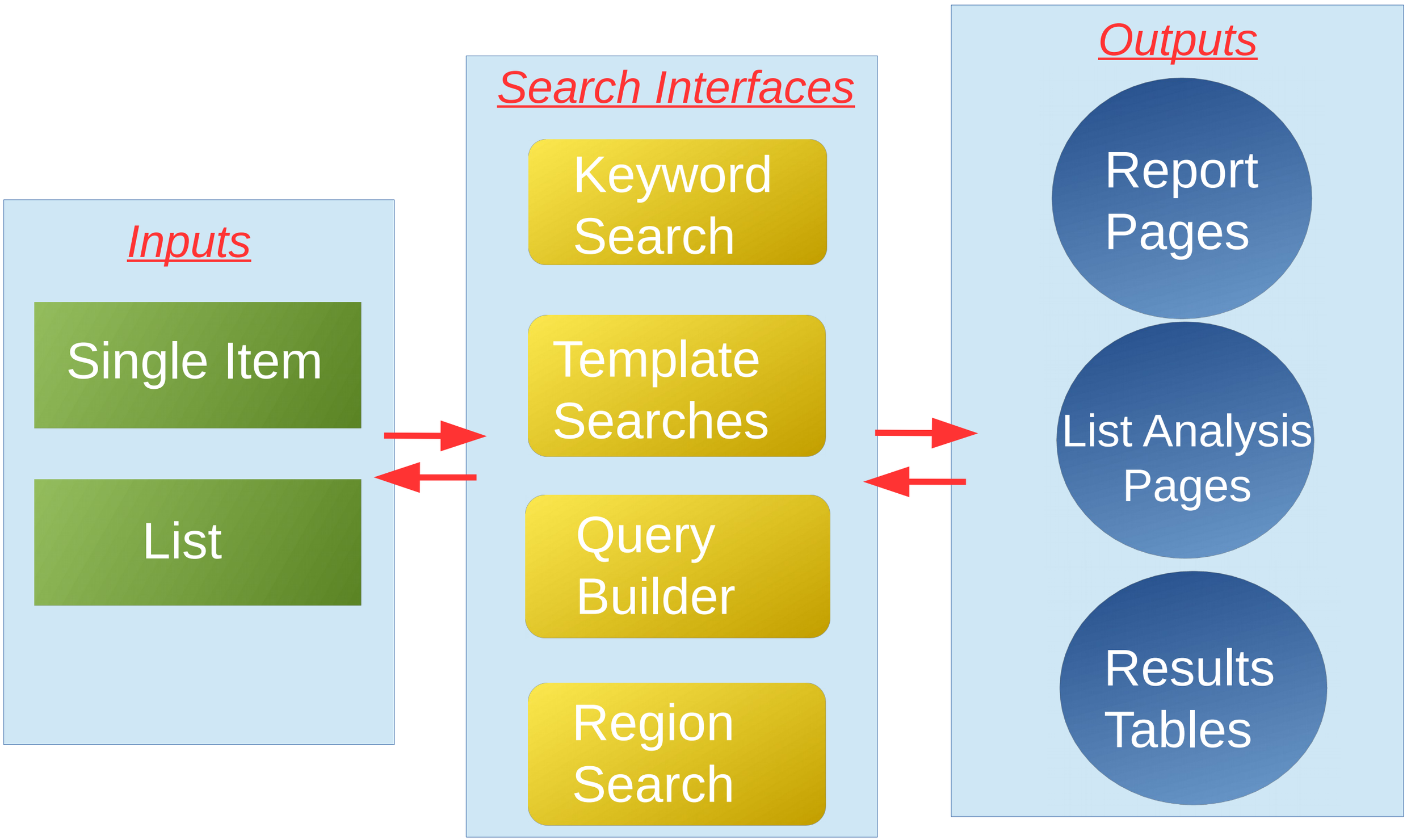
Using InterMine

Search

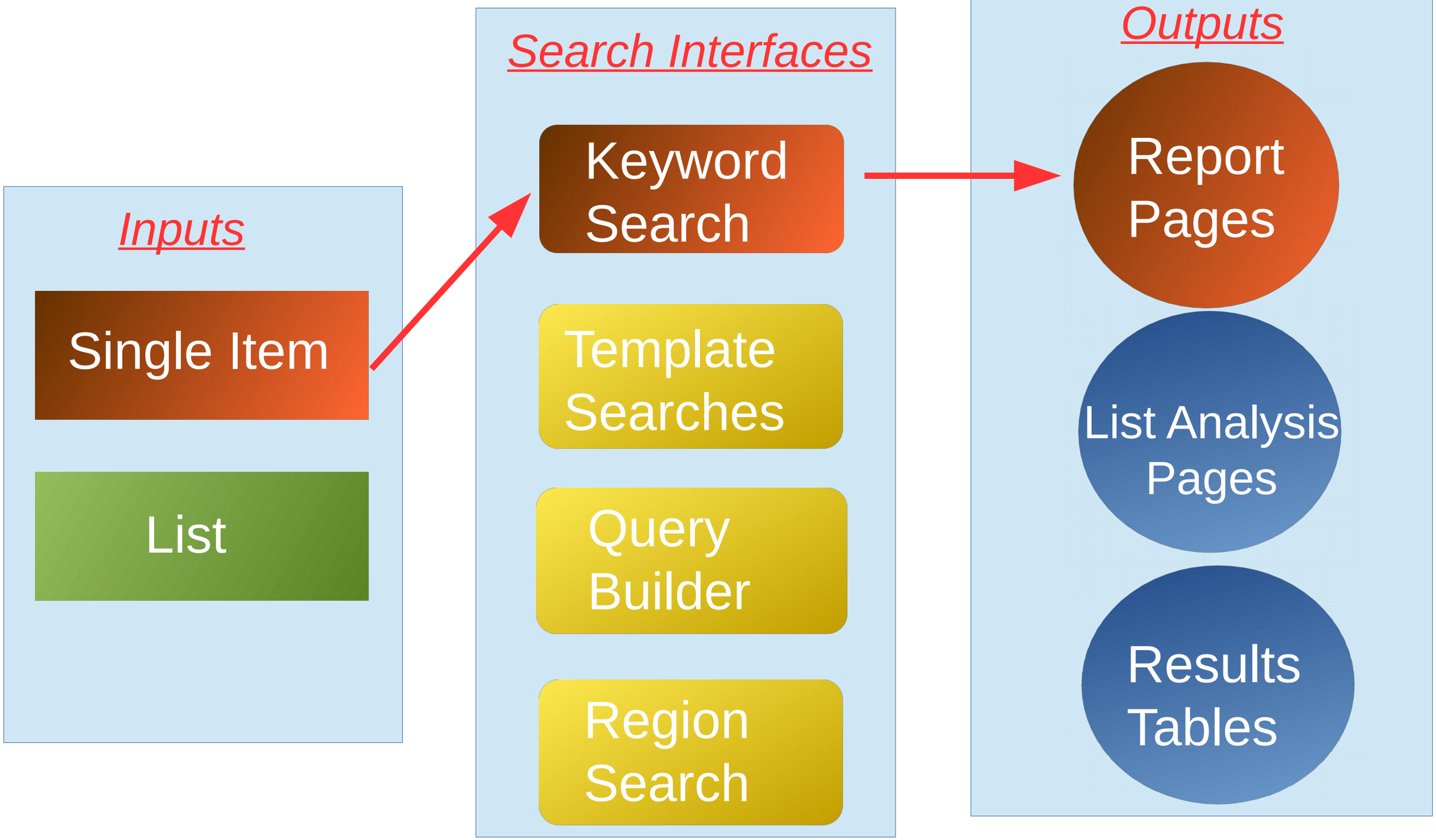
Explore

Analyse

The Web Interface



The Web Interface



Search and Explore

Faceted Keyword Search:

-Specific e.g. pax6, pparg
→ Report pages

-Exploratory e.g. Insulin, Diabetes
→ Report pages
→ Lists

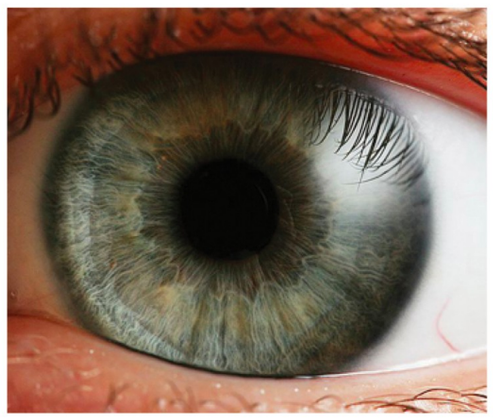
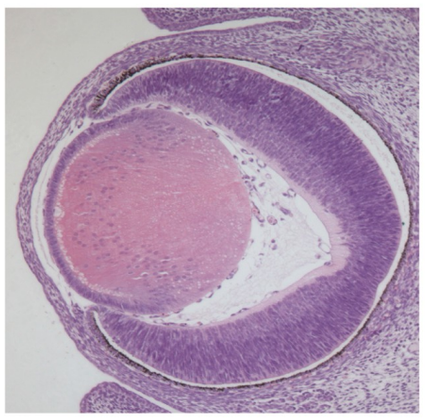
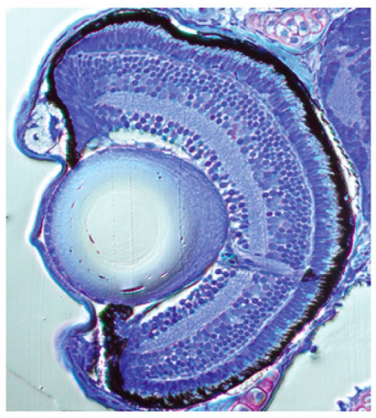
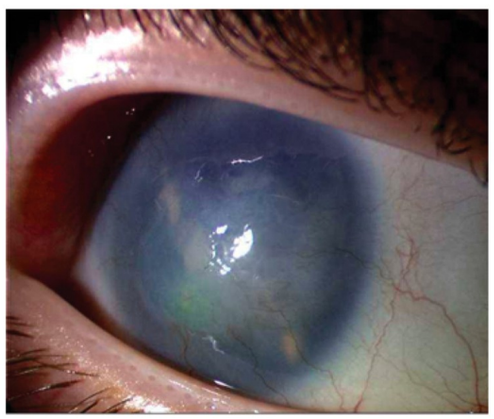
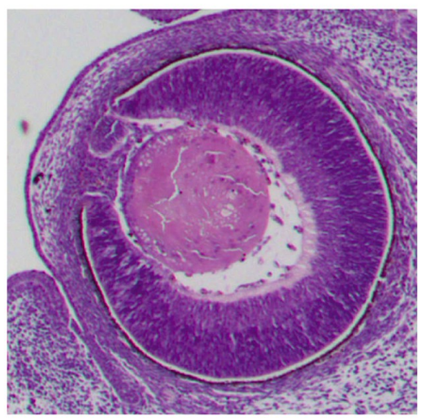
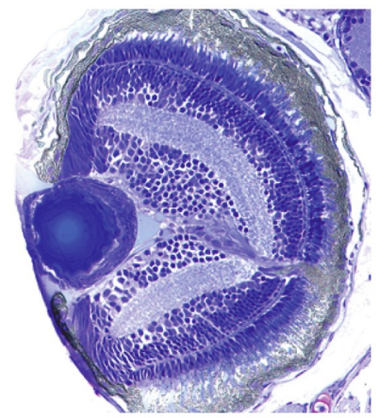
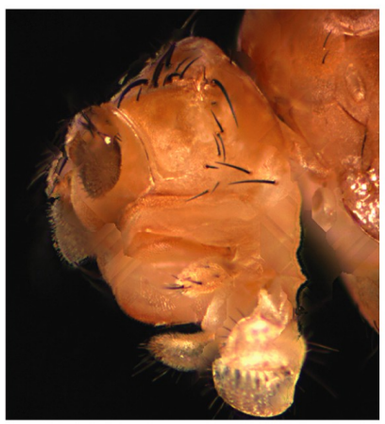
-Filter results by data type

Data Exploration: report pages

Every object (item) in InterMine has a report page

- Report pages collate all the data available for that object
- Report pages contain a mixture of tables, search results and graphical displays.
- The tables and graphical displays are interactive
- You can explore data by navigating through links to report pages for related objects.
- Report pages are accessed from keyword searches and results tables.
- Report pages for different InterMines can be configured differently

Explore: Pax6

	Human	Mouse	Zebrafish	*Drosophila*
WT				
mut				
	***PAX6*$^{+/-}$**	***Pax6*$^{-/-}$**	***pax6b*$^{-/-}$**	***ey*$^{-/-}$**
EQs	cornea opaque iris absent retina degenerate lens opaque aqueous humor of eyeball increased pressure	eye decreased size lens fused_to cornea iris morphology anterior chamber absent	eye decreased size lens decreased size retina malformed	eye absent

Washington NL, Haendel MA, Mungall CJ, Ashburner M, Westerfield M, Lewis SE. - Figure 1 of Washington et al.:
"Linking Human Diseases to Animal Models Using Ontology-Based Phenotype Annotation."Â PLoS BiolÂ 7(11): e1000247. doi:10.1371/journal.pbio.1000247

Exercise 1: Faceted Search

1. Search for one or more of the following in HumanMine:

- *PPARG*
- rs10509540
- *diabetes*

2. Filter and create a list:

- Search for *diabetes*
- Filter for publications
- Make a list of these publications

Note: If you filter for genes note that this is not a comprehensive search for diabetes genes as it does not check functional annotations.

Exercise2: Exploring a Gene

You are interested in the Human *PPARG* gene and want to know the following things about it:

1. On which chromosome is *PPARG* located?

2. Can I access the sequence for the *PPARG* gene?

3. With which diseases is *PPARG* associated?

4. In which tissues is *PPARG* most highly expressed?

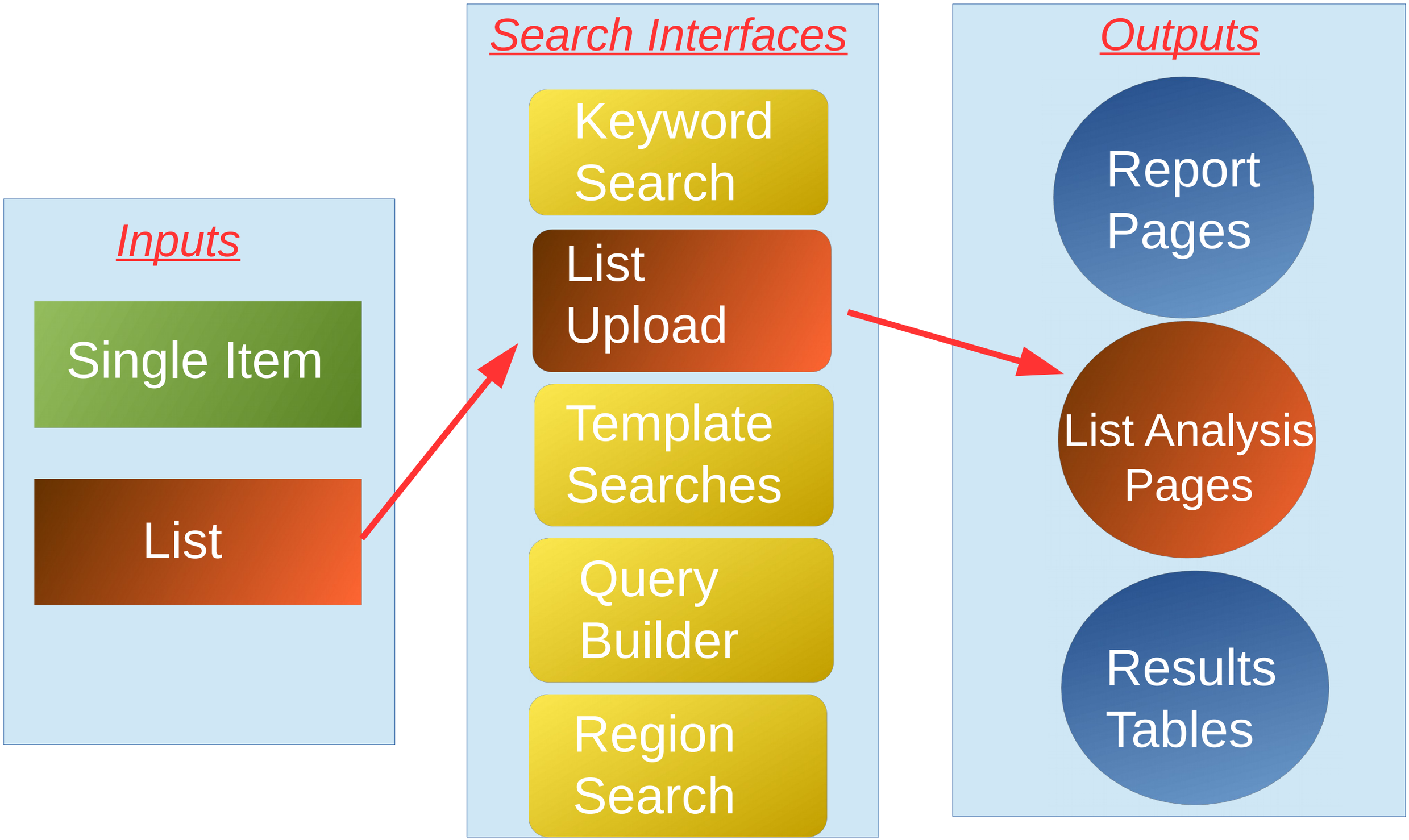
5. Does the *PPARG* protein have any know isoforms?

6. Is there a *PPARG* orthologue in *D. melanogaster*?

7. Does this orthologue interact with any other Genes/proteins? Identify the interaction type (genetic/physical)

8. For the interaction with CG3040, what was the original experiment and publication that determined this interaction

The Web Interface



List Analysis: Uploading a list

- Upload your own lists to InterMine
- Powerful identifier resolution system
- Convert old identifiers into an up-to-date set

Exercise 3:Uploading a list:

1. Use FlyMine: navigate to the lists tab and list upload sub-tab

2. Select the example list (leave type and organism as the default values).

3. Click “Create list”.

4. Examine and understand the list page, name and save your list.

Exercise 3:Uploading a list:

- E2f has matched two genes (**duplicates**) - in this case you need to decide which of the two genes you want in your list (or both). The action column allows you to do this.

- Two of the identifiers in the list matched the same gene: FBgn0010433 and ato. This is indicated in the **direct hits**.

- One of the identifiers is a protein identifier (TWIST_DROME). As the associated gene could be identified, this has been added to the list. This is shown under **non-gene identifiers**.

- Two of the identifiers matched a **synonym** (rather than a current identifier). As the synonyms matched only one gene, these are automatically added to the list.

Data Exploration. Lists

InterMine allows you to explore data for a whole list of objects

- Form a basic “unit” of analysis
- Can be uploaded or created from searches
- Identifier resolution
- All lists have a list analysis page
- Provide summary table, widgets and search results
- List set operations - union, intersect, subtract
- Allow workflows through iterative querying and set operations
- Public lists also available

List Analysis

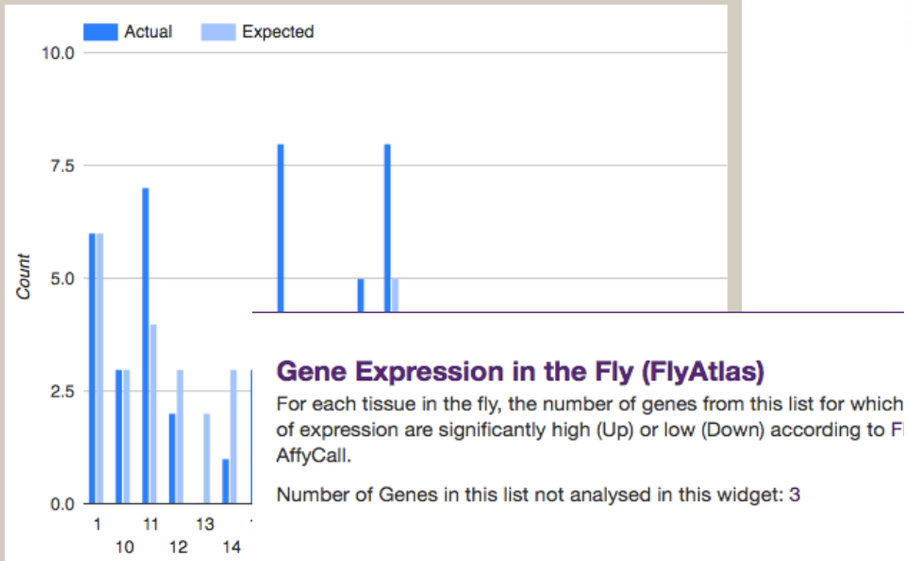
Chromosome Distribution

Actual: number of items in this list found on each chromosome. Expected: given the total number of items on the chromosome and the number of items in this list, the number of items expected to be found on each chromosome.

All items in your list have been analysed.

Organism

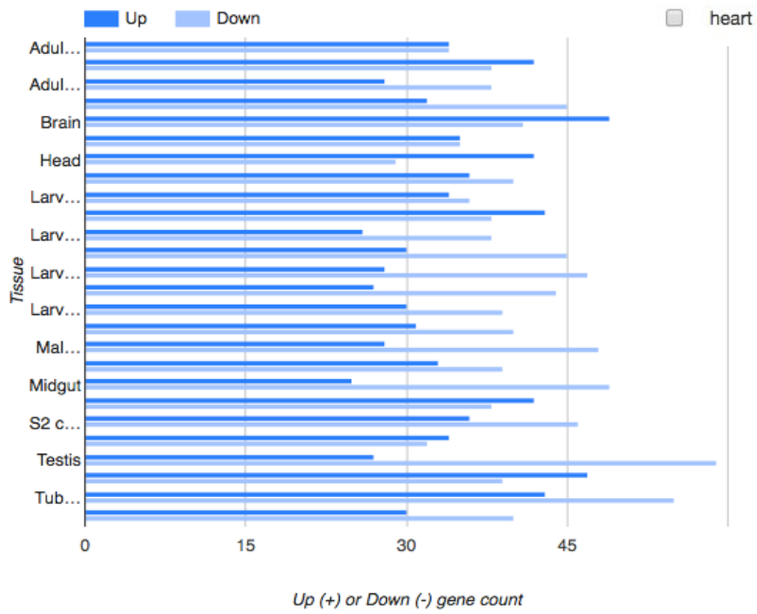
Homo sapiens



Gene Expression in the Fly (FlyAtlas)

For each tissue in the fly, the number of genes from this list for which of expression are significantly high (Up) or low (Down) according to F AffyCall.

Number of Genes in this list not analysed in this widget: 3



Gene Ontology Enrichment

GO terms enriched for items in this list.

Number of Genes in this list not analysed in this widget: 17

Test Correction: Holm-Bonferroni | Max p-value: 0.05 | Ontology: biological_process

Background population: Default [Change]

[View] [Download]

GO Term	p-Value	Matches
cell adhesion [GO:0007155]	8.077144e-5	13
biological adhesion [GO:0022610]		
Wnt signaling pathway [GO:0016055]		
cell-cell adhesion [GO:0098609]		
wing disc development [GO:0035220]		
cell-matrix adhesion [GO:0007160]		
tissue development [GO:0009888]		
heart development [GO:0007507]		

View homologues in other Mines:

- RatMine — R. norvegicus
- YeastMine — S. cerevisiae
- MouseMine — M. musculus
- HumanMine — H. sapiens
- ZebrafishMine — D. rerio

Mammalian Phenotype Ontology Enrichment

MP terms enriched for items in this list.

Number of Genes in this list not analysed in this widget: 72

Test Correction: Holm-Bonferroni | Max p-value: 0.05 | Background population: Default [Change]

[View] [Download]

MP Term	p-Value	Matches
abnormal DNA repair [MP:0008058]	1.831867e-66	52
increased sensitivity to induced cell death [MP:0008943]	1.383230e-62	59
abnormal induced cell death [MP:0008942]	1.175928e-60	65
abnormal chromosome stability [MP:0010094]	4.405108e-58	48
chromosomal instability [MP:0008866]	1.743731e-56	47
abnormal cell physiology [MP:0005621]	2.197659e-53	163
cellular phenotype [MP:0005384]	1.305734e-51	168
chromosome breakage [MP:0004028]	4.007892e-50	37
abnormal cell death [MP:0000313]	1.184952e-44	106

Exercise 4: List Analysis:

Examine the FlyMine public list: **PL FlyTF_putativeTFs (757 genes)**

1. In which tissues are these genes most highly expressed?

2. What is the most enriched GO term for this list?

3. How many genes in the list are annotated with this GO term?

4. Note: you could make a sub-list containing only genes from this list annotated with this term by clicking on the matches number

5. Navigate to the MouseMine database to examine the mouse orthologues for this list.

6. How many mouse orthologues are there for this list?

7. Are these mouse genes enriched for any phenotypes?

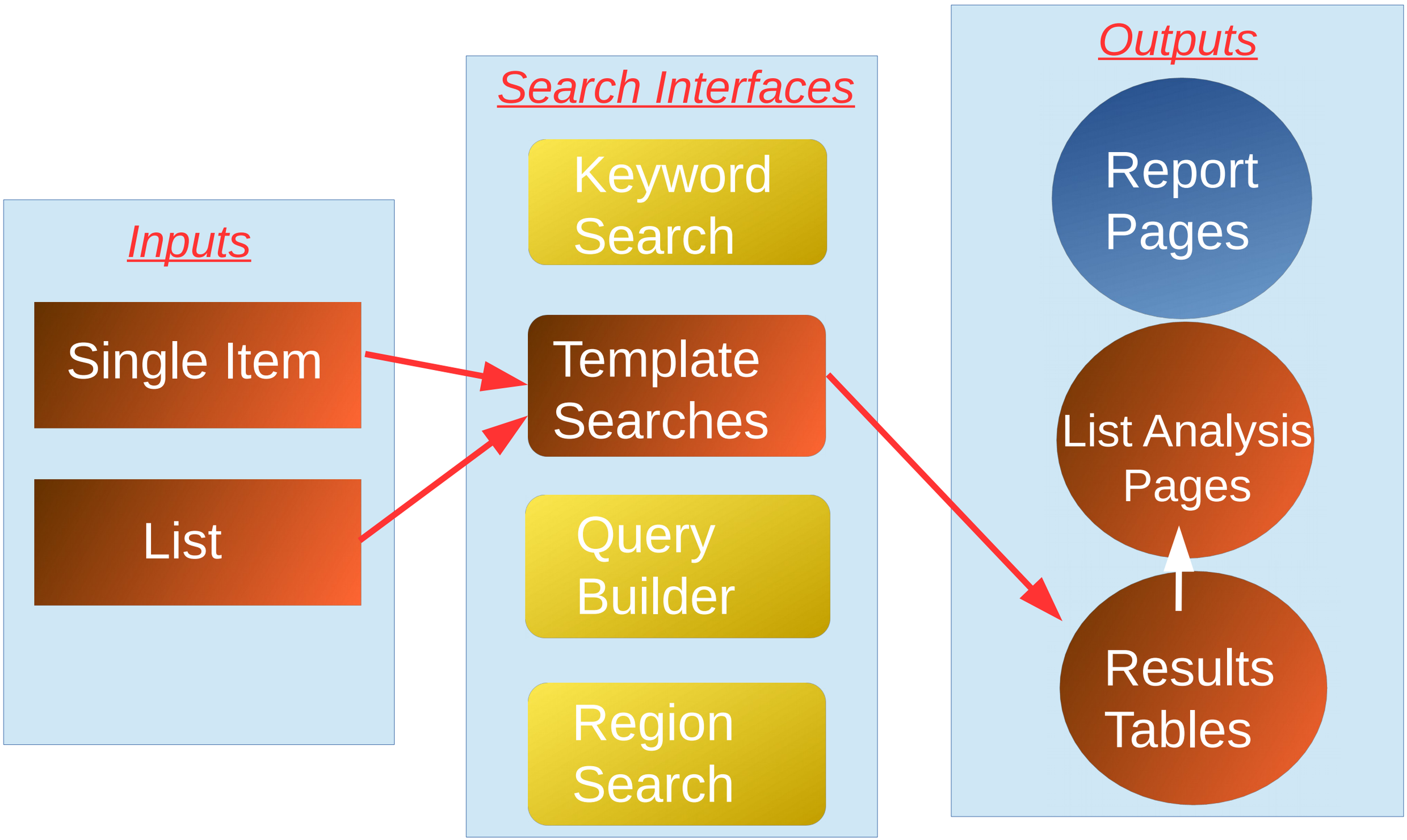
Using InterMine

Search

Explore

Analyse

The Web Interface



Data Analysis: Template Searches

- What other genes are involved in pancreatic function?
- Are there potential targets of pax6 in pancreatic tissue?
- Have these genes been implicated in pancreatic disease?
- What published data is there about these genes?

InterMine

Data Analysis: Template Searches

The template searches allow a more refined search than the keyword search and report pages but are still quick and easy to access.

- Pre-defined searches with simple filters
- Range from simple searches to more complex searches spanning several data types
- Run with single item or list
- Results are returned in sophisticated results tables
- Easy to add - just ask

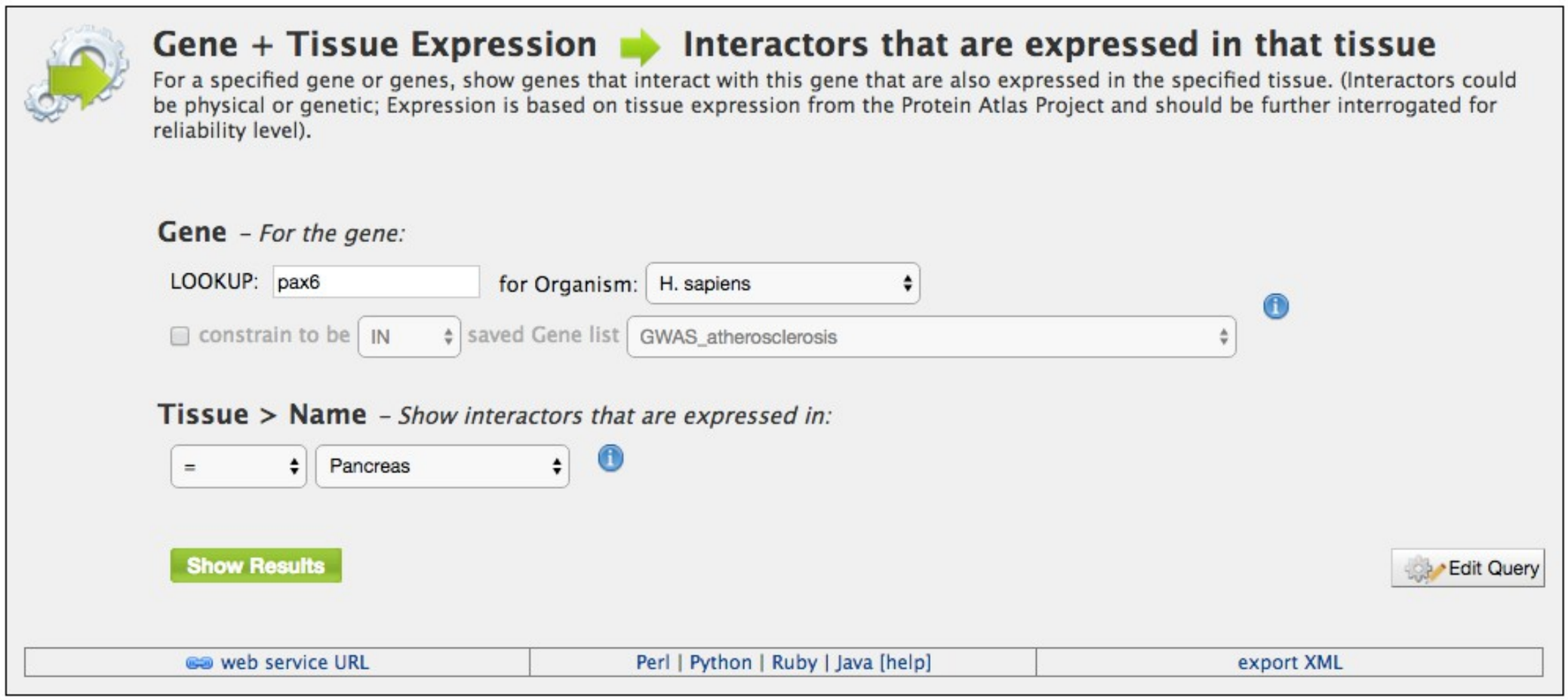
Many Many Searches.....

- Which other genes have this GO annotation?
- Are there mutant phenotypes for this gene?
- Where is this gene expressed?
- What does this gene interact with?
- Do any of the interacting genes share the mutant phenotypes?
- Does this gene have a human orthologue with a disease association?
- Have any variants been associated with this gene/disease?
- Which organisms have models for this disease/gene?

Data Analysis: Template Searches

Are there potential targets of pax6 in pancreatic tissue?

Gene + Tissue Expression → Interactors that are expressed in that tissue



InterMine

Data Analysis: Results Tables

Results tables allow further interactive analysis of the data through:

- Column summaries
- Column sorting
- Adding additional columns of data
- Filtering
- List creation
- Export

The power of column summaries

- Find the number of unique items in a column
- Find the number of items with a particular property
- Filter the table for one or more specific properties

Exercise 5: Template searches:

1. Browse the template searches in FlyMine and HumanMine - try running a few or changing the filters.

2. Use the search box to find template searches for interactions

3. Filter the FlyMine template searches to show only “expression” templates.

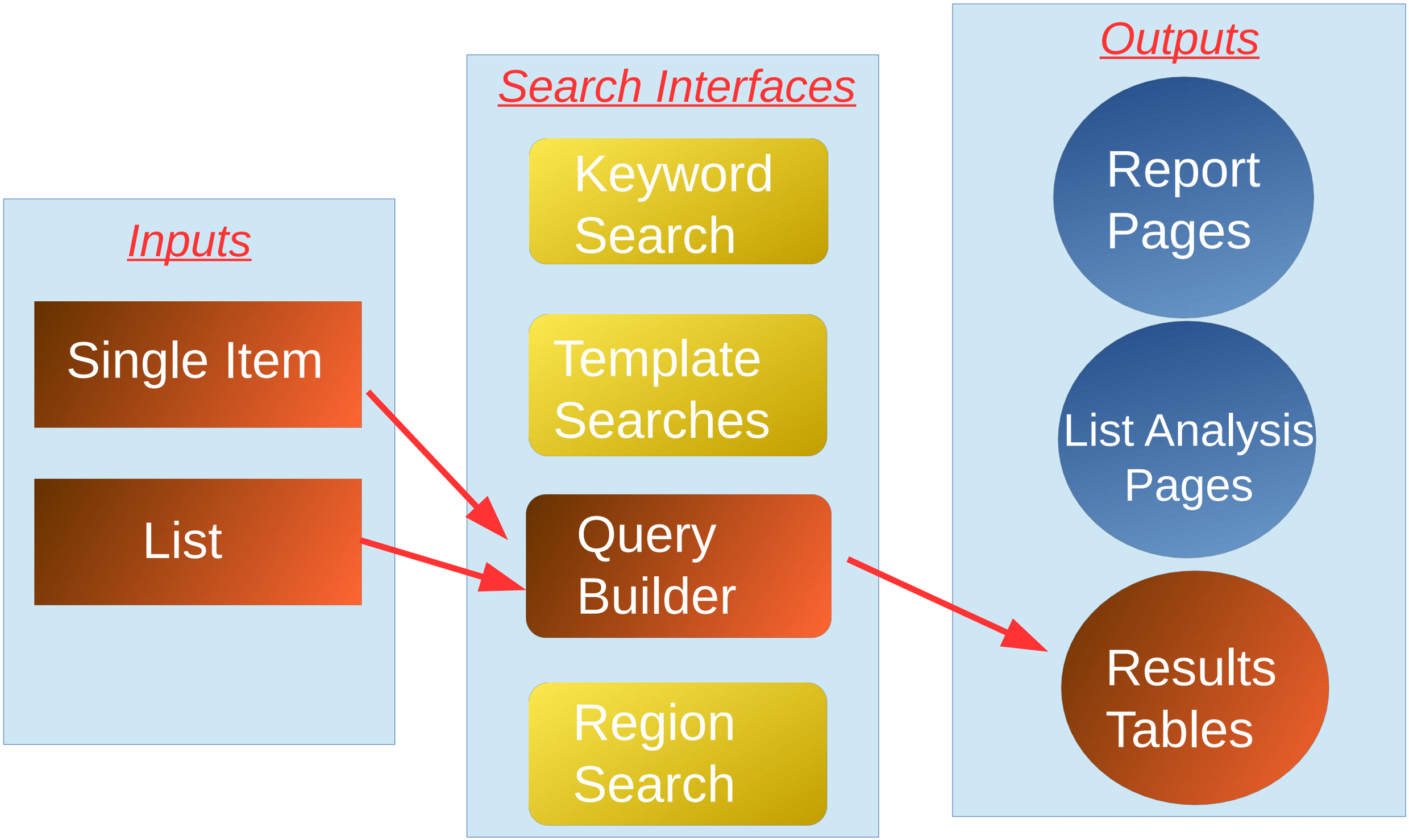
Exercise 6: Using template searches:

In our exploration of the PPARG gene we could see that it is Up-regulated in adipose tissue. Use the template searches in HumanMine to answer the following questions:

1. What other genes are up-regulated in adipose tissue according to the array express dataset? Don't forget to add a filter for "UP" expression. Save a list of these genes.

2. Do any of the genes identified in 1. interact with PPARG? Save a list of these genes.

The Web Interface



Advanced Search:Query Builder

The Query Builder is InterMine's custom query builder, allowing you to create and save your own searches.

- Build your own Searches
- Modify template searches
- Combine any data:
  - And, Or,
  - Intersect; Union

Data Analysis: Query Builder

Three steps to construct a query:

1. Navigate the data model to find the class or attribute you need
2. Add the appropriate constraint (filter) to the class/attribute
3. Decide on the columns you want to view in your results

Exercise 7: Query Builder:

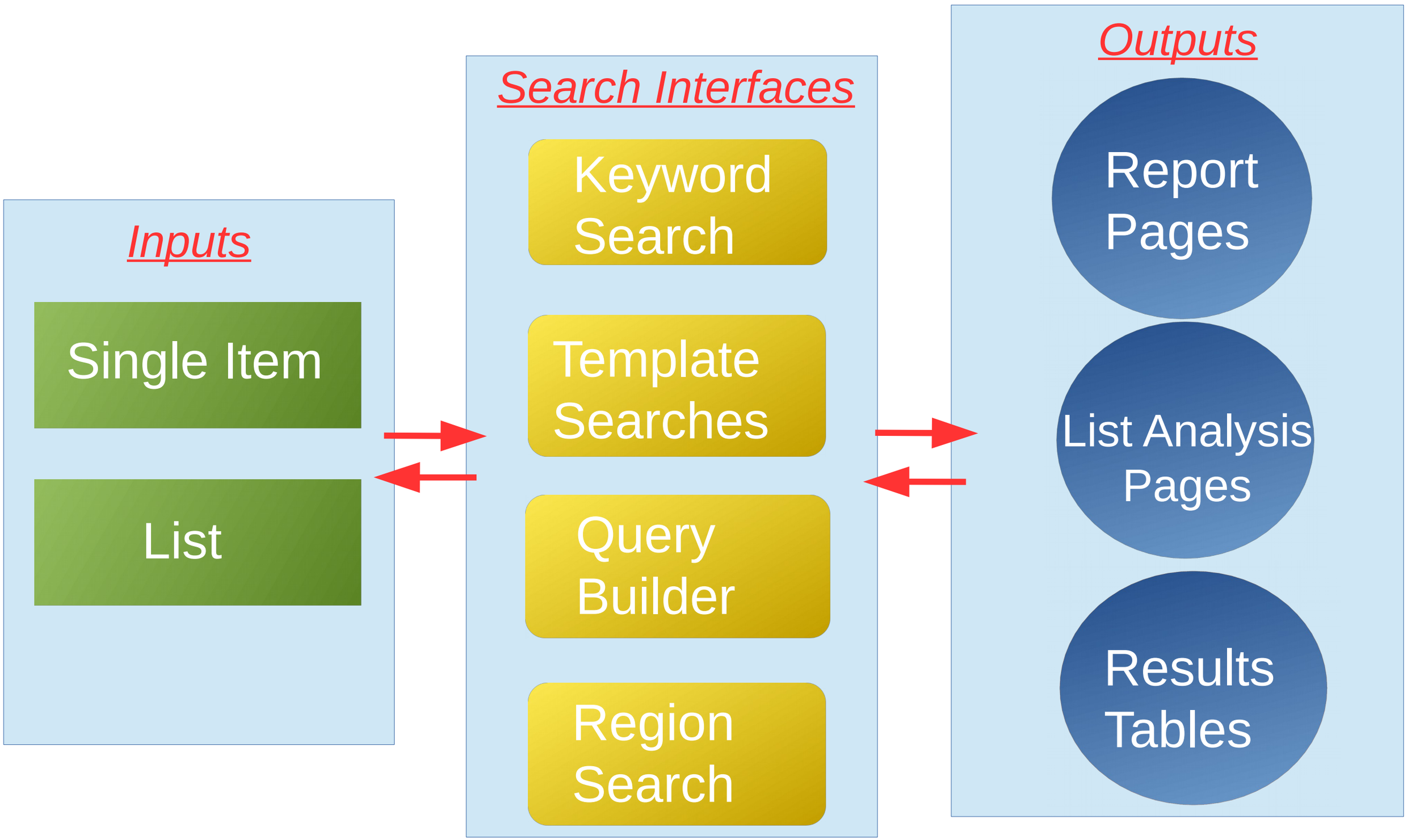
Using HumanMine: we will build a query to show Human genes and OMIM diseases, and then add a further constraint to show genes associated with all types of Diabetes.

1. Start your query from Gene
2. Constrain “Organism” to Homo Sapiens
3. Add the columns of data we want in our results:

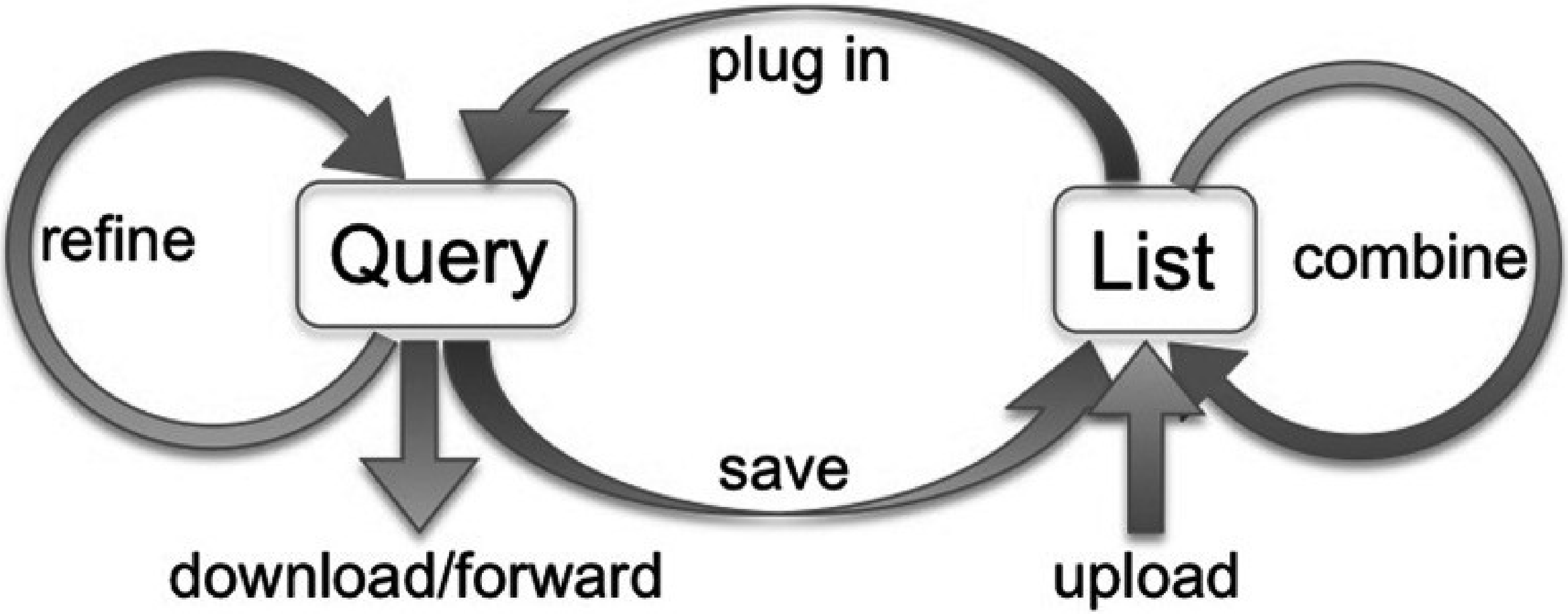
Gene: Primary identifier and Symbol
Disease: name

4. Run this search - ‘Show results’.
5. Return to the query (Use the “Trail” in the top left) and add a constraint to Disease name for “CONTAINS *Diabetes*”
6. Run the search and save the set of genes

The Web Interface

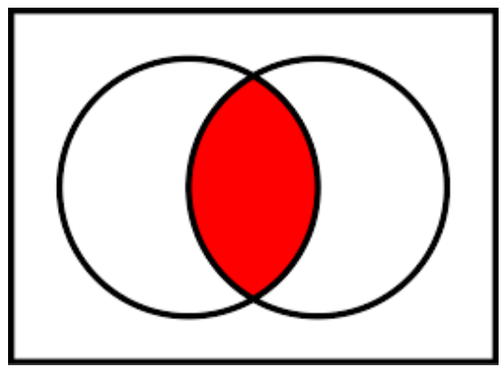


Analysis Workflows

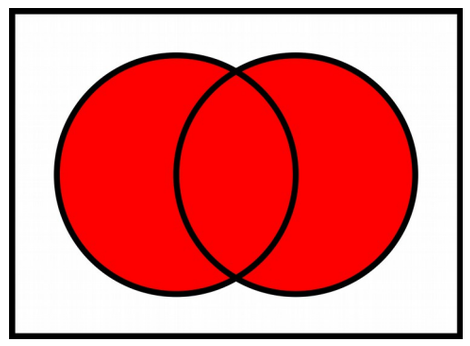


Motenko H, Neuhauser SB, O'Keefe M, Richardson JE. MouseMine: a new data warehouse for MGI. Mamm Genome. 2015 Aug;26(7-8):325-30. doi: 10.1007/s00335-015-9573-z. PubMed PMID: 26092688; PubMed Central PMCID: PMC4534495

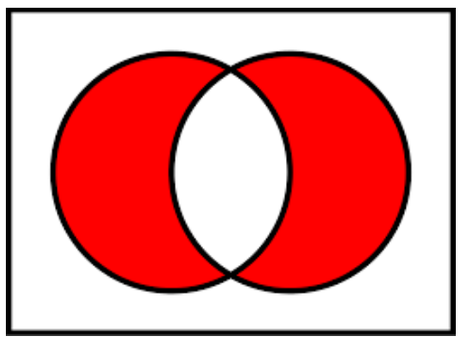
Combine Lists



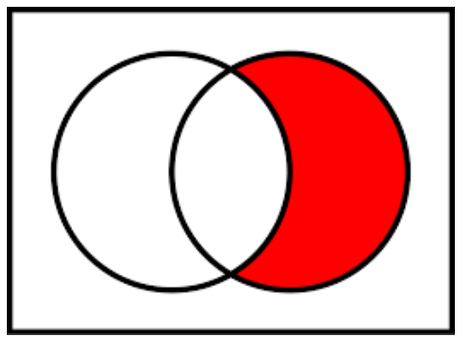
Intersect



Union



Subtraction



Asymmetric difference

Exercise 8: Analysis Workflows

In exercise 6 we ran a template search, saved a set of genes and fed this set back into another template. In this exercise we will first combine our final set of genes from exercise 6 with our set of diabetes genes from exercise 7.

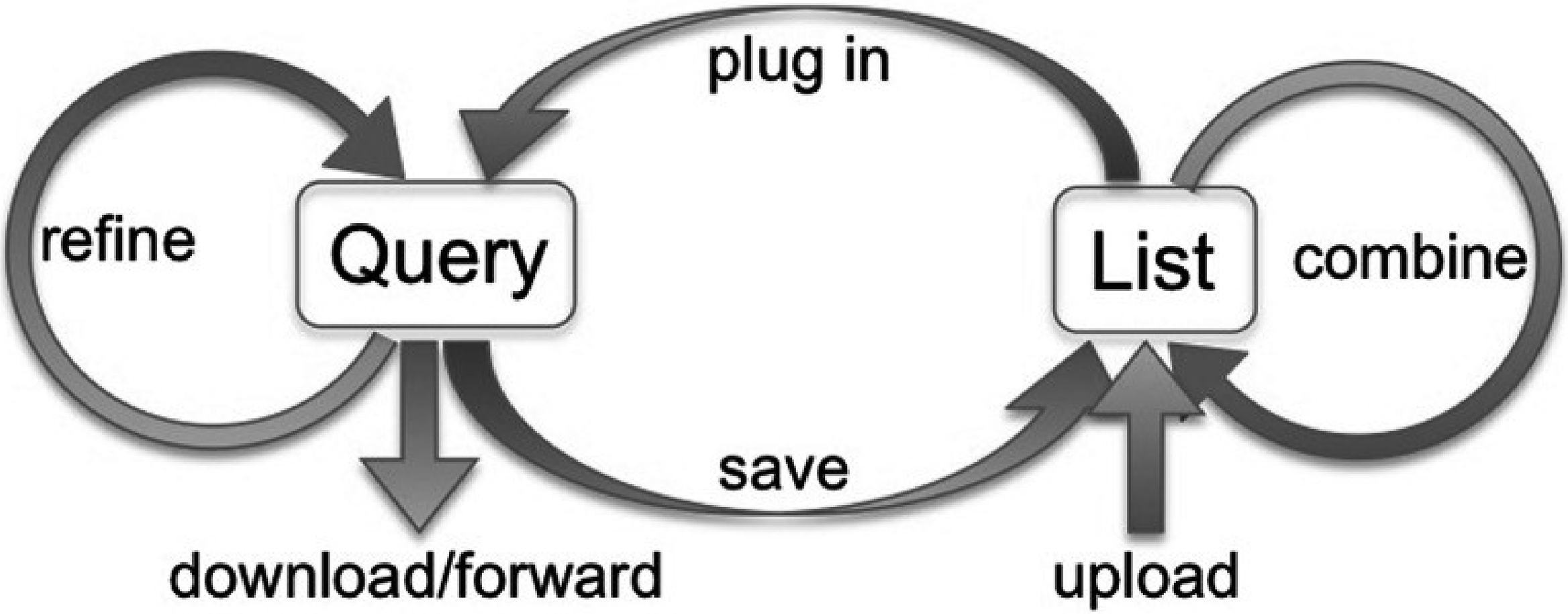
1. Identify the sets of genes you have created under the lists “view” tab.

2. Use the list set operations available on this page to intersect the list of diabetes genes you created with the query builder with your previous set of genes (genes expressed in adipose tissue that interact with pparg) created in exercise 6.

3. We now want to know if any of these genes have been identified in GWAS studies. Run a template on your intersected list to find this out.

4. Use the column summary to find if any of the GWAS phenotypes are related to diabetes.

Analysis Workflows

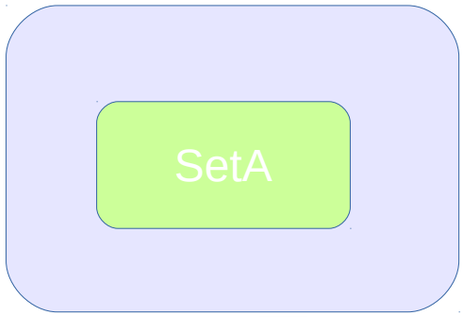


Motenko H, Neuhauser SB, O'Keefe M, Richardson JE. MouseMine: a new data warehouse for MGI. Mamm Genome. 2015 Aug;26(7-8):325-30. doi: 10.1007/s00335-015-9573-z. PubMed PMID: 26092688; PubMed Central PMCID: PMC4534495

Analysis Workflow

1. We looked at gene expression in adipose tissue and refined this to find gene UP regulated in this tissue. We saved these as a list – Set A.

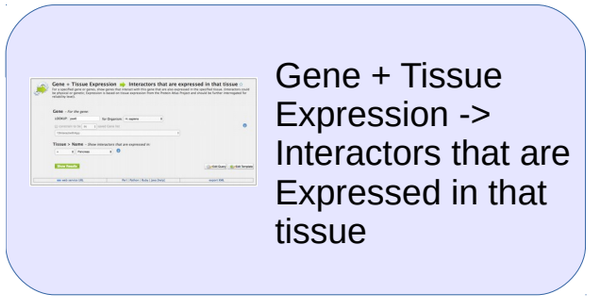
Tissue → Gene Expression
(Array Express E-MTAB-62)



UP regulated in adipose tissue

Analysis Workflow

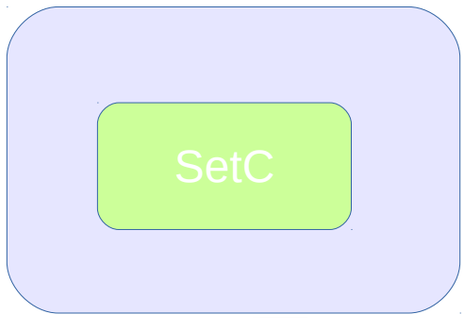
2. Genes saved as SetA were “plugged in” to a second template to identify if any of these genes are known to interact with Pparg. These genes were saved - SetB



Genes that interact with PPARG

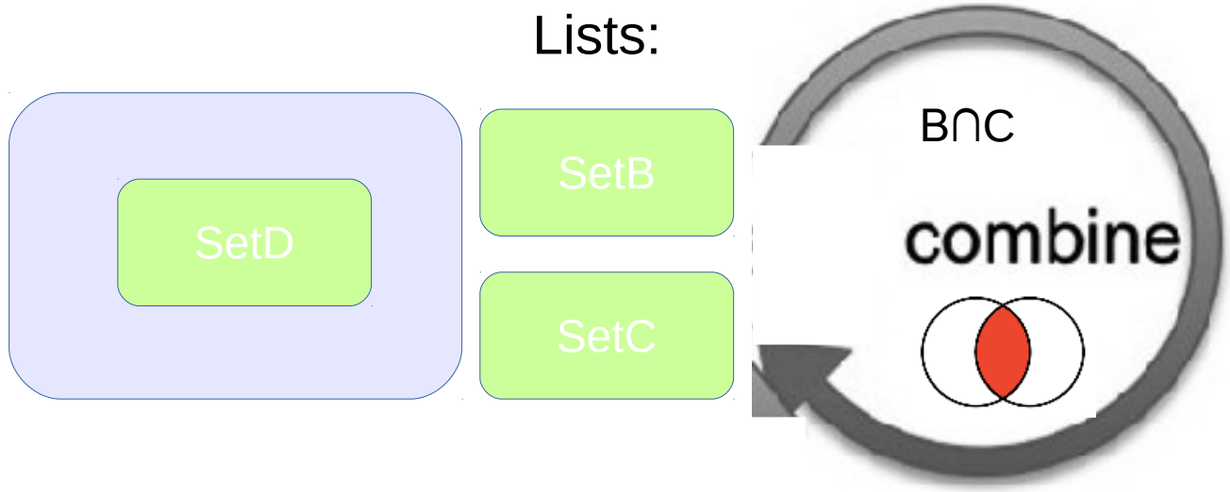
Analysis Workflow

3. A query built through the query builder identified a set of genes involved in Diabetes -> setC



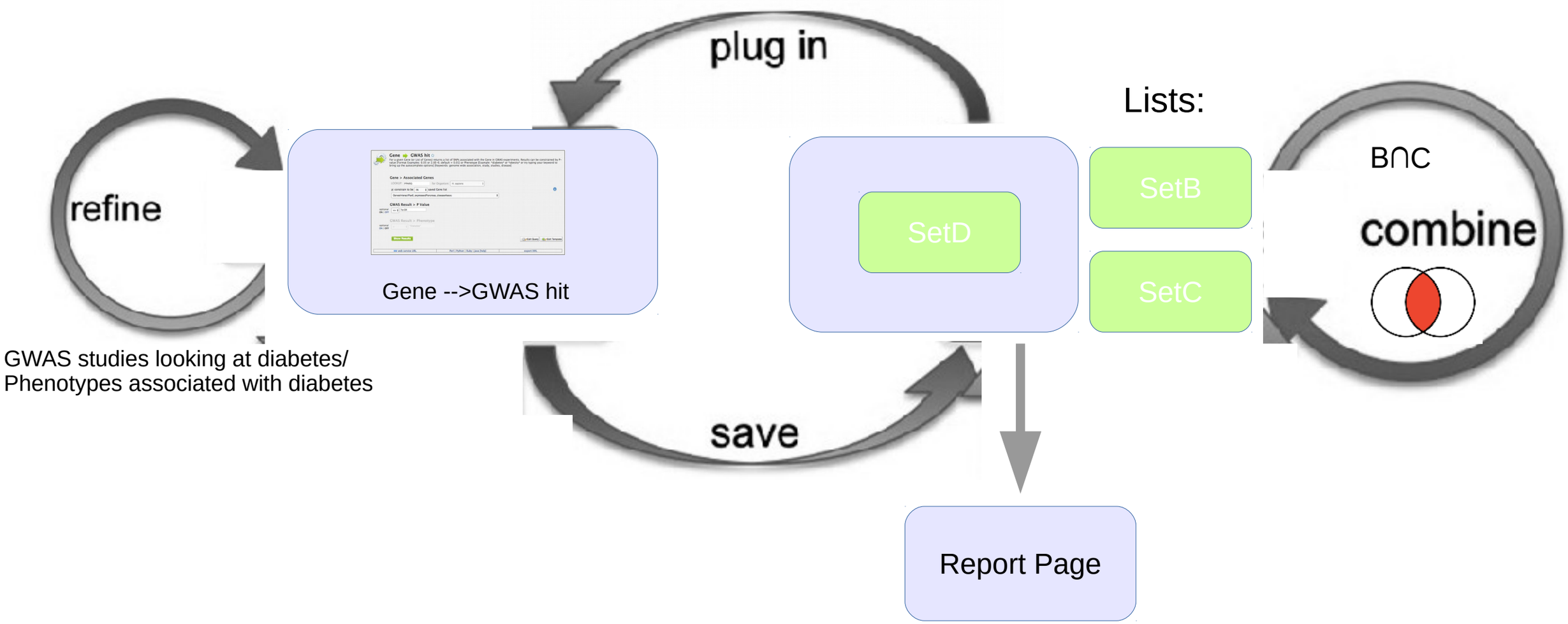
Analysis Workflow

4. A list intersection between setB and setC identified genes expressed in adipose tissue that interact with PPARG that also have some association with diabetes →setD



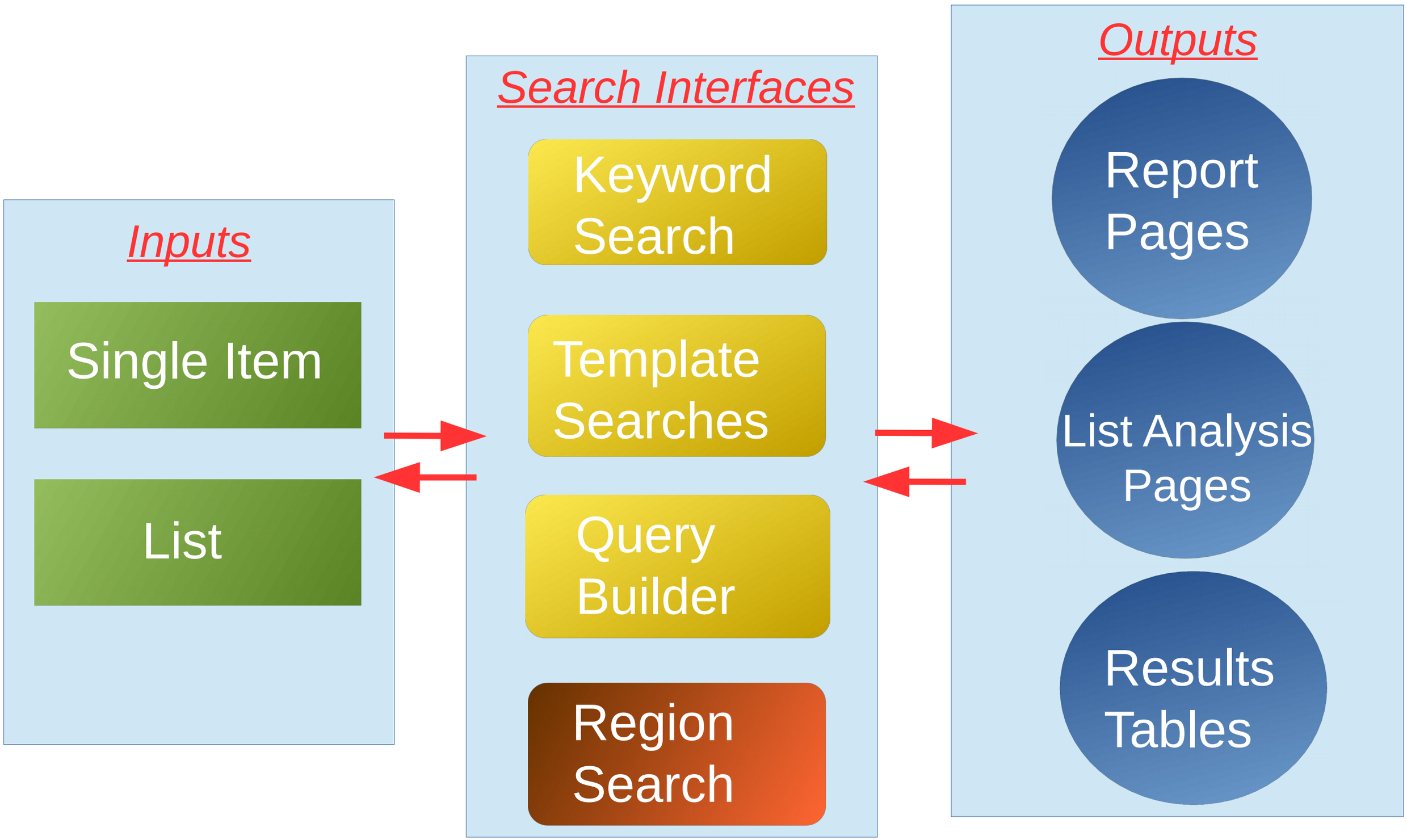
Analysis Workflow

5. ListD contained only one gene but we could explore this further through its report page and through further templates – for example looking at GWAS studies.



GWAS studies looking at diabetes/
Phenotypes associated with diabetes

The Web Interface



Data Analysis: RegionSearch

The Region Search allows you to search for features that overlap a list of genome coordinates.

- Any or selected genome features can be searched.
- Accepts base or interbase coordinates
- Region to be searched can be extended upstream and downstream

Exercise 9: Region Search:

Using **FlyMine:**

1. Select the example set of regions
2. De-select the features and re-select Genes and Regulatory regions

3. Extend the search by 5kb
4. Run the search

Examine the results and:

5. Create a list of all genes found.
6. Create a list of the regulatory regions found in the first genomic span.

HELP?

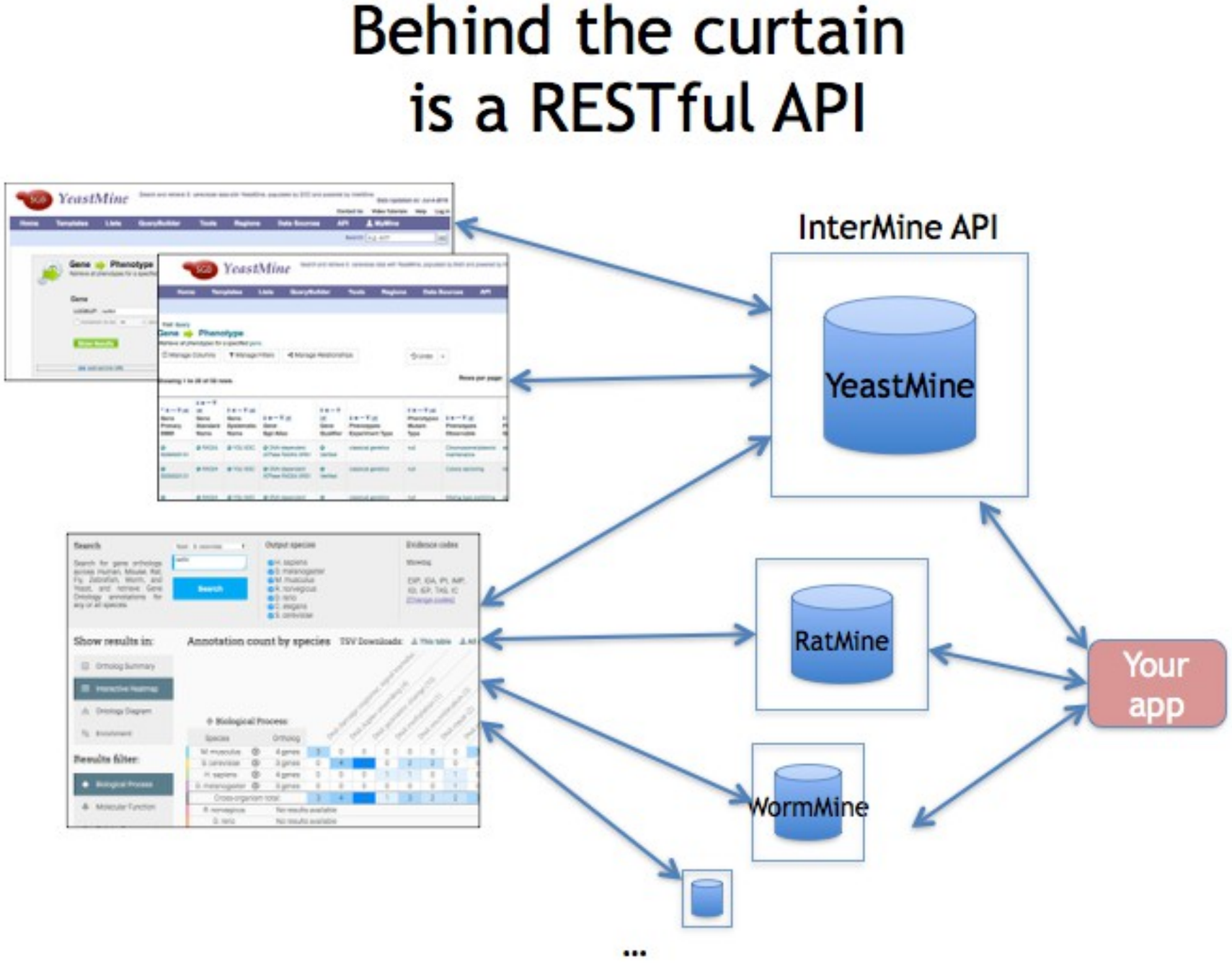
- FlyMine - extensive manual and videos under the ‘help’ link. These apply to all InterMine databases

- Each InterMine has it’s own help pages, videos and tutorials

Email Us: Every InterMine page has a ‘questions/comments’ link.
Use this if you get stuck - we try and reply within 24 hrs.

support@flymine.org; support@humanmine.org

Web Services



Web Services

Using the API, you can:

- Execute an arbitrary query
- Discover the set of available templates, and execute one, supplying it parameter values
- Do a keyword (text) search
- Upload a batch of identifiers and resolve them to database objects
- Create lists, combine them with set operations
- Operate in authenticated or non-authenticated (public) mode

Etc… pretty much anything you can do via the web interface

Extensive documentation at www.intermine.org

Web Services

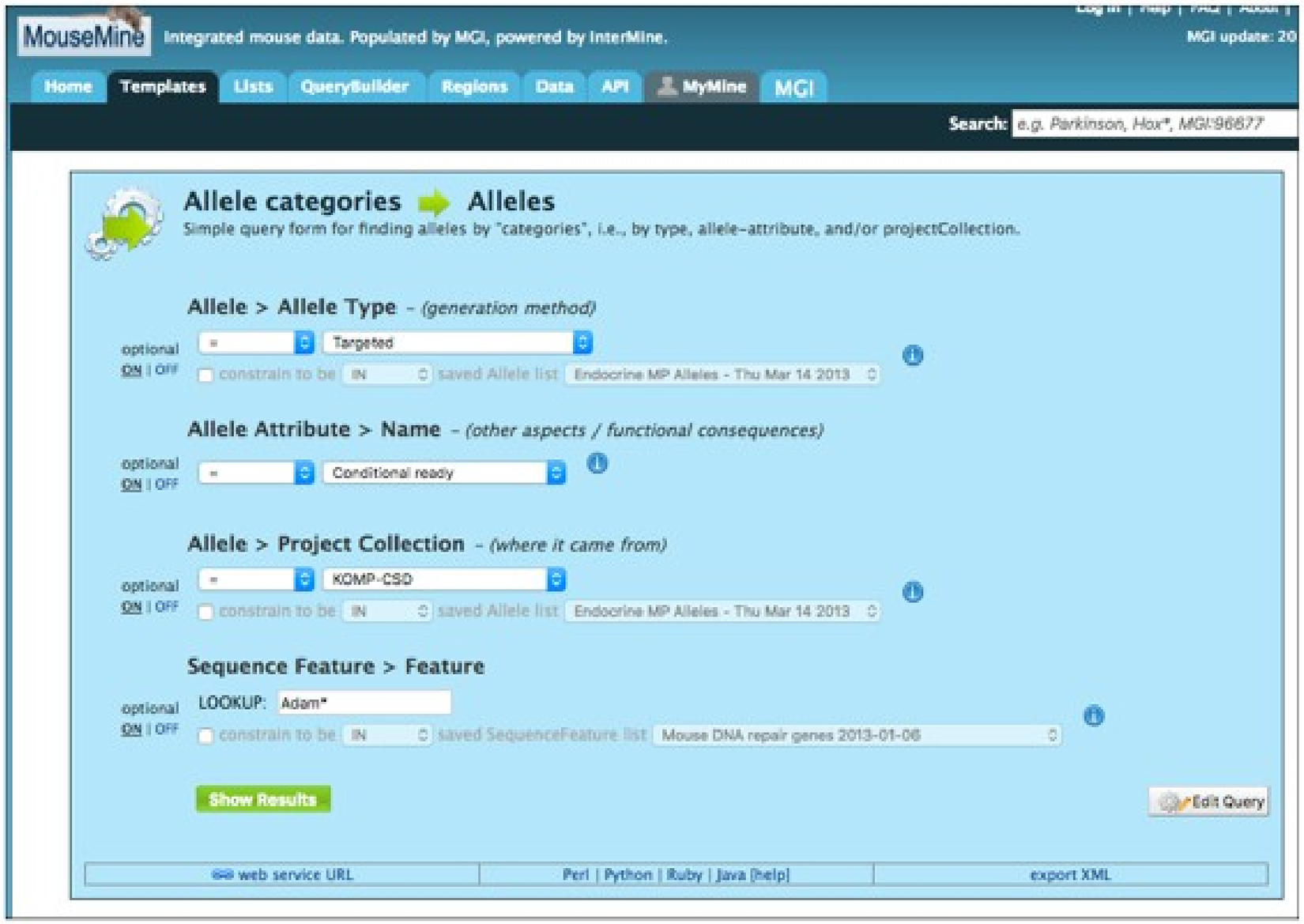
Options for accessing the API

- Directly, via URLs (curl example)
- Helper libraries available for:
  - Python
  - Perl
  - Java
  - Ruby
  - JavaScript

Web Services

Access the API in 3 easy steps:

1. Pick a template and fill out any search fields.



Web Services

2. Click “web service URL” and copy the displayed URL.

Web Services

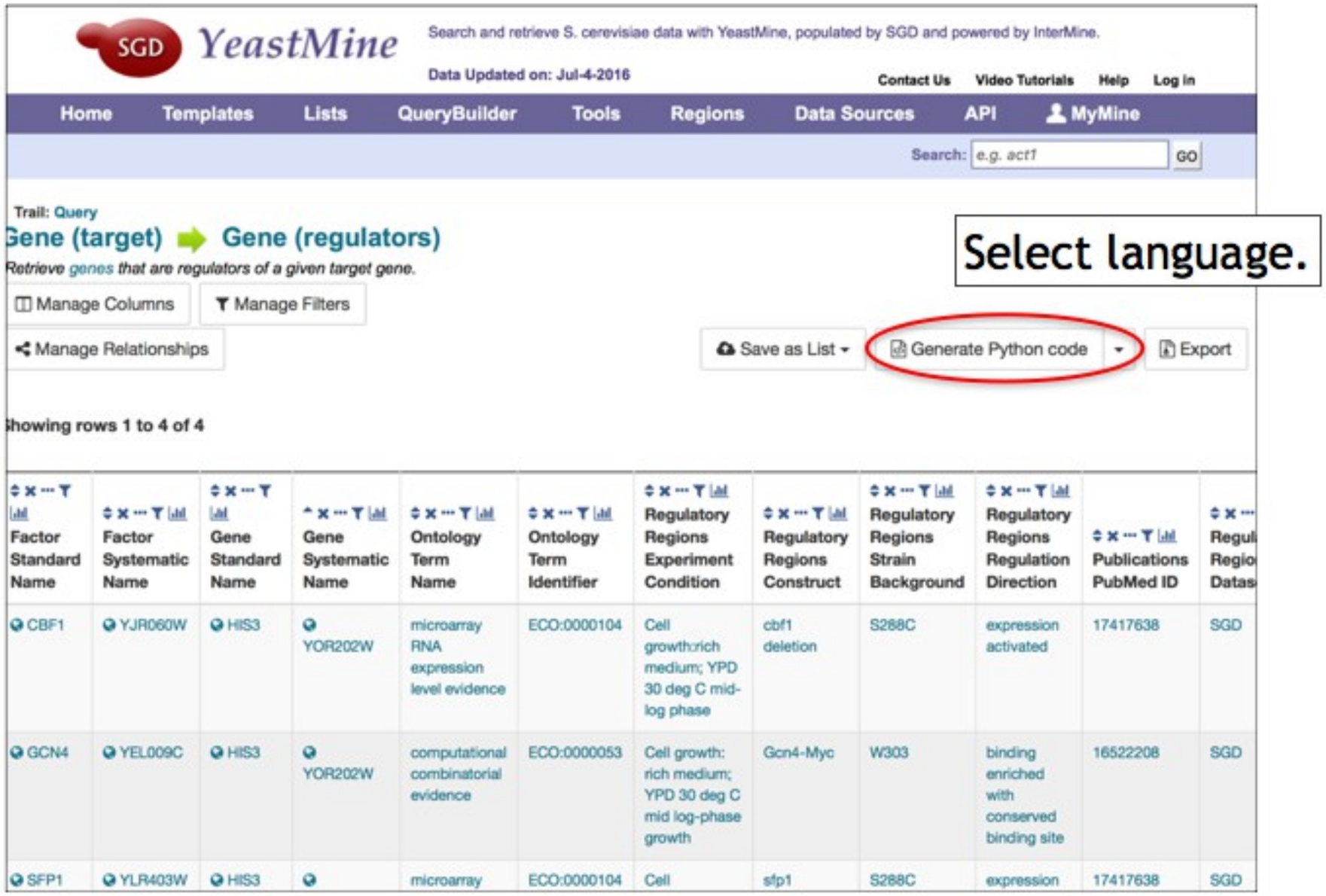
3. Run the query with curl.

```
jer — -bash — 80×24
MLG-OCELOT:~ jer$ curl "http://www.mousemine.org/mousemine/service/template/resu
lts?name=Q_Allele_by_categories&constraint1=Allele.feature&op1=LOOKUP&value1=Ada
m*&extra1=&constraint2=Allele.alleleType&op2=eq&value2=Targeted&constraint3=Alle
le.alleleAttributes.name&op3=eq&value3=Conditional+ready&constraint4=Allele.proj
ectCollection&op4=eq&value4=KOMP-CSD&format=tab&size=10"
```

```
jer — -bash — 80×24
m*&extra1=&constraint2=Allele.alleleType&op2=eq&value2=Targeted&constraint3=Alle
le.alleleAttributes.name&op3=eq&value3=Conditional+ready&constraint4=Allele.proj
ectCollection&op4=eq&value4=KOMP-CSD&format=tab&size=10"
Adam11<tm1a(KOMP)Mbp>   MGI:4840783     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adam18<tm1a(KOMP)Wtsi>  MGI:4458641     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adam21<tm1a(KOMP)Wtsi>  MGI:4820033     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adam32<tm1a(KOMP)Wtsi>  MGI:5293959     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adamdec1<tm1a(KOMP)Wtsi>        MGI:5426623     Targeted        Conditional read
y, Null/knockout, Reporter      KOMP-CSD
Adamts17<tm1a(KOMP)Wtsi>        MGI:4362561     Targeted        Conditional read
y, Null/knockout, Reporter      KOMP-CSD
Adamts3<tm1a(KOMP)Wtsi> MGI:4841315     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adamts6<tm1a(KOMP)Wtsi> MGI:5426651     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adamts7<tm1a(KOMP)Wtsi> MGI:4363113     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
Adamts9<tm1a(KOMP)Wtsi> MGI:4841048     Targeted        Conditional ready, Null/
knockout, Reporter      KOMP-CSD
MLG-OCELOT:~ jer$
```

Web Services

Let InterMine write the code!



Web Services: The basics of a query

3 basic steps:

1. Import service class and create interMine object
2. Define “views”: i.e the results output (like “show” in query builder)
3. Define constraints

4. Process the results

Web Services: The basics of a query

An example using python:

```
from intermine.webservice import Service
Service= Service("http://www.humanmine.org/humanmine/service")
query = service.new_query("Gene")
query.add_view("primaryIdentifier", "symbol")
query.add_constraint("organism.name", "=", "Homo sapiens", code = "A")
query.add_constraint("diseases.name", "CONTAINS", "diabetes", code = "B")
for row in query.rows():
print row["primaryIdentifier"], row["symbol"]
```

Web Services: The basics of a query

An example using python:

```
from intermine.webservice import Service
Service= Service("http://www.humanmine.org/humanmine/service")
query = service.new_query("Gene")
query.add_view("primaryIdentifier", "symbol")
query.add_constraint("organism.name", "=", "Homo sapiens", code = "A")
query.add_constraint("diseases.name", "CONTAINS", "diabetes", code = "B")
for row in query.rows():
print row["primaryIdentifier"], row["symbol"]
```

Web Services: The basics of a query

An example using python:

```
from intermine.webservice import Service
Service= Service("http://www.humanmine.org/humanmine/service")
query = service.new_query("Gene")
query.add_view("primaryIdentifier", "symbol")
query.add_constraint("organism.name", "=", "Homo sapiens", code = "A")
query.add_constraint("diseases.name", "CONTAINS", "diabetes", code = "B")
for row in query.rows():
print row["primaryIdentifier"], row["symbol"]
```

Web Services: The basics of a query

An example using python:

```
from intermine.webservice import Service
Service= Service("http://www.humanmine.org/humanmine/service")
query = service.new_query("Gene")
query.add_view("primaryIdentifier", "symbol")
query.add_constraint("organism.name", "=", "Homo sapiens", code = "A")
query.add_constraint("diseases.name", "CONTAINS", "diabetes", code = "B")
for row in query.rows():
print row["primaryIdentifier"], row["symbol"]
```

Web Services: The basics of a query

An example using python:

```
from intermine.webservice import Service
Service= Service("http://www.humanmine.org/humanmine/service")
query = service.new_query("Gene")
query.add_view("primaryIdentifier", "symbol")
query.add_constraint("organism.name", "=", "Homo sapiens", code = "A")
query.add_constraint("diseases.name", "CONTAINS", "diabetes", code = "B")
for row in query.rows():
    print row["primaryIdentifier"], row["symbol"]
```

Web Services: The basics of a query

An example using perl:

```
use strict;
use warnings;
$, = "\t";
binmode(STDOUT, 'utf8');
no warnings ('uninitialized');
use Webservice::InterMine 'http://www.humanmine.org/humanmine';
my $query = new_query(class => 'Gene');
$query->add_view(qw/
    primaryIdentifier
    symbol
/);
$query->add_constraint(
    path  => 'Gene.organism.name',
    op    => '=',
    value => 'Homo sapiens',
    code  => 'A',
);
$query->add_constraint(
    path  => 'Gene.diseases.name',
    op    => 'CONTAINS',
    value => 'diabetes',
    code  => 'B',
);
my $it = $query->iterator();
while (my $row = <$it>) {
    print $row->{'primaryIdentifier'}, $row->{'symbol'}, "\n";
}Close
```

Web Services: The basics of a query

An example using perl:

```
use strict;
use warnings;
$, = "\t";
binmode(STDOUT, 'utf8');
no warnings ('uninitialized');
use Webservice::InterMine 'http://www.humanmine.org/humanmine';
my $query = new_query(class => 'Gene');
$query->add_view(qw/
    primaryIdentifier
    symbol
/);
$query->add_constraint(
    path  => 'Gene.organism.name',
    op    => '=',
    value => 'Homo sapiens',
    code  => 'A',
);
$query->add_constraint(
    path  => 'Gene.diseases.name',
    op    => 'CONTAINS',
    value => 'diabetes',
    code  => 'B',
);
my $it = $query->iterator();
while (my $row = <$it>) {
    print $row->{'primaryIdentifier'}, $row->{'symbol'}, "\n";
}Close
```

Web Services: The basics of a query

An example using perl:

```
use strict;
use warnings;
$, = "\t";
binmode(STDOUT, 'utf8');
no warnings ('uninitialized');
use Webservice::InterMine 'http://www.humanmine.org/humanmine';
my $query = new_query(class => 'Gene');
$query->add_view(qw/
    primaryIdentifier
    symbol
/);
$query->add_constraint(
    path  => 'Gene.organism.name',
    op    => '=',
    value => 'Homo sapiens',
    code  => 'A',
);
$query->add_constraint(
    path  => 'Gene.diseases.name',
    op    => 'CONTAINS',
    value => 'diabetes',
    code  => 'B',
);
my $it = $query->iterator();
while (my $row = <$it>) {
    print $row->{'primaryIdentifier'}, $row->{'symbol'}, "\n";
}Close
```

Web Services: The basics of a query

An example using perl:

```
use strict;
use warnings;
$, = "\t";
binmode(STDOUT, 'utf8');
no warnings ('uninitialized');
use Webservice::InterMine 'http://www.humanmine.org/humanmine';
my $query = new_query(class => 'Gene');
$query->add_view(qw/
    primaryIdentifier
    symbol
/);
$query->add_constraint(
    path  => 'Gene.organism.name',
    op    => '=',
    value => 'Homo sapiens',
    code  => 'A',
);
$query->add_constraint(
    path  => 'Gene.diseases.name',
    op    => 'CONTAINS',
    value => 'diabetes',
    code  => 'B',
);
my $it = $query->iterator();
while (my $row = <$it>) {
    print $row->{'primaryIdentifier'}, $row->{'symbol'}, "\n";
}Close
```

Web Services: The basics of a query

An example using perl:

```
use strict;
use warnings;
$, = "\t";
binmode(STDOUT, 'utf8');
no warnings ('uninitialized');
use Webservice::InterMine 'http://www.humanmine.org/humanmine';
my $query = new_query(class => 'Gene');
$query->add_view(qw/
    primaryIdentifier
    symbol
/);
$query->add_constraint(
    path  => 'Gene.organism.name',
    op    => '=',
    value => 'Homo sapiens',
    code  => 'A',
);
$query->add_constraint(
    path  => 'Gene.diseases.name',
    op    => 'CONTAINS',
    value => 'diabetes',
    code  => 'B',
);
my $it = $query->iterator();
while (my $row = <$it>) {
    print $row->{'primaryIdentifier'}, $row->{'symbol'}, "\n";
}Close
```

Web Services: Processing results

In Python:

1. Dictionary form
2. List
3. Result row object

1. for gene in query.results (row = “rr”):
   print(gene)

2. for gene in query.results(“list”):
   print(gene[1], gene[2])

3. for gene in query.rows():
   print(gene)

Web Services: Processing results

In Perl:

We will use the results_iterator:

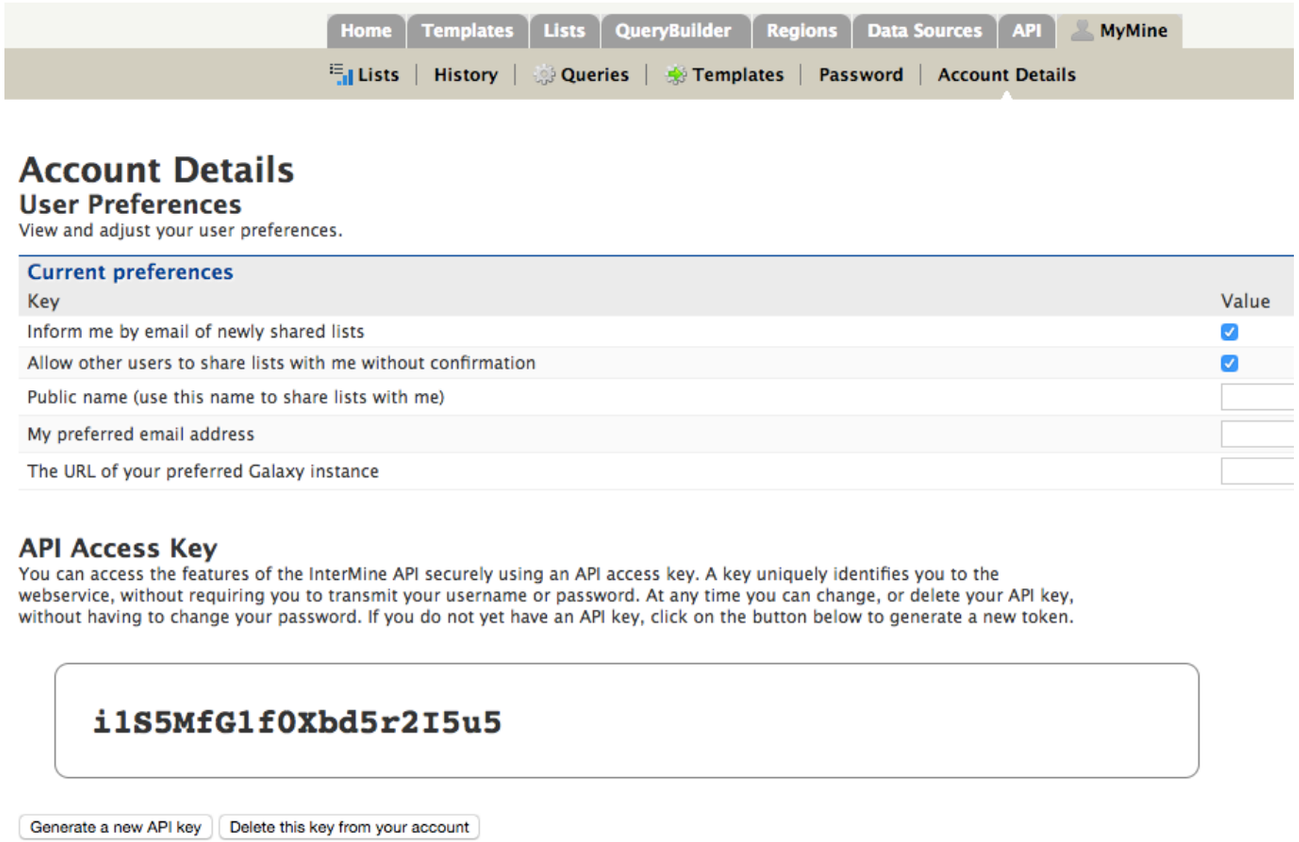
Returns a results object that allows you to iterate through the results row by row, in whatever format you choose.

See the online documentation for other ways to process results.

Web Services: API Tokens

If you need to access your account through the API you need to use an API token. You will need this to create and access your lists.

You can get your API token from your MyMine account:



Web Services: API Tokens

If you need to access your account through the API you need to use an API token. You will need this to create and access your lists.

You can get your API token from your MyMine account:

```
from intermine.webservice import Service
service = Service("http://www.humanmine.org/humanmine/service",
token = "YOUR-API-KEY")
```

```
use Webservice::InterMine
'http://www.humanmine.org/humanmine','i1S5MfG1f0Xbd5r2I5u5';
```

Templates - python

```
from intermine.webservice import Service
service = Service("http://www.humanmine.org/humanmine/service")

# For a given human disease returns a list of associated or
implicated human
# genes (source: OMIM)

template = service.get_template('Dis_Gene')

# You can edit the constraint values below
# A    Disease.name

rows = template.rows(
    A = {"op": "=", "value": "*Diabetes*"}
)

for row in rows:
    print row["genes.primaryIdentifier"], row["genes.symbol"],
row["genes.name"], row["name"], \
        row["identifier"], row["genes.organism.shortName"]
```

Templates - perl

```
use Webservice::InterMine 0.9904 'http://www.humanmine.org/humanmine';

# Description: For a given human disease returns a list of associated or
# implicated human genes (source: OMIM)

my $template = Webservice::InterMine->template('Dis_Gene')
    or die 'Could not find a template called Dis_Gene';

# Use an iterator to avoid having all rows in memory at once.
my $it = $template->results_iterator_with(
    # A:  Disease.name
    opA    => '=',
    valueA => '*Diabetes*',
);

while (my $row = <$it>) {
    print $row->{'genes.primaryIdentifier'}, $row->{'genes.symbol'}, $row-
                    >{'genes.name'},
        $row->{'name'}, $row->{'identifier'}, $row-
>{'genes.organism.shortName'}, "\n";
}
```

Exercise1:

Python:

Tutorial1: The basics of a query
Tutorial2: Adding constraints to a query
Tutorial5: Query results
Tutorial7: Templates

Perl:

http://search.cpan.org/~intermine/Webservice-InterMine-1.0405/lib/Webservice/InterMine/Cookbook.pod

Recipe 1: The bare basics
Recipe 2: Adding constraints
Recipe 5: Dealing with results
Templates::Recipe1 : templates

Exercise2:

Re-create the workflow we did using the webapp using either Python (recommended) or Perl. Remember, most of the code can be auto-generated from the webapp.

Summary:

1. Genes that are upregulated in Adipose tissue (array express data) → LIST A

2. Those genes from LISTA that interact with PPARG → LISTB

3. Genes involved in the disease diabetes (omim) → LISTC

4. Intersect of list B and List C

The InterMine Team

PI: Gos Micklem

Software developers: Julie Sullivan
Sergio Contrino
Joshua Heimbach
Yo Yehudi
Justin Clark-Casey

Biologist: Rachel Lyne

MouseMine: Joel Richardson

Ex-Team: Richard Smith, Alex Kalderimis, Mike Lyne, Kim Rutherford, Matthew Wakeling, Xavier Watkins, Andrew Varley, Mark Woodbridge, Tom Riley, Peter Mclaren, Debashis Rana, Wenyan Ji, Markus Brosch, Florian Reising, Matthew Chadwick

The Web Interface

Inputs

Search Interfaces

Outputs