

# Integrating Ontological and Genomic Analysis with Galaxy

This tutorial accompanies the “[Integrating Ontological and Genomic Analysis with Galaxy](#)” workshop at [ICBO 2016](#), on August 1, 2016. This tutorial walks through all the steps taken during the workshop, and can be used any time after the workshop to reproduce the example analysis used in the workshop.

[What's our motivation here?](#)

[Our question: Repeats and coding regions](#)

[The plan](#)

[Time allowing ...](#)

[Training datasets](#)

[Galaxy](#)

[Get data from UCSC](#)

[Genes \(actually transcripts\)](#)

[Repeats](#)

[Meaningful names](#)

[Count Gene-Repeat overlaps](#)

[A question of position: Operate on Genomic Intervals](#)

[Beware the intron](#)

[Counting overlaps](#)

[Transcripts and Genes](#)

[Which overlap count?](#)

[Deep thoughts 1](#)

[Any patterns here?](#)

[Except don't](#)

**This tutorial is available online at**

**[http://bit.ly/ICBO\\_GXY\\_PDF](http://bit.ly/ICBO_GXY_PDF)**

## What's our motivation here?

Our goal with this tutorial is to demonstrate how to integrate Galaxy with ontological analysis. Galaxy was created to support genomic analysis. There are also a wealth of ontology tools out there. This example will

- do some basic genomic analysis in Galaxy,
- run those results through external ontology tools,
- load the result of the ontology tools back into Galaxy, and
- use Galaxy to analyze those imported, ontology-based datasets

We are going to ask a simple question and then highlight how to use Galaxy and then show one way to integrate genomic and ontological analysis inside Galaxy.

If you are already a Galaxy expert then there won't be much new in this tutorial.

## Our question: Repeats and coding regions

*Note: The instructor has a computing background. This is a question that even a computer scientist can understand.*

I have an intuition that repeats overlapping with the coding regions of genes in genomic sequence should be an uncommon and possibly rare thing. However, I have also learned that if you can imagine something, then it probably happens somewhere in the tree of life.

## The plan

The big picture plan is

1. Find genes have repeats that overlap with their coding regions
2. Identify genes that have a high number overlapping repeats
3. Identify any common function(s) in the genes with high numbers of overlaps
4. Explore how the definition of "high number of overlapping repeats" affects the identified gene functions.

We are going to ask this question about the human genome, using the latest and greatest assembly and annotation.

## Time allowing ...

If there is time, we'll also

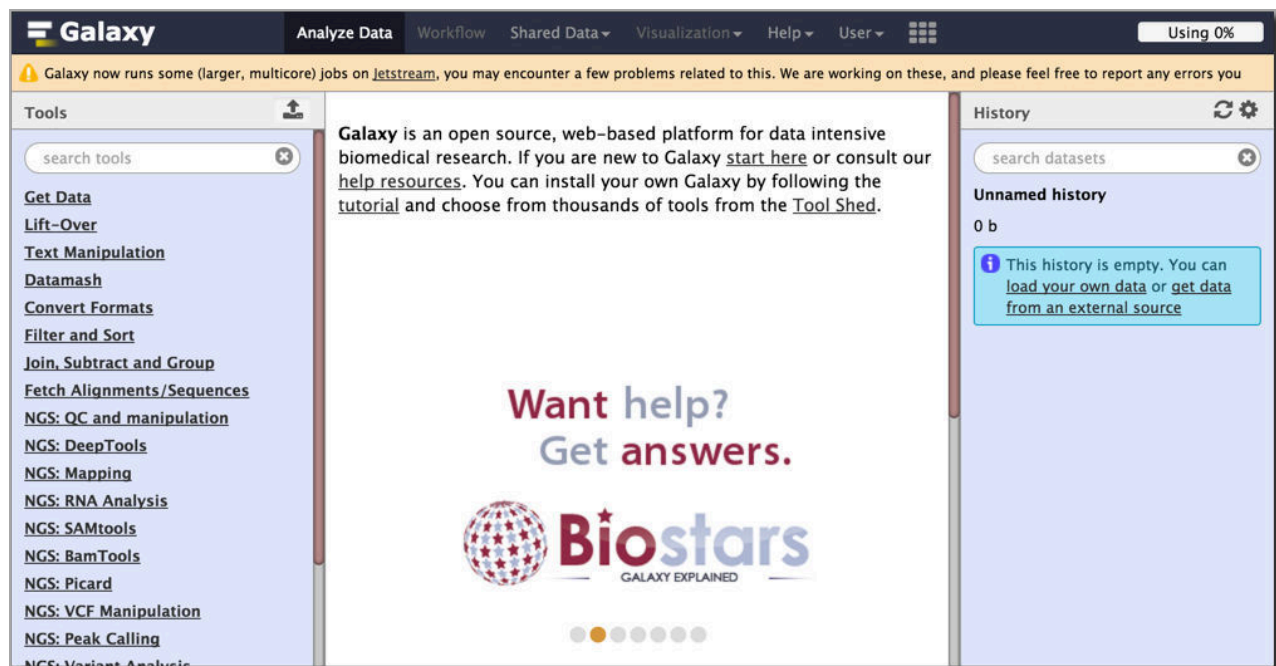
1. Explore how using different repeat identification software affects the identified genes and gene functions, and determine if our analysis is robust when using different software.

## Training datasets

This tutorial operates on subsets of the full datasets most of the time. This reduces disk space usage, and wait times while tasks run. At certain key times we will abandon our work so far and “fast-forward” to examples that use complete datasets. This allows us to learn Galaxy with smaller datasets, and to then run final analysis steps with complete datasets.

## Galaxy

At the ICBO workshop we'll be using Galaxy instances on the AWS Cloud that were created specifically for the workshop (see slides for URLs). If you are following this tutorial after the workshop, you can run it on [usegalaxy.org](http://usegalaxy.org), the Galaxy Project's public server, or [test.galaxyproject.org](http://test.galaxyproject.org), the project's test server. Published histories and workflows for this exercise are available on the test server.



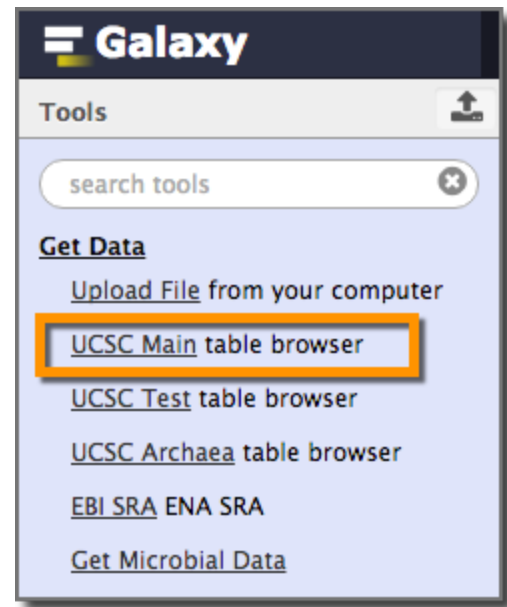
Galaxy instances (typically) have a three panel design. The left panel shows tools available on this Galaxy instance. The right panel shows your current history - the work that you have done so far. The center panel is used to run tools and to examine results.

## Get data from UCSC

The [UCSC Table Browser](#) exposes the data behind all of the tracks shown in the [UCSC Genome Browser](#). UCSC hosts genomic data for a broad swath of animal and microbial species, and if you are doing genomic research on a species that UCSC has then it is an excellent source of genomic data.

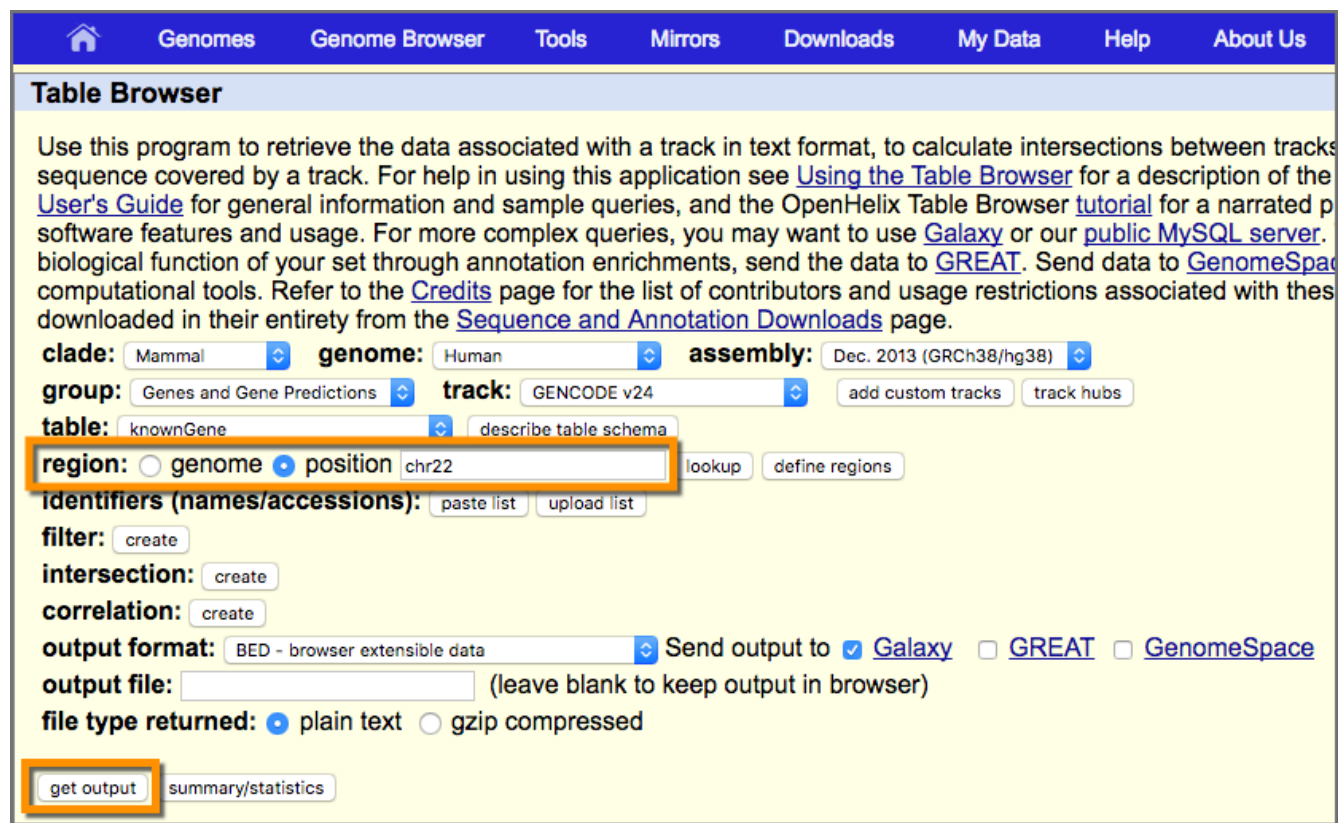
Since we are interested in human, the species that the UCSC browser was first created for, let's get our gene and repeat data from there.

To get to the UCSC table browser from Galaxy *click* **Get Data** in the tool panel, and then *click* **UCSC Main Table Browser**.



## Genes (actually transcripts)

In the table browser, leave everything at it's default values (Mammal, Human, Dec 2013 GRCh38/hg38, ...) except **region**. *Select position* and *enter chr22*. *Click get output*.

A screenshot of the UCSC Table Browser web interface. The page has a blue header with navigation links: Genomes, Genome Browser, Tools, Mirrors, Downloads, My Data, Help, and About Us. The main content area is titled 'Table Browser' and contains a detailed description of the tool. Below the description are several dropdown menus and input fields: 'clade' (Mammal), 'genome' (Human), 'assembly' (Dec. 2013 (GRCh38/hg38)), 'group' (Genes and Gene Predictions), 'track' (GENCODE v24), 'table' (knownGene), and 'region' (position). The 'region' dropdown is highlighted with an orange box, and 'chr22' is entered in the adjacent text field. Below these are buttons for 'lookup' and 'define regions'. Further down are sections for 'Identifiers (names/accessions)', 'filter', 'intersection', and 'correlation', each with a 'create' button. The 'output format' is set to 'BED - browser extensible data', and 'Send output to' is checked for 'Galaxy'. The 'output file' field is empty, and 'file type returned' is set to 'plain text'. At the bottom, the 'get output' button is highlighted with an orange box, and there is a link for 'summary/statistics'.

Which does not get you any output, but does send you to the next and final page before you do get output:

Output knownGene as BED

☐ Include [custom track](#) header:

name=

description=

visibility=

url=

**Create one BED record per:**

☒ Whole Gene

☐ Upstream by  bases

☐ Exons plus  bases at each end

☐ Introns plus  bases at each end

☐ 5' UTR Exons

☐ Coding Exons

☐ 3' UTR Exons

☐ Downstream by  bases

Note: if a feature is close to the beginning or end of a chromosome and upstream/downstream bases are added, the order to avoid extending past the edge of the chromosome.

Ignore the custom track header section.

In the **Create one BED record per** section we have a choice. For now, we are going to ignore that choice. Leave **Whole Gene** selected, and *click Send query to Galaxy*.

Your browser is returned to your Galaxy instance and a big green box shows up on the screen which is then quickly replaced by the main Galaxy window. An item appears in your history, initially as a gray box. The gray indicates that the upload (or any other task) has been queued and is waiting to run. This is a small file and the actual upload won't take very long. However, sometimes usegalaxy.org is very heavily loaded and even simple tasks like uploads may be queued for a while. Once a task starts its color changes to yellow. When the task completes the box becomes green. If there is an error, the box becomes red and a bug icon will be displayed. Basically:

Gray → Yellow → Green = Task done!

Let's take a look at what we have. First to see a preview, *click on the dataset name*. This displays a summary of information about the dataset, including how big it is (~4K genes on chr22), what datatype it is (BED, more on that later), and that it uses coordinates from the hg38 human reference genome assembly.

History

search datasets

Unnamed history

1 shown

462.45 KB

1: UCSC Main on Human: knownGene (chr22:1-50818468)

4,093 regions

format: bed, database: hg38

display in IGB View

1.Chrom	2.Start	3.End	4.Name
chr22	10736170	10736283	uc062bdo.1

It also shows a one line preview of the dataset itself. To see the full dataset *click* the dataset's **eye icon**. This displays the dataset in the middle panel.

1	2	3	4	5	6	7	8	9	10	11	12
chr22	10736170	10736283	uc062bdo.1	0	-	10736170	10736170	0	1	113,	0,
chr22	10936022	10936161	uc062bdp.1	0	-	10936022	10936022	0	1	139,	0,
chr22	11065973	11067346	uc062bdq.1	0	-	11065973	11067346	0	2	42,12,	0,1361,
chr22	11066500	11068089	uc062bdr.1	0	+	11066500	11068089	0	2	15,105,	0,1484,
chr22	11249808	11249959	uc062bds.1	0	-	11249808	11249808	0	1	151,	0,
chr22	11253604	11253719	uc062bdt.1	0	-	11253604	11253604	0	1	115,	0,
chr22	11275564	11275637	uc062bdu.1	0	-	11275564	11275564	0	1	73,	0,
chr22	11456855	11456937	uc062bdv.1	0	+	11456855	11456855	0	1	82,	0,
chr22	11478157	11478241	uc062bdw.1	0	-	11478157	11478157	0	1	84,	0,
chr22	11607866	11607939	uc062bdx.1	0	-	11607866	11607866	0	1	73,	0,
chr22	11628659	11628774	uc062bdy.1	0	+	11628659	11628659	0	1	115,	0,
chr22	11685894	11685976	uc062bdz.1	0	+	11685894	11685894	0	1	82,	0,
chr22	12629371	12629473	uc062bea.1	0	-	12629371	12629371	0	1	102,	0,
chr22	12779258	12779341	uc062beb.1	0	-	12779258	12779258	0	1	83,	0,
chr22	12779975	12780057	uc062bec.1	0	-	12779975	12779975	0	1	82,	0,
chr22	12780119	12780203	uc062bed.1	0	-	12780119	12780119	0	1	84,	0,
chr22	15273854	15273961	uc062bee.1	0	+	15273854	15273854	0	1	107,	0,
chr22	15282556	15288670	uc062bef.1	0	-	15282556	15282556	0	1	6114,	0,
chr22	15298377	15304556	uc062beg.1	0	-	15298377	15298377	0	1	6179,	0,
chr22	15528157	15529139	uc011agd.3	0	+	15528158	15529139	0	1	982,	0,
chr22	15557576	15560694	uc062beh.1	0	-	15557576	15557576	0	3	1015,109,170,	0,2579,2948,
chr22	15600907	15604882	uc062bei.1	0	-	15600907	15600907	0	3	1607,82,41,	0,3144,3934,

This is a 12 column BED file. [BED format](#) is used to represent genomic annotation, in this case, the *transcripts* on chromosome 22. BED files can have 3 or more columns. The first 3 columns are always the chromosome ID/name (or scaffold, or contig, or whatever you have), and the start and stop position of the feature on that chromosome.

Other columns of interest:

- 4: The UCSC transcript name
- 6: The strand the feature is on
- 10-12: The intron/exon structure of this transcript.

*Note that we don't have the gene name or symbol anywhere.*

## Repeats

Next let's get the repeats from UCSC as well. Again *click* **Get Data** in the tool panel, and then *click* **UCSC Main Table Browser**. *Change group* to **Repeats**. Leave the track as **RepeatMasker.**, and leave everything else unchanged as well. *Click* **Send query to Galaxy**. [RepeatMasker](#) is "a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences." It uses a curated database of known repeats to identify repetitive regions. This retrieves regions that were identified as repeats by UCSC using RepeatMasker.



**Table Browser**

Use this program to retrieve the data associated with a track in text format, to calculate intersections between tracks sequence covered by a track. For help in using this application see [Using the Table Browser](#) for a description of the [User's Guide](#) for general information and sample queries, and the OpenHelix Table Browser [tutorial](#) for a narrated presentation of software features and usage. For more complex queries, you may want to use [Galaxy](#) or our [public MySQL server](#). For biological function of your set through annotation enrichments, send the data to [GREAT](#). Send data to [GenomeSpace](#) computational tools. Refer to the [Credits](#) page for the list of contributors and usage restrictions associated with these data downloaded in their entirety from the [Sequence and Annotation Downloads](#) page.

clade:  genome:  assembly:

group:  track:

table:

region: ☐ genome ☒ position

identifiers (names/accessions):

filter:

intersection:

output format:  Send output to ☒ [Galaxy](#) ☐ [GREAT](#) ☐ [GenomeSpace](#)

output file:  (leave blank to keep output in browser)

file type returned: ☒ plain text ☐ gzip compressed

This again sends you to the 2nd page. Leave **Whole Gene** selected, and *click* **Send query to Galaxy**.

The resulting dataset has almost 80,000 records in it. Each of these is a region of the genome identified by RepeatMasker as a repeat. The RepeatMasker dataset is a 6 column BED:

1	2	3	4	5	6
chr22	10510227	10510528	AluSx1	2021	+
chr22	10511018	10511332	L1MC5a	781	-
chr22	10511479	10511791	L1MB1	524	+
chr22	10511878	10512212	L1MB1	313	+
chr22	10512454	10512692	L1MB1	656	+
chr22	10512706	10514778	L1MB1	11092	+
chr22	10514778	10515050	AluSx1	1933	+
chr22	10515050	10515074	L1MB1	11092	+
chr22	10515074	10515121	(GAAG)n	52	+
chr22	10515121	10516103	L1MB1	11092	+
chr22	10516114	10516222	(TA)n	47	+
chr22	10516223	10516285	LTR66	237	-
chr22	10516287	10516630	L1MB1	1504	+
chr22	10516635	10517247	L2a	1062	-
chr22	10517290	10517437	L1MEh	237	-
chr22	10518783	10519114	MLT1A0	1234	+
chr22	10519673	10519746	AluJo	474	-
chr22	10519746	10519816	MER52A	291	-
chr22	10519799	10520945	MER52A	3779	-
chr22	10520950	10521193	AluJo	1258	-
chr22	10522243	10522328	MLT1A0	1424	+
chr22	10522328	10522608	AluSg	2274	+
chr22	10522608	10522644	(AATA)n	39	+
chr22	10522644	10522926	MLT1A0	1424	+
chr22	10523619	10523679	(TAATTGA)n	14	+
chr22	10523878	10524059	OldhAT1	276	+

History

search datasets

Unnamed history  
2 shown

3.22 MB

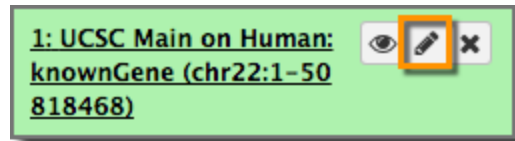
2: UCSC Main on Human: rmsk (chr22:1-50818468)  
79,521 regions  
format: bed, database: hg38  
display in IGB View

1: UCSC Main on Human: knownGene (chr22:1-50818468)  
4,093 regions  
format: bed, database: hg38  
display in IGB View

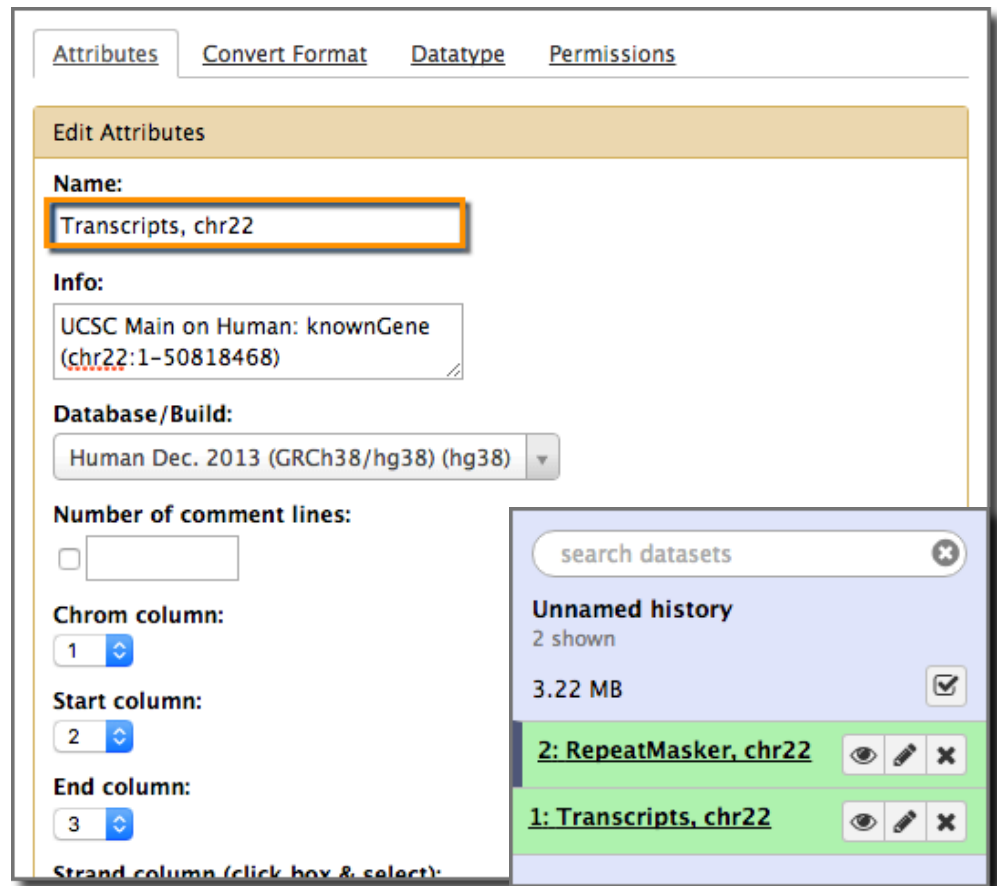
The first 3 columns contain the same information. Column 4 is the type of repeat, and Column 5 is the score for this region from RepeatMasker.

## Meaningful names

We have two datasets with long and obscure names. This is a good time to give them clearer names. To rename a dataset, *click* on its **pencil icon**.



This brings up the **Edit Attributes** window in the middle frame. *Change* the **Name** to something like **Transcripts, chr22** or **RepeatMasker, chr22**. You can also copy the previous name to the Info field, as it is informative (albeit somewhat opaque). *Click* **Save**. Repeat for the other dataset. You will now have the much more memorable history:

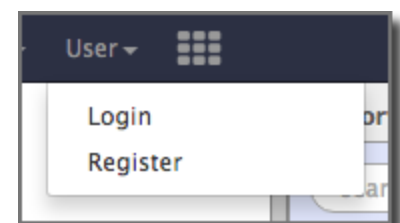


## Create a login and name our history

While we are naming things, let's name our current history. To do that we first need to create an account on this Galaxy server. You don't have to login to use Galaxy, but if you want to use the full power of Galaxy (and have more than one history) you'll need to login.

To create a login *click* on the **User** pulldown and *select* **Register**. This brings up a form asking for:

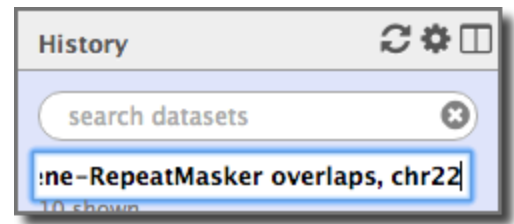
1. **Your email address:** Use one you can remember
2. **Your password:** This connection is not secure. *Use a throwaway password.*
3. **Your username:** Use your last name.





And *click* **Submit**. You are now logged in. *Click* **Analyze Data** in the top menu to return to the home page.

You can now name your history by *clicking* on **Unnamed history**, *typing* a new name, and *hitting* **return**. Use a name that has RepeatMasker in it.



## Count Gene-Repeat overlaps

We now have the two datasets we are interested in: genes (well, transcripts), and repeats. Let's see how they relate to each other - specifically how they overlap in the genome.

### The Plan

We need to first find which transcripts have overlapping repeats, and then count which ones have the most. To do that, we'll first join any transcripts and overlapping repeats side by side. In Galaxy, a common approach is get all relevant information into a record and then ask questions using those very wide records.

Once we have all the transcript-repeat pairings we'll count the number of pairings each transcript participates in. This will tell us how many repeats overlap with each coding region. This won't quite tell us how many overlaps the transcript has, but it's a step.

## A question of position: Operate on Genomic Intervals

This is fundamentally a question about how items in two datasets relate to each other in space. It's a question about positions in the genome. When you have a question about position, a good place to look for tools to help answer it is the **Operate on Genomic Intervals** toolbox. Click on it to see what the tools are.

Several have promising names including **Intersect**, **Join**, **Concatenate**, and **Merge**. Click on each tool name to see its parameters and description. After some poking around we settle on **Join** as the most promising tool.

Join takes two datasets, tests that they have at least the minimum specified overlap, and

then produces a paired record for each such overlap. In this case we could join the transcripts dataset with the RepeatMasker dataset.

### Beware the intron

However, in this case there's a problem. The **Operate on Genomic Intervals** tools look at only the first 3 columns of BED files. Our transcripts dataset has information about exons, but it's embedded in columns 10-12. If we run Join using transcripts we'll find repeats that overlap with introns as well as exons.

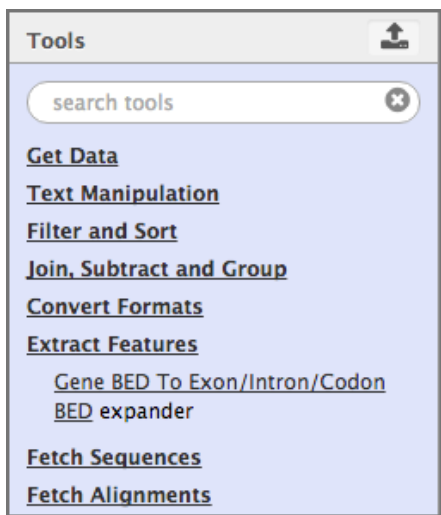
What to do? We have a couple options here. First, remember the second page at UCSC when we retrieved the transcript information:

At that time we could have selected **Coding Exons** instead of **Whole Gene**. This would have returned a list of exons, instead of transcripts, and we could then do the overlap comparison directly between exons and repeats. However, the transcript name would now be buried in the exon name and this approach would take an extra step or two to isolate the transcript name. *This approach works as well - we just aren't using it.*

**Create one BED record per:**

- ☒ Whole Gene
- ☐ Upstream by  bases
- ☐ Exons plus  bases at each end
- ☐ Introns plus  bases at each end
- ☐ 5' UTR Exons
- ☐ **Coding Exons**
- ☐ 3' UTR Exons
- ☐ Downstream by  bases

Note: if a feature is close to the beginning or end of a chromosome, this tool will extend the feature in order to avoid extending past the edge of the chromosome.



A second approach, and the one we'll take, is to use a tool inside Galaxy to generate a coding exon BED dataset from the transcript BED dataset. The tool we want is **Gene BED To Exon/Intron/Codon BED expander**. It can be found in one of two places, depending on which server you are on. On usegalaxy.org it's in the **Operate on Genomic Intervals** toolkit. On test and cloud instances it's in the **Extract Features** toolkit.

Select **Coding Exons only** and the **Transcripts** dataset and then *click Execute*.

This tool looks at columns 10-12 and generates and exon record in the output dataset for every exon in every transcript.

Gene BED To Exon/Intron/Codon BED expander (Galaxy Version 1.0.0) Options

**Extract**

Coding Exons only

**from**

1: Transcripts, chr22

this history item must contain a 12 field BED (see below)

The tool adds a new dataset to our history. Poke it in the eye to take a closer look:

1	2	3	4	5	6
chr22	11065973	11066015	uc062bdq.1	0	-
chr22	11067334	11067346	uc062bdq.1	0	-
chr22	11066500	11066515	uc062bdr.1	0	+
chr22	11067984	11068089	uc062bdr.1	0	+
chr22	15528158	15529139	uc011agd.3	0	+
chr22	15690077	15690709	uc010gqp.3	0	+
chr22	15695370	15695485	uc010gqp.3	0	+
chr22	15695644	15695818	uc010gqp.3	0	+
chr22	15698661	15698768	uc010gqp.3	0	+
chr22	15700077	15700215	uc010gqp.3	0	+
chr22	15702685	15702756	uc010gqp.3	0	+
chr22	15708019	15708090	uc010gqp.3	0	+
chr22	15709781	15709826	uc010gqp.3	0	+
chr22	15710867	15711034	uc010gqp.3	0	+
chr22	15719659	15719777	uc010gqp.3	0	+
chr22	15690077	15690314	uc062bej.1	0	+
chr22	15690425	15690709	uc062bej.1	0	+
chr22	15695370	15695485	uc062bej.1	0	+
chr22	15695644	15695818	uc062bej.1	0	+
chr22	15698661	15698768	uc062bej.1	0	+
chr22	15700077	15700215	uc062bej.1	0	+

1.Chrom	2.Start	3.End	4.Name
chr22	11065973	11066015	uc062bdq.1

The name of each generated exon is the name of the transcript it is a part of.

*Rename* the new dataset something with **Exons** in the name.

We can now do the join operation. Open the **Operate on Genomic Intervals** toolbox and select **Join**. Select the **Exons** dataset as the **First dataset** and the **RepeatMasker** dataset for the **Second dataset**. Leave **with min overlap** as **1**.

Click **Execute** to run the join.

The resulting dataset has 12 columns. The first 6 are from the Exons dataset (which we specified as the first dataset), and the last 6 are from the RepeatMasker dataset. Each record is a pairing of exon and repeat information where the two overlap:

1	2	3	4	5	6	7	8	9	10	11	12
chr22	11065973	11066015	uc062bdq.1	0	-	chr22	11064567	11067436	REP522	6038	+
chr22	11067334	11067346	uc062bdq.1	0	-	chr22	11064567	11067436	REP522	6038	+
chr22	11066500	11066515	uc062bdr.1	0	+	chr22	11064567	11067436	REP522	6038	+
chr22	11067984	11068089	uc062bdr.1	0	+	chr22	11067982	11068155	REP522	348	-
chr22	15697373	15697532	uc062bek.1	0	+	chr22	15697530	15697651	AluJb	1340	-
chr22	15697373	15697532	uc062bek.1	0	+	chr22	15697355	15697497	AluJb	1340	-
chr22	15697373	15697532	uc062bek.1	0	+	chr22	15697497	15697530	(TTTTTA)n	32	+
chr22	17105852	17105954	uc002zly.5	0	+	chr22	17105952	17106169	MIRb	424	+
chr22	17108306	17109820	uc002zly.5	0	+	chr22	17109651	17109678	(GGA)n	15	+
chr22	17108306	17109820	uc062bfr.1	0	+	chr22	17109651	17109678	(GGA)n	15	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120913	17120929	(TGCTGC)n	15	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120320	17120347	(GTG)n	15	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120596	17120630	(GCA)n	15	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120739	17120793	GA-rich	22	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120929	17120970	(GCC)n	19	+
chr22	17119390	17121127	uc002zmb.3	0	-	chr22	17120970	17120993	(TGCTGC)n	15	+
chr22	17534743	17534747	uc062bgi.1	0	+	chr22	17534499	17534802	AluY	2037	-
chr22	17550242	17550333	uc062bgm.1	0	+	chr22	17550132	17550323	AluSx1	1189	-

A check of the first record shows that the exon indeed does overlap with the matching repeat. (In fact the exon is entirely contained in the repeat. If we wanted to find all repeats like this we could use the Filter or Select tools.)

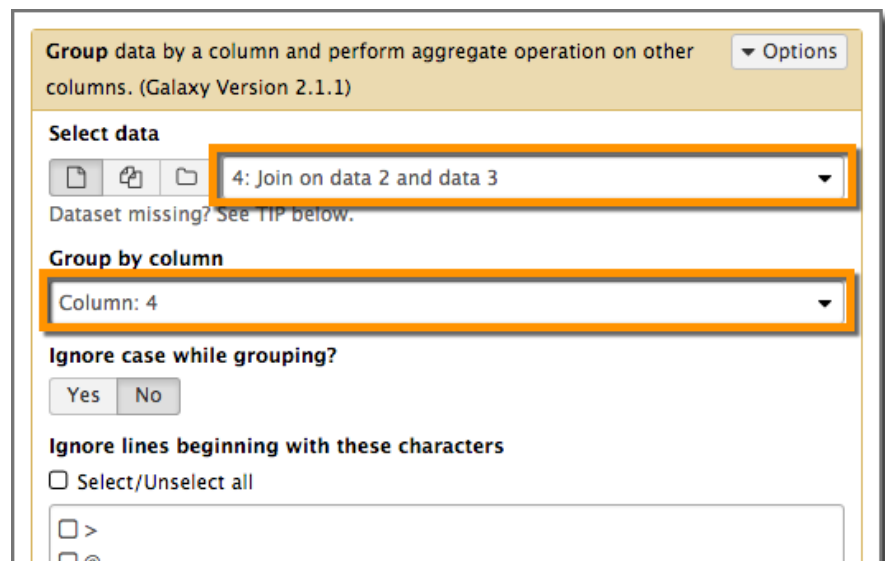
## Counting overlaps

Now that we know which *exons* and repeats overlap, let's count the number repeats that overlap with each *transcript*. Um, how? *We are going to use the **Group** tool and take advantage of the fact that each exon from a given transcript has the transcript as its name.* Group is a simple but very powerful tool.

Group takes a single dataset as input, and you specify which column in that dataset will be used to form *groups*. Each value that column has becomes a different group in the output dataset. In addition you can include summary statistics for each group in the output. The statistics can be per record (like counting), or on any of the columns in the input dataset.

We'll use group to count how many times each transcript name occurs in the joined dataset. That will tell us how many overlapping repeats each transcript has.

**Group** can be found in the **Join, Subtract and Group** toolbox. Set the input dataset to results of the join operation (should already be selected, and then set **Group by column** to the column containing the transcript name, **Column 4**:



Group data by a column and perform aggregate operation on other columns. (Galaxy Version 2.1.1) Options

**Select data**

4: Join on data 2 and data 3

Dataset missing? See TIP below.

**Group by column**

Column: 4

**Ignore case while grouping?**

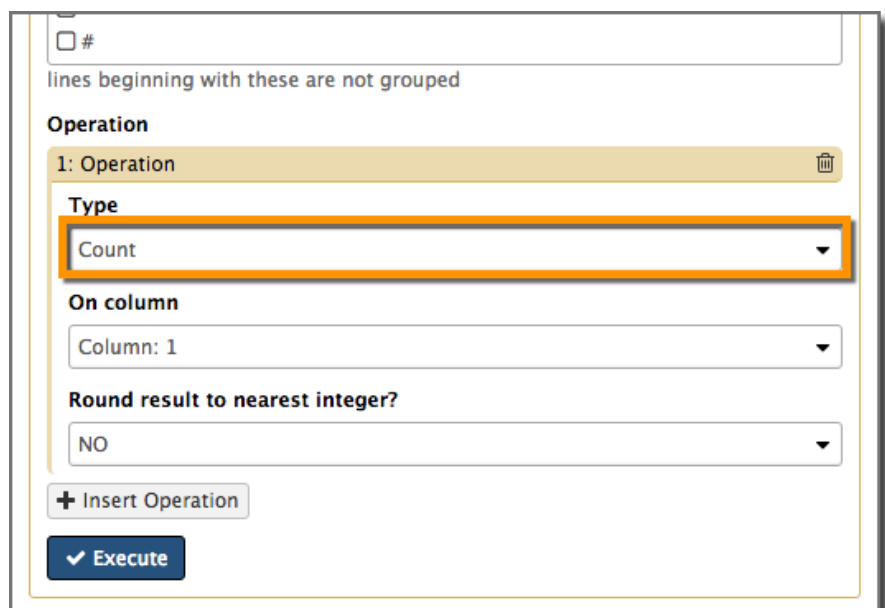
**Ignore lines beginning with these characters**

☐ Select/Unselect all

☐ >

☐ @

At the bottom of the **Group** form, click **+ Insert Operation**. Each inserted operation adds a summary statistic column to the output dataset. Set **Type** to **Count**. Count is record wide operation. All it does is count the number of input records that mapped to each output group. Therefore, it does not matter what column you select. Click **Execute**.



☐ #

lines beginning with these are not grouped

**Operation**

1: Operation trash icon

**Type**

Count

**On column**

Column: 1

**Round result to nearest integer?**

NO



The group tool takes the 900+ record input dataset (the joined exons and repeats) and condenses it down to 600+ *groups*, one for each transcript that has 1 or more overlapping repeats. The first column is the name of the transcript; the second is the number of repeats that overlap with that transcript.

1	2
uc002zly.5	2
uc002zmb.3	6
uc002zmx.5	1
uc002zmx.4	1
uc002zmy.5	1
uc002zmz.4	1
uc002zng.5	3
uc002zny.4	1
uc002zou.4	1
uc002zoy.5	1
uc002zoz.6	2
uc002zpf.2	1
uc002zpi.4	2
uc002zpk.2	1
uc002zpm.3	1
uc002zpo.3	1
uc002zqa.2	4
uc002zqb.4	2
uc002zqc.4	2
uc002zqg.4	1
uc002zqh.4	1
uc002zqi.4	1
uc002zqu.4	1

History
search datasets
Unnamed history
5 shown
3.84 MB
5: Group on data 4
628 lines
format: tabular, database: hg38
--Group by c4: count[c1]
1 2
uc002zly.5 2
4: Join on data 2 and data 3
911 regions
format: interval, database: hg38
1.Chrom 2.Start 3.End 4.Name
chr22 11065973 11066015 uc062bdq.1

Now if we sort it we'll have the transcripts on chromosome 22 that have the most overlapping repeats:

Open the **Filter and Sort** toolbox and *select Sort*. Sort the dataset by counts first, and then alphabetically by transcript name second. *Rename* the dataset to something with **Transcripts** and **RepeatMasker** in it.

1	2
uc002zmb.3	6
uc062bsq.1	6
uc062bss.1	6
uc062fon.2	6
uc003apg.4	5
uc003bcj.3	5
uc003bck.3	5
uc062bsr.1	5
uc062fom.2	5
uc002zqa.2	4
uc002zsd.5	4
uc002zsp.4	4
uc002zsq.4	4
uc002zsr.4	4
uc002zuy.5	4



## Transcripts and Genes

At this point we could declare victory! We have identified all the transcripts on chromosome 22 that have overlapping repeats and we have identified which have the most. Many ontology tools, *including the one we are going to use*, also know about transcript IDs

However, 1) the tool we are going to use does not know about UCSC transcript IDs. (It does know about RefSeq and possibly Ensembl transcript IDs though.) 2) Even if our tool of choice did know about UCSC transcript IDs it doesn't handle the case where more than one transcript from a gene is submitted. In this case it counts each transcript as an occurrence of the corresponding gene and results in a lot of *super* enriched functions.

We need to translate these transcript IDs to gene symbols. There are several online resources that can help us map UCSC transcript IDs to gene symbols. [Ensembl's BioMart server](#) is one way (see [this tutorial](#) for how). Another is to use UCSC to get this information. We'll use UCSC here.

Open the **Get Data** toolbox and **select UCSC Main Table Browser**. Reset the **group** to **Genes and Gene Predictions**. This presents a very familiar looking page. Right now it has the same settings we used to get the transcripts at the beginning.

There are a bunch of options (described in the [Table Browser User's Guide](#)) that we haven't touched yet. Click on the **describe table schema** button. This describes the data in the currently selected table and includes and links to related tables at UCSC as well. Poke around some and then *hit* the **back button** in your browser.

Table Browser

Use this program to retrieve the data associated with a track in text format, to calculate intersections between sequence covered by a track. For help in using this application see [Using the Table Browser](#) for a detailed [User's Guide](#) for general information and sample queries, and the OpenHelix Table Browser [tutorial](#) for software features and usage. For more complex queries, you may want to use [Galaxy](#) or our [public MySQL](#) biological function of your set through annotation enrichments, send the data to [GREAT](#). Send data to computational tools. Refer to the [Credits](#) page for the list of contributors and usage restrictions associated downloaded in their entirety from the [Sequence and Annotation Downloads](#) page.

clade: Mammal genome: Human assembly: Dec. 2013 (GRCh38/hg38)

group: Genes and Gene Predictions track: GENCODE v24

table: knownGene

region: genome position chr22:1-50818468

identifiers (names/accessions):

filter: create

intersection: create

correlation: create

output format: selected fields from primary and related tables

output file: (leave blank to keep output in browser)

file type returned: plain text gzip compressed

get output summary/statistics

Change output format to **selected fields from primary and related tables** and click **get output**.

By selecting this output format we are telling UCSC that we want to determine which data about genes is returned to Galaxy. The form shows all the attributes of the knownGene table, along with an explanation of each, and all the attributes of the (pre-)linked kgXref table. This is followed by a list of additional related tables that can also be added to the query.

For our purposes knownGene and kgXref have everything we need.

Our goal here is to get the gene symbol for our transcripts, and that appears to be included in the kgXref table.

Just to be safe, *select* any column in the two tables that has to do with **names or IDs**:

Once all 13 items have been checked, *click done with selections*. We are now requesting that a 13 column dataset be sent from UCSC to Galaxy.

### Select Fields from hg38.knownGene

<input checked="" type="checkbox"/>	name	Name of gene
<input type="checkbox"/>	chrom	Reference sequence chromosome or scaffold
<input type="checkbox"/>	strand	+ or - for strand
<input type="checkbox"/>	txStart	Transcription start position
<input type="checkbox"/>	txEnd	Transcription end position
<input type="checkbox"/>	cdsStart	Coding region start
<input type="checkbox"/>	cdsEnd	Coding region end
<input type="checkbox"/>	exonCount	Number of exons
<input type="checkbox"/>	exonStarts	Exon start positions
<input type="checkbox"/>	exonEnds	Exon end positions
<input checked="" type="checkbox"/>	proteinID	UniProt display ID, UniProt accession, or RefSeq protein ID
<input checked="" type="checkbox"/>	alignID	Unique identifier (GENCODE transcript ID for GENCODE Basic)

done with selections
cancel
check all
clear all

### hg38.kgXref fields

<input checked="" type="checkbox"/>	kgID	Known Gene ID
<input checked="" type="checkbox"/>	mRNA	mRNA ID
<input checked="" type="checkbox"/>	spID	UniProt protein Accession number
<input checked="" type="checkbox"/>	spDisplayID	UniProt display ID
<input checked="" type="checkbox"/>	geneSymbol	Gene Symbol
<input checked="" type="checkbox"/>	refseq	RefSeq ID
<input checked="" type="checkbox"/>	protAcc	NCBI protein Accession number
<input checked="" type="checkbox"/>	description	Description
<input checked="" type="checkbox"/>	rfamAcc	Rfam accession number
<input checked="" type="checkbox"/>	tRnaName	Name from the tRNA track

check all
clear all

### Linked Tables

<input type="checkbox"/>	gc	gcPart
--------------------------	----	--------

*Click Send query to Galaxy* to make this happen.

And what arrives is in fact a very wide 13 column dataset with all the IDs and names we requested. Two columns are of particular interest to us: Column 1 which contains the transcript ID, and Column 8 which contains the gene symbol.

*Rename* the dataset something more meaningful.

1	8
#hg38.knownGene.name	hg38.kgXref.geneSymbol
uc062bdo.1	U2
uc062bdp.1	CU459211.1
uc062bdq.1	CU104787.1
uc062bdr.1	BAGES
uc062bds.1	5_8S_rRNA
uc062bdt.1	AC137488.1
uc062bdu.1	AC137488.2
uc062bdv.1	CU013544.1
uc062bdw.1	CT867976.1
uc062bdx.1	CT867977.1
uc062bdy.1	CT978678.1
uc062bdz.1	CU459202.1
uc062bea.1	AC116618.1
uc062beb.1	CU463998.1
uc062bec.1	CU463998.3
uc062bed.1	CU463998.2
uc062bee.1	U6

To get the gene symbol in our latest dataset associated with our transcripts and our scores, we use the **Join** command in the **Join, Subtract and Group** toolbox. This join tool, like the one in the Operate on Genomic Intervals toolbox, joins two related records into one record. However, this join brings together records based on identity, rather than on position.

In this case join the two column dataset containing Transcript names and counts with the 13 column dataset we just got from UCSC. We are joining the datasets where they have a common Transcript name, which is in column 1 in both datasets.

*Execute* the join.

Join two Datasets side by side on a specified field (Galaxy Version 2.0.2) Options

Join

6: Transcripts w # RepeatMasker overlaps, chr22

using column

Column: 1

with

7: Transcript IDs and Xrefs, chr22

and column

Column: 1

Keep lines of first input that do not join with second input

No

Keep lines of first input that are incomplete

No

Fill empty columns

No

Execute

The resulting dataset has a record in it for each transcript that has overlapping repeats. There are 15 columns of information in it, include the overlapping repeat count, the transcript name, and the gene symbol the transcript is associated with.

### Which overlap count?

As mentioned previously, some genes will have multiple transcripts that overlap with repeats. This means we may have multiple overlap counts to pick from. Which one should we pick? I'll argue that we should pick the transcript with the most overlapping repeats and use that count for the gene.

(We could also use the average overlap count across all transcripts, or do something more sophisticated that takes into account things like how long a transcript is.)

To do this, *open* the **Join, Subtract and Group** toolbox and *select* **Group**. We want to create a group for each gene symbol. After the join the gene symbol is in column 10, so set the **Group by column** to **Column 10**. Add a single operation, **Maximum** on **Column 2**, the overlap score.

**Group data by a column and perform aggregate operation on other columns. (Galaxy Version 2.1.1)** Options

**Select data**

8: Join two Datasets on data 7 and data 6

Dataset missing? See TIP below.

**Group by column**

Column: 10

**Ignore case while grouping?**

☐ Yes ☒ No

**Ignore lines beginning with these characters**

☐ Select/Unselect all

☐ >

☐ @

**Operation**

1: Operation trash

**Type**

Maximum

**On column**

Column: 2

**Round result to nearest integer?**

NO

+ Insert Operation

✓ Execute

And *click* **Execute**.

And the result shows ~230 genes on chromosome 22 that have one or more overlapping RepeatMasker identified repeats.

Sort the dataset to get the high scorers and the top, and you'll see:

This history is available in on <https://test.galaxyproject.org/> [here](#).

1	2
CECR6	6
SCARF2	6
SHANK3	6
MYH9	5
TCF20	5
AC007326.1	4
BAIAP2L2	4
CACNA1I	4
CRELD2	4
DRICH1	4
MAPK8IP2	4
MED15	4
RIMBP3	4
RIMBP3C	4
SLIT2	4

## Deep thoughts 1

Woohoo! It only took 3 trips to UCSC, and 7 processing steps in Galaxy to identify genes on chromosome 22 that have the most overlapping RepeatMasker identified repeats!

OK, that seems like a lot for such a simple question. Well it is and it isn't.

It isn't a lot because each of those steps was simple and easy to grasp. Galaxy's basic tool set is like that. Each one does one task. Once you get the hang of them it's trivial to string them together.

It is a lot because when you are working on step 1 it's hard to know what step 7 will be. We also used our ability to see into the future (that's handy) to avoid several pitfalls. In reality, answering this question during your first visit to Galaxy would result in a lot of false starts and backtracking.

However, now that you've done it once, you might already see an easier way to answer this question...

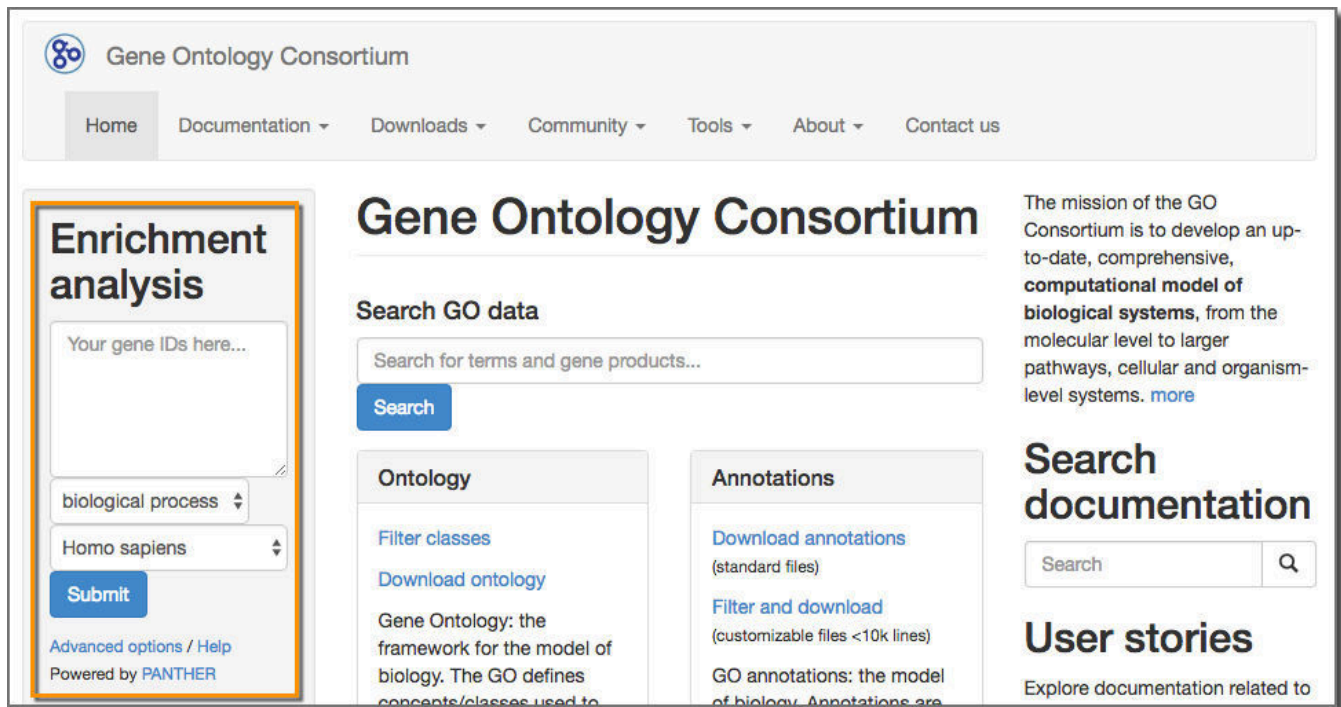
## Any patterns here? (1)

We now have a gene list. What does it mean?

This whole conference is about ontologies and we are now going to use the **power of ontologies** in, well, a fairly blunt way. We are going to look for GO term enrichment using the produced list. We'll experiment with different cutoffs and use Galaxy to analyze what we get back from GO.

The tool we'll use is the Enrichment Analysis tool on the Gene Ontology home page at [geneontology.org](http://geneontology.org). You paste a gene list in the box and it tells you which GO terms are under or over-represented in the provided sample. Our hope is that some patterns will emerge in the gene list our analysis has produced.





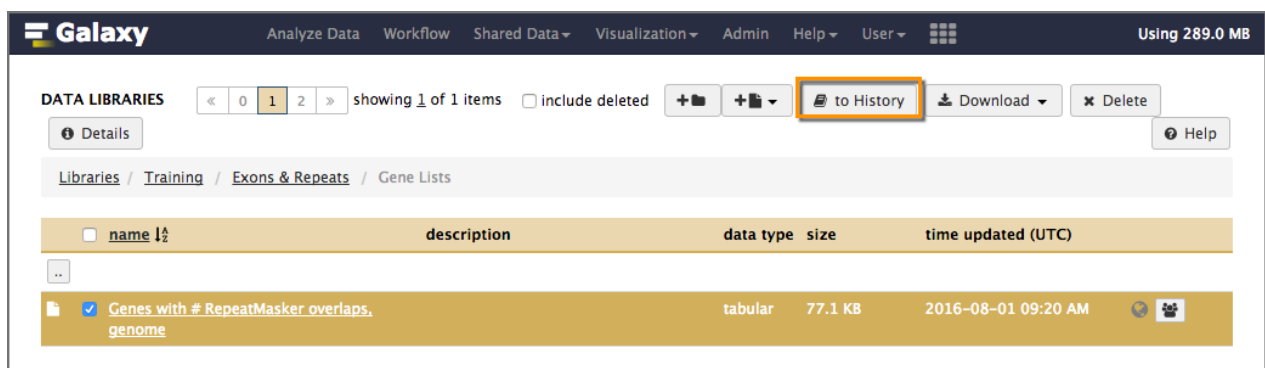
To do this, *poke* our final dataset **in the eye** and then *cut and paste* the list into the **Enrichment analysis box** at GO, and then ....

## Except don't (1)

It turns out that our gene list from chromosome 22 doesn't cover enough of the genome to produce even a potentially interesting enrichment analysis. To get that let's use a gene list produced from an analysis that examined the entire genome.

## Shared Data

To get the gene list for the entire genome, *click* on **Shared Data** in the top menu, and *select* **Data Libraries**. Then *select* **Training, Exons & Repeats**, and then **Gene Lists**. *Select* **Genes with # RepeatMasker overlaps, genome**. Import it to a new history.





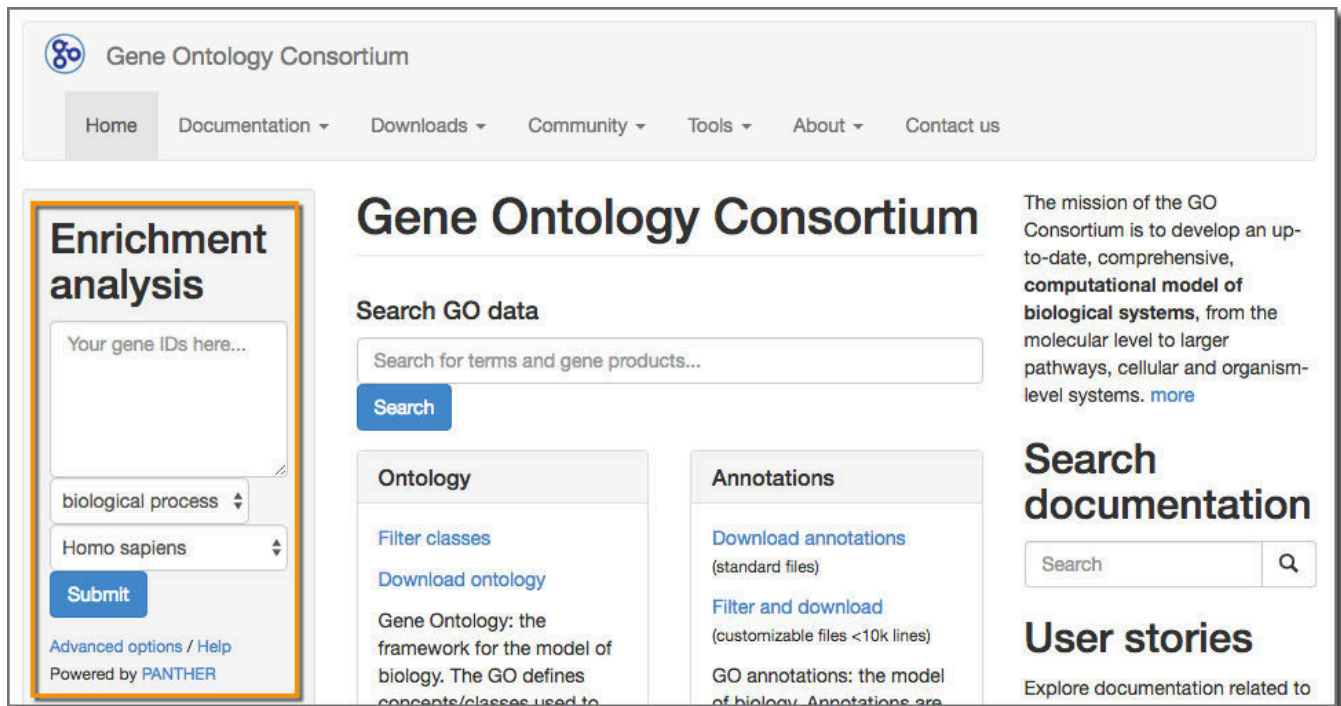
Note: The history that produced this dataset is available in **Shared Data** → **Histories** → **ICBO Gene-RepeatMasker Overlaps, genome** ([here](#)).

## Any patterns here? (2)

We now have a gene list. What does it mean?

This whole conference is about ontologies and we are now going to use the **power of ontologies** in, well, a fairly blunt way. We are going to look for GO term enrichment using the produced list. We'll experiment with different cutoffs and use Galaxy to analyze what we get back from GO.

The tool we'll use is the Enrichment Analysis tool on the Gene Ontology home page at [geneontology.org](http://geneontology.org). You paste a gene list in the box and it tells you which GO terms are under or over-represented in the provided sample. Our hope is that some patterns will emerge in the gene list our analysis has produced.



The screenshot shows the Gene Ontology Consortium website. The header includes the logo and navigation links: Home, Documentation, Downloads, Community, Tools, About, and Contact us. The main content area features the 'Gene Ontology Consortium' title and a 'Search GO data' section with a search bar and a 'Search' button. Below this are sections for 'Ontology' and 'Annotations'. On the left, a sidebar titled 'Enrichment analysis' is highlighted with an orange border. It contains a text input field for 'Your gene IDs here...', a dropdown menu for 'biological process', a dropdown menu for 'Homo sapiens', and a 'Submit' button. At the bottom of the sidebar are links for 'Advanced options / Help' and 'Powered by PANTHER'. On the right side of the main content area, there is a 'Search documentation' section with a search bar and a magnifying glass icon, and a 'User stories' section with the text 'Explore documentation related to'.

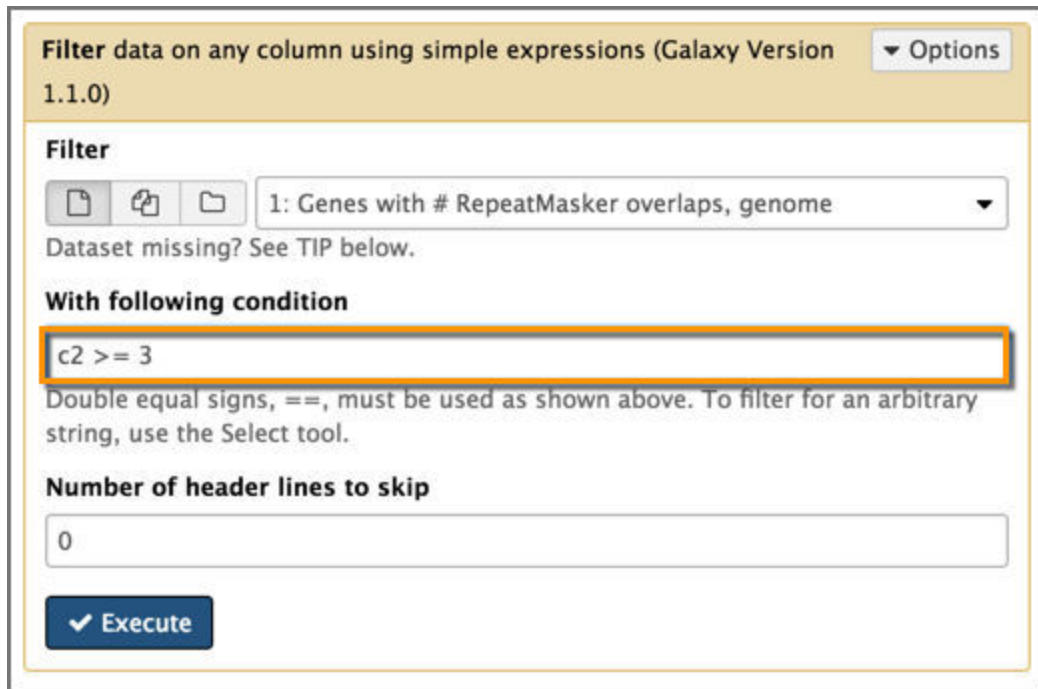
To do this, *poke* our final dataset **in the eye** and then *cut and paste* the list into the **Enrichment analysis box** at GO, and then ....

## Except don't (2)

I have no idea if geneontology.org can support a roomful of people all simultaneously submitting 9000+ genes for enrichment analysis. **So, please don't.**

So, what *can* we do? First, let's reduce the size of that list. Let's create versions of the gene list that have only genes with 3 or more overlaps, 4 or more overlaps, 5 or more, and finally 6 or more overlaps.

To do this, use the **Filter** tool in the **Filter and Sort** toolbox.



Specify a condition that records must satisfy to be kept. Use the shorthand "c1", "c2" for columns 1 and 2.

Here we are specifying that the # of repeats must be greater than or equal to 3.

Name the resulting dataset something with "3+" in the name.

Now create datasets that have only 4+, 5+ and 6+ overlapping RepeatMasker repeats.

### Any patterns here? (3)

OK, now we can post the smaller gene lists to the Enrichment Analysis tool on the Gene Ontology home page at [geneontology.org](http://geneontology.org). The gene lists, with the counts, can be directly pasted into the Enrichment Analysis box on the GO home page. The tool just ignores the counts in the second column. You can also use the Cut tool to remove the second column before pasting the lists to GO.

Running enrichment analysis on the 3+ dataset results in many terms being enriched, while running with 6+ dataset results in only a few terms being enriched:

[illegible]

Results ?		
Mapped IDs:	Reference list <a href="#">20972</a> out of 20972	upload_1 <a href="#">106</a> out of 108
Unmapped IDs:	<a href="#">0</a>	<a href="#">2</a>
Multiple mapping information:	0	<a href="#">2</a>
Export results		
Displaying only results with P<0.05; <a href="#">click here to display all results</a>		
	Homo sapiens (REF)	upload_1 (▼ Hierarchy NEW! ?)
GO biological process complete	#	# expected Fold Enrichment +/- P value
<a href="#">vocalization behavior</a>	<a href="#">14</a>	<a href="#">5</a> .07 69.35 + 1.13E-04
<a href="#">chromatin remodeling</a>	<a href="#">141</a>	<a href="#">10</a> .73 13.77 + 3.24E-05
↳ <a href="#">chromatin organization</a>	<a href="#">636</a>	<a href="#">20</a> 3.28 6.11 + 7.83E-07
↳ <a href="#">chromosome organization</a>	<a href="#">984</a>	<a href="#">21</a> 5.07 4.14 + 2.35E-04
<a href="#">chromatin assembly or disassembly</a>	<a href="#">149</a>	<a href="#">8</a> .77 10.43 + 9.80E-03
<a href="#">chromatin modification</a>	<a href="#">289</a>	<a href="#">15</a> 1.49 10.08 + 2.59E-07
<a href="#">transcription, DNA-templated</a>	<a href="#">2558</a>	<a href="#">31</a> 13.17 2.35 + 2.81E-02
↳ <a href="#">nucleic acid-templated transcription</a>	<a href="#">2559</a>	<a href="#">31</a> 13.18 2.35 + 2.83E-02
<a href="#">regulation of cellular macromolecule biosynthetic process</a>	<a href="#">3938</a>	<a href="#">41</a> 20.28 2.02 + 2.02E-02
↳ <a href="#">regulation of macromolecule biosynthetic process</a>	<a href="#">4053</a>	<a href="#">41</a> 20.87 1.96 + 4.25E-02
Unclassified	<a href="#">4136</a>	<a href="#">15</a> 21.30 .70 - 0.00E00

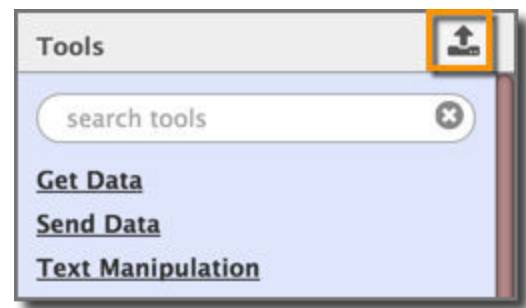
The 4+ and 5+ results fall in between those two extremes.

Now, upload the reports to your Galaxy instance...

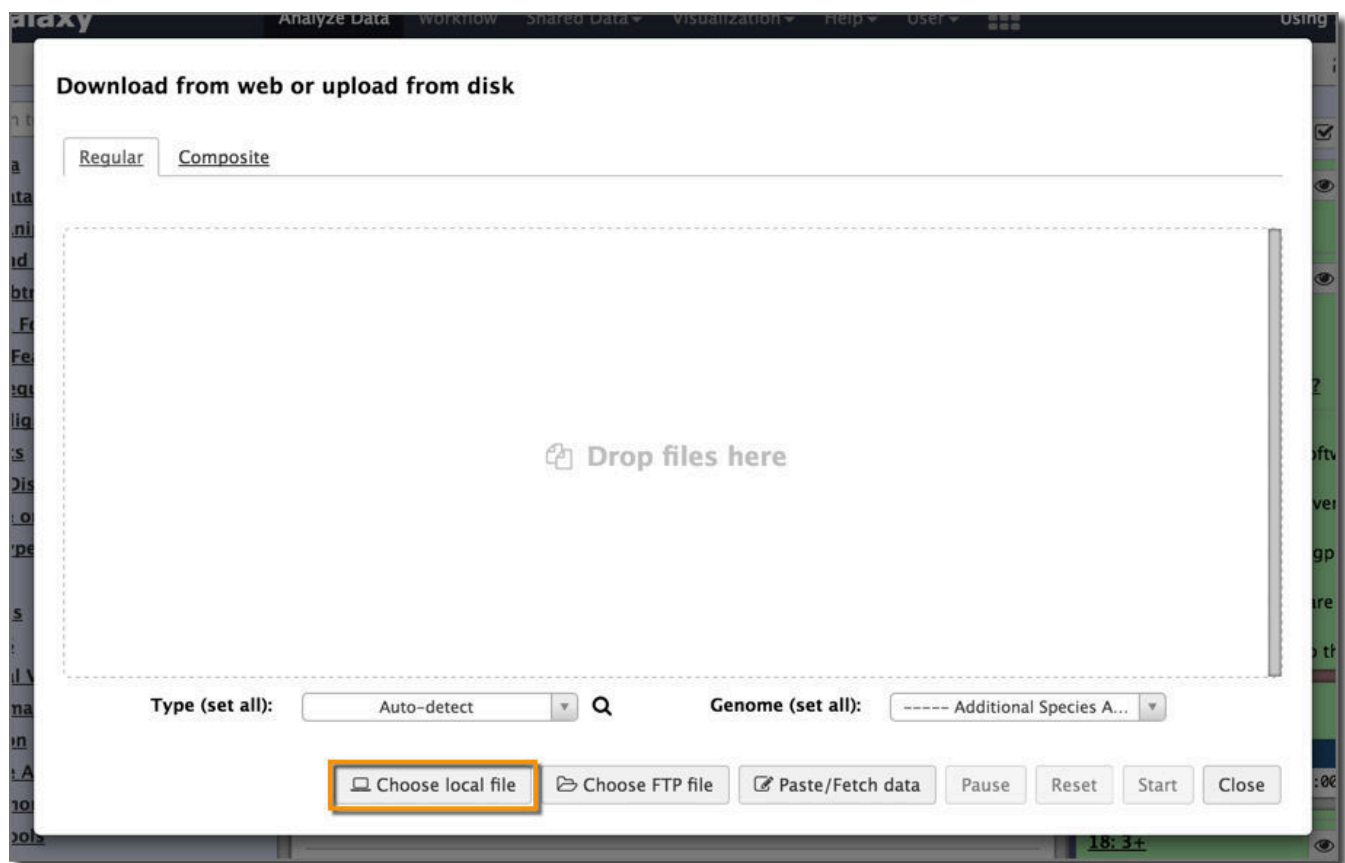
## Upload data

So far we've seen how to get data from UCSC. Now let's see how to upload data from your local computer. First, *click* on the **upload/download icon** at the top of the tool panel.

This opens a general purpose data import window that can be used to upload data from local disk (what we will do), by fetching it from a URL, using this server's FTP server (useful for large files), and by directly typing in (short) datasets.

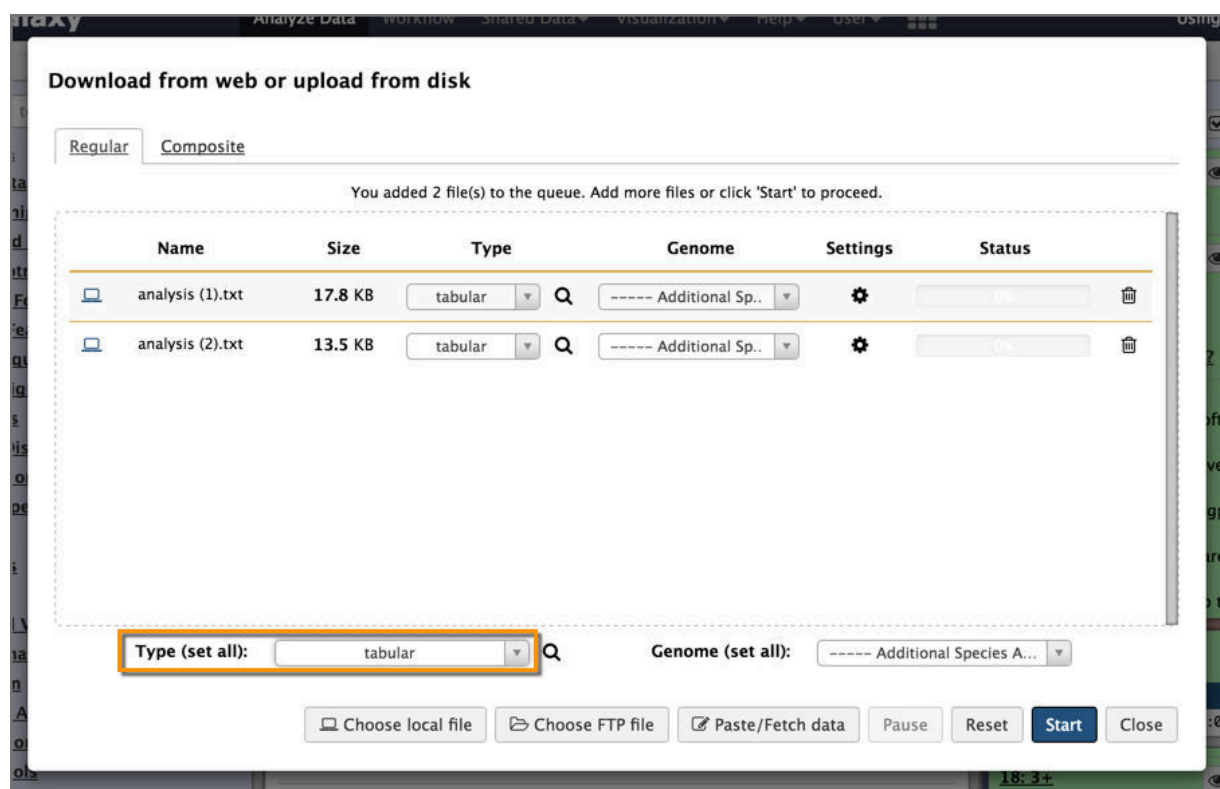


Click on **Choose local file**.



Select your downloaded reports from GO (they'll be named something like **Analysis.txt** and **Analysis (1).txt** (on Macs)).

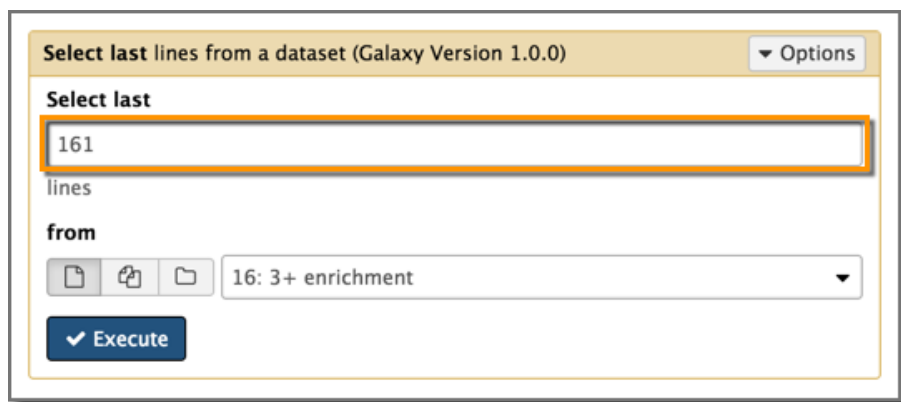
The two files will appear in your window. Now set the file **Type** for both to **tabular**:



Then *click* **Start** and then **Close**. *Rename* the two uploaded datasets to have **3+** and **6+** and **enrichment** in their names.

The files are actually a mixture of plain text (the first 5 lines), and tab delimited (everything else). To enable Galaxy tools to fully work with the datasets, we need to remove the five leading lines of both datasets.

Use the **Select Last** tool in the **Text manipulation** toolbox. For the 3+ dataset select the last 161 lines (at the time of this writing) and for the 6+ dataset select the 12 lines (at the time of this writing):



*Rename* the resulting datasets to have **3+** and **6+** in their names.

## Is one cutoff better than the others?

What terms are identified as enriched for the both large gene list (3+ overlaps) and the small (6+ overlaps)? What terms are in the 6+ list, but aren't in the 3+ list?

To answer these questions, let's use the **Compare** tool in the **Join, Subtract and Group** toolbox.

*Select* the **6+** enriched terms as the first dataset, and the **3+** as the second. We are going to compare the two datasets on GO term, and this occurs in **Column 1** in both. *Set To find to Matching Rows of 1st dataset*. This will return every term that exists in both the large 3+ dataset and the small 6+ dataset. Then rerun the query but *change To find to Non Matching Rows of 1st dataset*.

The results:

Shared terms

1
chromatin modification (GO:0016568)
chromatin organization (GO:0006325)
chromatin remodeling (GO:0006338)
chromosome organization (GO:0051276)
GO biological process complete
nucleic acid-templated transcription (GO:0097659)
regulation of cellular macromolecule biosynthetic process (GO:2000112)
regulation of macromolecule biosynthetic process (GO:0010556)
transcription, DNA-templated (GO:0006351)
Unclassified (UNCLASSIFIED)

Terms in 6+ dataset, but missing from 3+ dataset:

1	2	3	4	5	6	7
chromatin assembly or disassembly (GO:0006333)	149	8	.77	+	10.43	9.80E-03
vocalization behavior (GO:0071625)	14	5	.07	+	69.35	1.13E-04

Our nearly 70 fold enriched “vocalization behavior” disappears entirely between the 6+ and 3+ datasets. What does that mean (and at which cutoff does it disappear)?

Compare using the 4+ and 5+ datasets as well. Is there a “sweet spot” for setting a cutoff? Does requiring more stringent p-values have any impact?



## Comparing repeat calling software

We used repeats identified by RepeatMasker in our analysis, but there are other repeat callers out there. UCSC gives us access to calls from at least two other repeat callers.

[Tandem Repeats Finder](#) (TRF): Identifies only simple tandem repeats. (RepeatMasker identifies these and longer transposon repeats as well.)

[WindowMasker](#) (WM): Repeats are identified entirely based on what pattern recur in the analyzed genome. WindowMasker does not use any prior knowledge to identify repeats. This is in contrast to RepeatMasker which is driven by a database of known repeats.

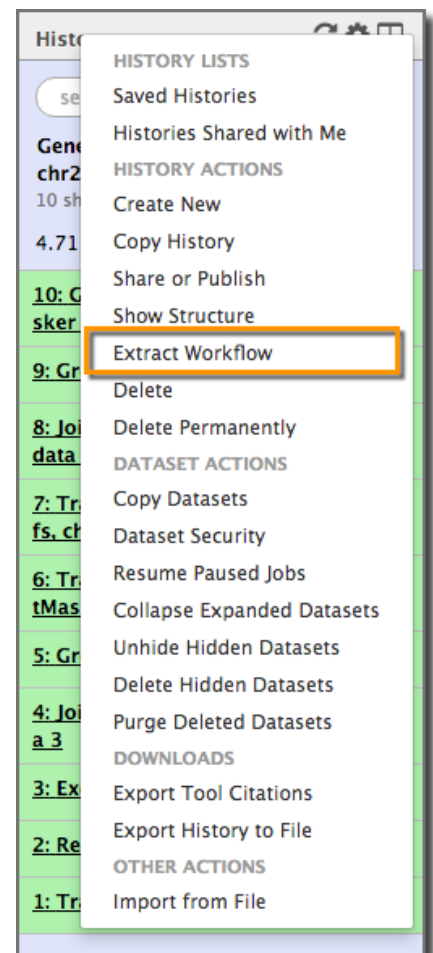
In addition to using Galaxy to examine the impacts of different cutoff policies, we can also use it to compare different repeat callers and how robust our methods are with different callers. To do this we can run the same analysis steps as with the RepeatMasker dataset, but this time using the TRF or WM

## But first, Workflows

We already have a defined series of steps to take to run this analysis. Do we have to rerun each step again when we use the TRF and again when we use WM datasets? *No*.

We can create a Galaxy *workflow* from the analysis we have already done, and then reuse that workflow when processing the TRF and WM datasets.

To create a workflow from a history, *click* on the **cog icon** at the top of the history panel and *select* **Extract Workflow**.



The following list contains each tool that was run to create the datasets in your current history. Please select those that you wish to include in the workflow.

Tools which cannot be run interactively and thus cannot be incorporated into a workflow will be shown in gray.

**Workflow name**

Gene-Repeats Overlap

Create Workflow

Check all

Uncheck all

Tool	History items created
UCSC Main <i>This tool cannot be used in workflows</i>	<b>1: Transcripts, chr22</b> <input checked="" type="checkbox"/> Treat as input dataset
UCSC Main <i>This tool cannot be used in workflows</i>	<b>2: RepeatMasker, chr22</b> <input checked="" type="checkbox"/> Treat as input dataset
Gene BED To Exon/Intron/Codon BED <input checked="" type="checkbox"/> Include "Gene BED To Exon/Intron/Codon BED" in workflow	<b>3: Exons, chr22</b>
Join <input checked="" type="checkbox"/> Include "Join" in workflow	<b>4: Join on data 2 and data 3</b>
Group <input checked="" type="checkbox"/> Include "Group" in workflow	<b>5: Group on data 4</b>
Sort <input checked="" type="checkbox"/> Include "Sort" in workflow	<b>6: Transcripts w # RepeatMasker overlaps, chr22</b>
UCSC Main <i>This tool cannot be used in workflows</i>	<b>7: Transcript IDs and Xrefs, chr22</b> <input checked="" type="checkbox"/> Treat as input dataset
Join two Datasets <input checked="" type="checkbox"/> Include "Join two Datasets" in workflow	<b>8: Join two Datasets on data 7 and data 6</b>
Group <input checked="" type="checkbox"/> Include "Group" in workflow	<b>9: Group on data 8</b>
Sort <input checked="" type="checkbox"/> Include "Sort" in workflow	<b>10: Genes w # RepeatMasker overlaps, chr22</b>

This launches the workflow extraction page, which shows all the input datasets and analyses steps in the history we are extracting the workflow from.

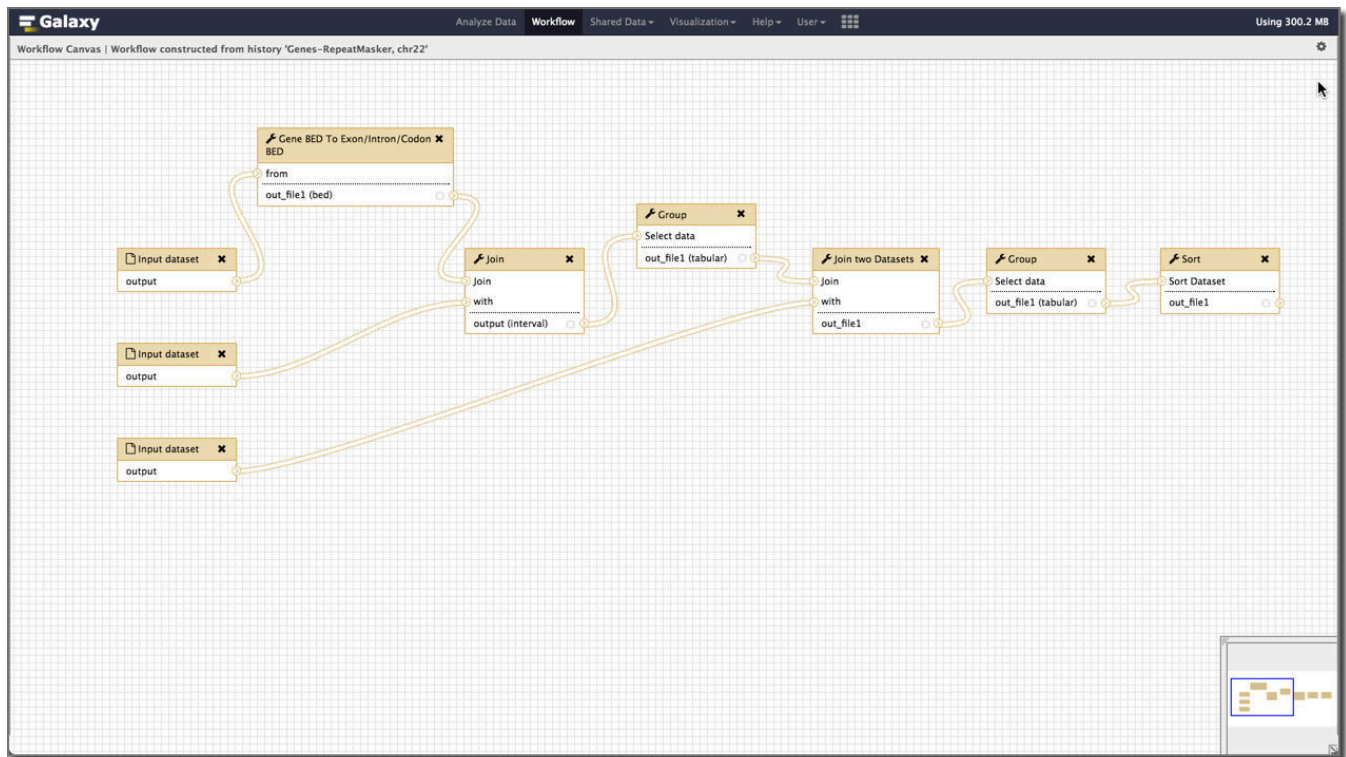
At this point, if there were mis-steps in the analysis we can omit those steps from the workflow by unchecking them here.

Once we have settled on the required datasets and steps (and in this case, it is all of them), **click Create Workflow.**

A workflow is created and you are offered the choice of editing or running it. Since a workflow is intended to be reused, it is worthwhile to edit it and add descriptions to at least the input datasets and to the final output datasets.

This workflow, titled "Transcripts-Repeats Overlap to Genes", is available on <https://test.galaxyproject.org> [here](#).

## The workflow editor view of our analysis



To run the analysis again, but using different repeat callers, create a new history, get the two transcript datasets (the original BED file, and the kgXref file) into it, and then go to UCSC and get the SimpleRepeats and WM + DustFM repeats datasets. Rerun the example with both.

Histories that produce the overlap counts for the whole genome using RepeatMasker, SimpleRepeats and WindowMasker are on the <https://test.galaxyproject.org/> here:

- [RepeatMasker history](#)
- [SimpleRepeats history](#)
- [WindowMasker history](#)

What do we learn? Mostly we learn that SimpleRepeats is almost a proper subset of RepeatMasker repeats, while WindowMasker produces a much larger number of repeats that are poorly aligned with those identified by RepeatMasker. Here's a comparison of RepeatMasker and WindowMasker using the complete genome:

RepeatMasker				WindowMasker				# Common			
#overlaps	Genes	GO Terms		#overlaps	Genes	GO Terms		Genes	%*	Terms	%*
3+	1055	160		15+	1221	166		217	21	84	53
4+	425	109		20+	591	120		74	17	47	11
5+	202	74		30+	185	51		22	12	2	4
6+	109	11		35+	118	43		7	6	2	18

\* Percentage of the smaller of the two sets.

These comparisons were done in Galaxy using tools used in this tutorial.