Training Materials

- Ensembl training materials are protected by a CC BY license
- http://creativecommons.org/licenses/by/4.0/
- If you wish to re-use these materials, please credit Ensembl for their creation
- If you use Ensembl for your work, please cite our papers
- http://www.ensembl.org/info/about/publications.html

Ensembl REST API Workshop

Astrid Gall
Ensembl Outreach Officer

Reykjavik, 2 October 2019

EMBL-EBI

Ensembl

This Workshop

- Ensembl and the gene model
- What is REST
- Ensembl REST API features
- Fetching a single endpoint
- Decoding the response
- Linking endpoints together
- POST endpoints
- Rate limiting

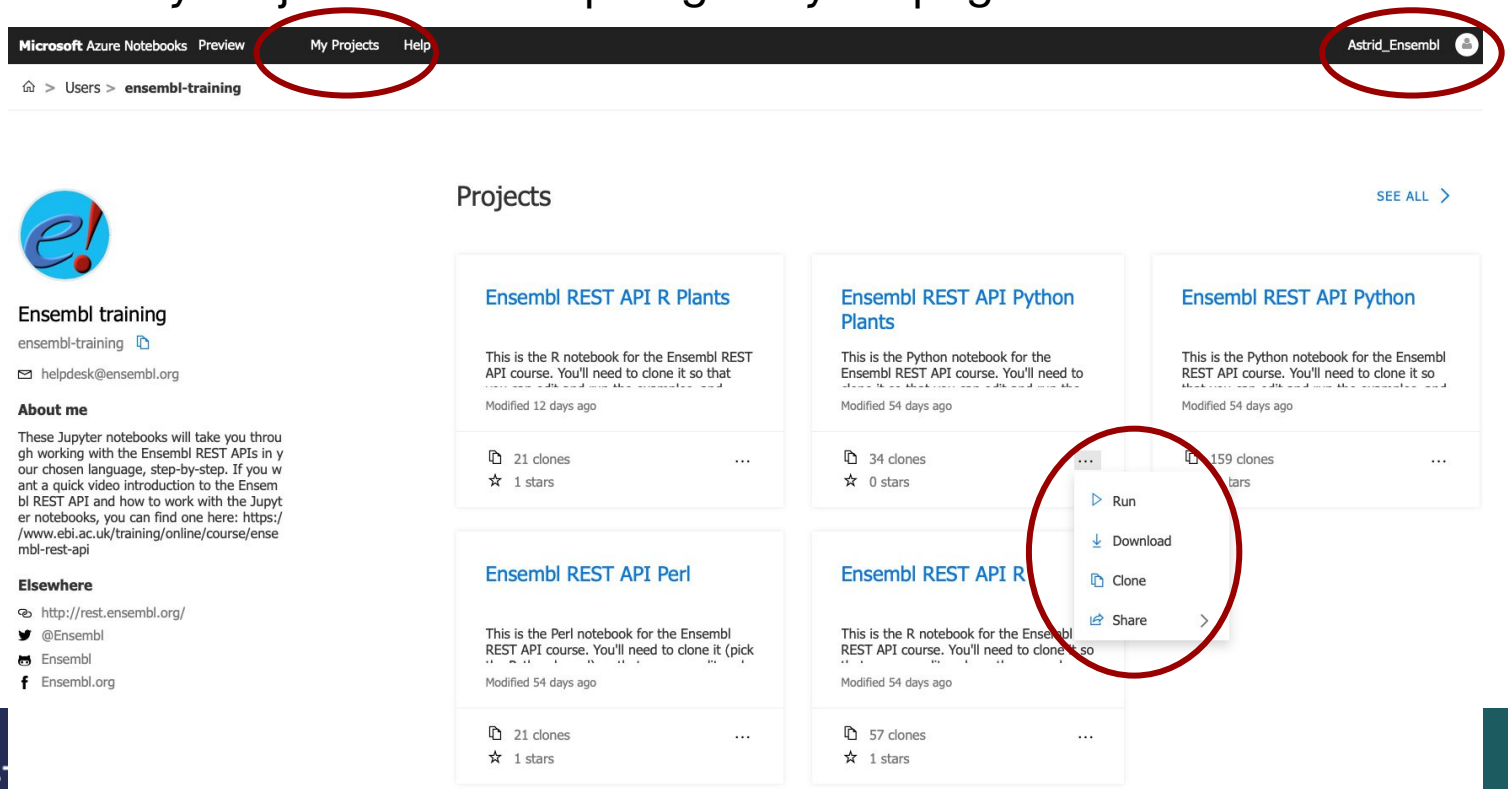
Course Materials

http://training.ensembl.org/events/

- Presentation
- Notebooks in Python, R and Perl (human) and in Python and R (plants)
  - Use whichever notebook you feel comfortable with
  - You will need to clone it with your Microsoft Account
- When we demo the example answers we will use Python only

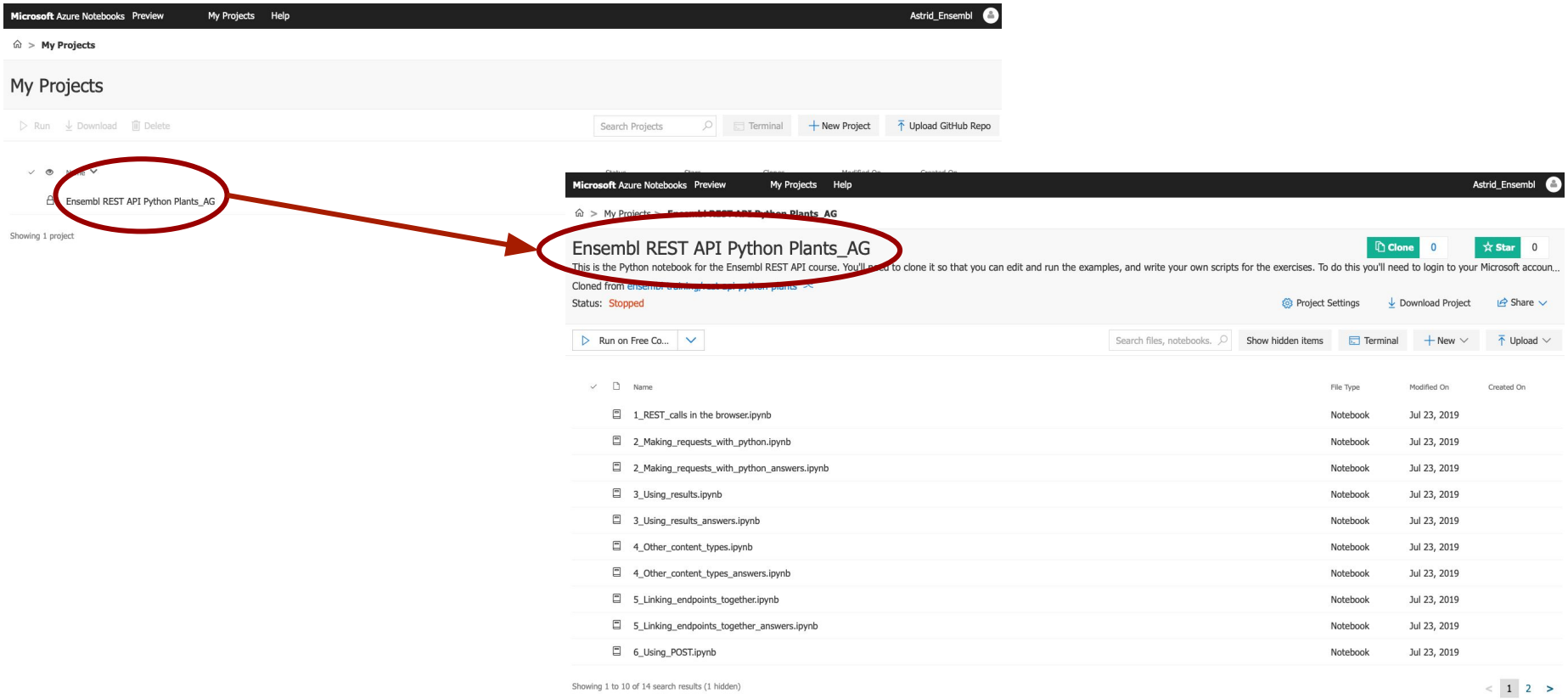
Cloning the Notebook

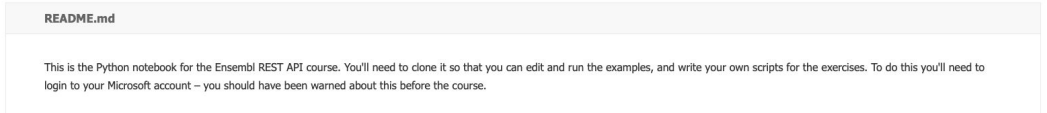
- Go to http://training.ensembl.org/events/
- Follow the ‘Notebooks’ link from today’s event.
- Log in to Microsoft Azure with your account.
- In Microsoft Azure, choose a notebook on the Ensembl training page, click on ‘...’ and select ‘Clone’.
- Click ‘My Projects’ at the top to go to your page.



Cloning the Notebook

- The cloned notebook appears under ‘My Projects’ on your page.
- Use the clone so that you can edit and run the examples.
- Click on it to go to the table of contents.

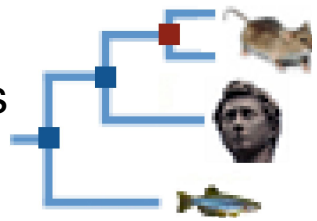


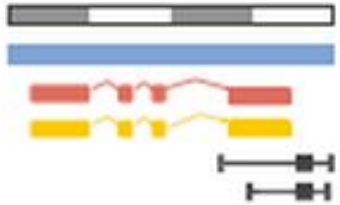



Questions?

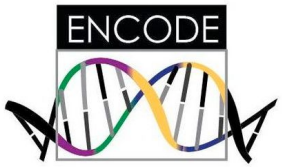
Ensembl Features

- Genomes and gene builds for >100 species
- Variation data
- Alignment, gene trees, homologues
- Regulatory build (ENCODE)



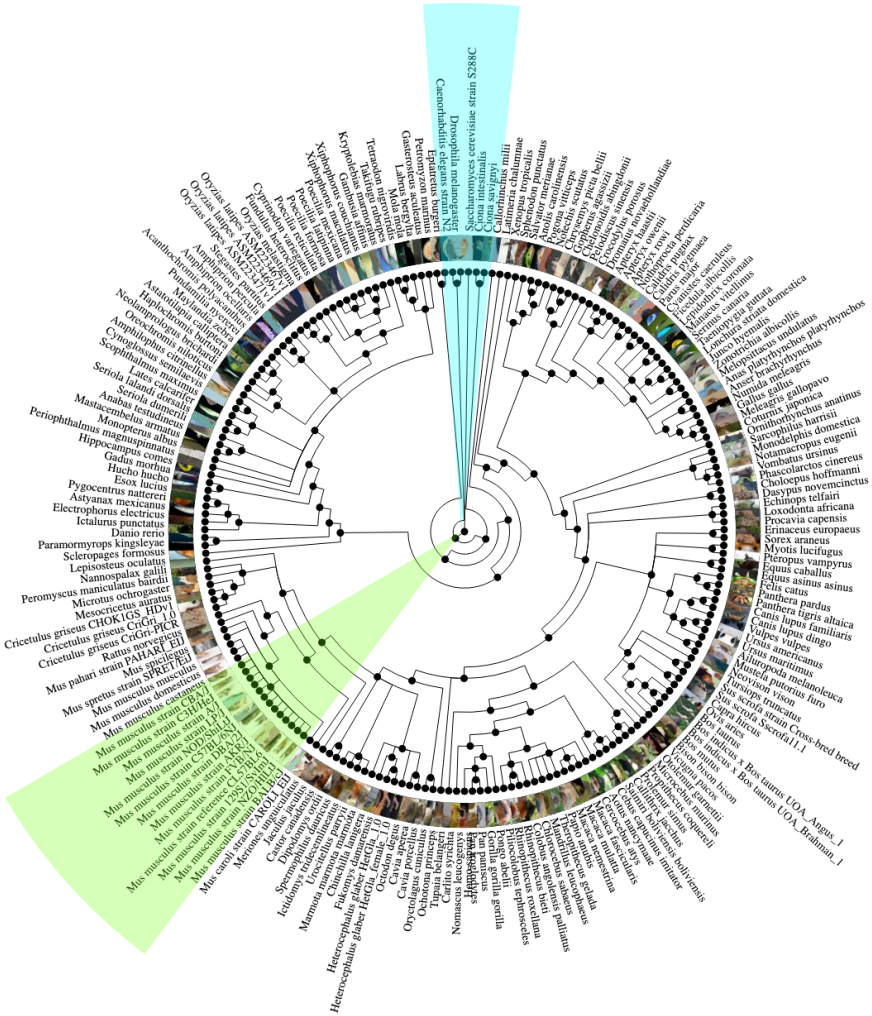


- BioMart (data export)
- Tools for data processing, e.g. VEP
- Display your own data
- Programmatic access via APIs
- Completely Open Source (FTP, GitHub)



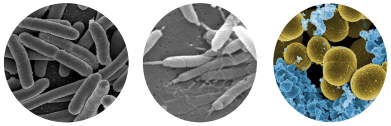
www.ensembl.org

Ensembl: Vertebrates and Model Organisms



www.ensembl.org

Ensembl Genomes: Other Taxa



Bacteria

www.ensembl.org

Vertebrates

Fungi

www.ensemblgenomes.org

Protists

Plants

Metazoa

www.ensemblgenomes.org

Ensembl and Ensembl Genomes

	Ensembl	Ensembl Genomes
Released	2000	2009
Species	Vertebrates (fly, worm and yeast as outgroups)	Non-vertebrates (protists, plants, fungi, metazoa, bacteria)
Annotation	By Ensembl	In collaboration with the scientific communities
Website	www.ensembl.org	www.ensemblgenomes.org
REST API	**rest.ensembl.org**	**rest.ensembl.org**

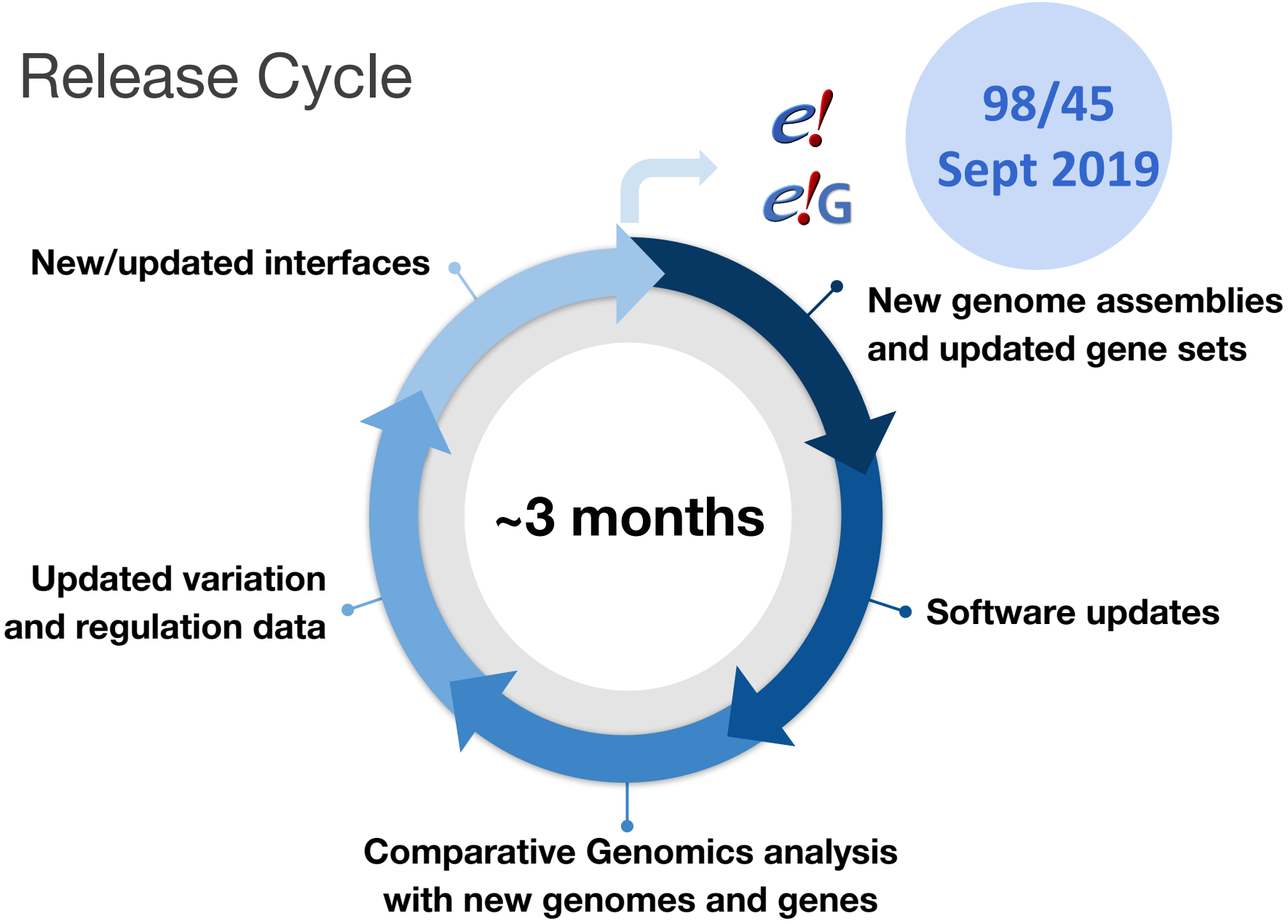
Human Genome Assemblies

- GRCh38 (aka hg38) 

  - No gaps. Many rare/private alleles replaced.
  - **rest.ensembl.org**
  - Software regularly updated
  - Data regularly updated
- GRCh37 (aka hg19) 

  - 250 gaps
  - **grch37.rest.ensembl.org**
  - Software regularly updated
  - Data only rarely updated

Release Cycle



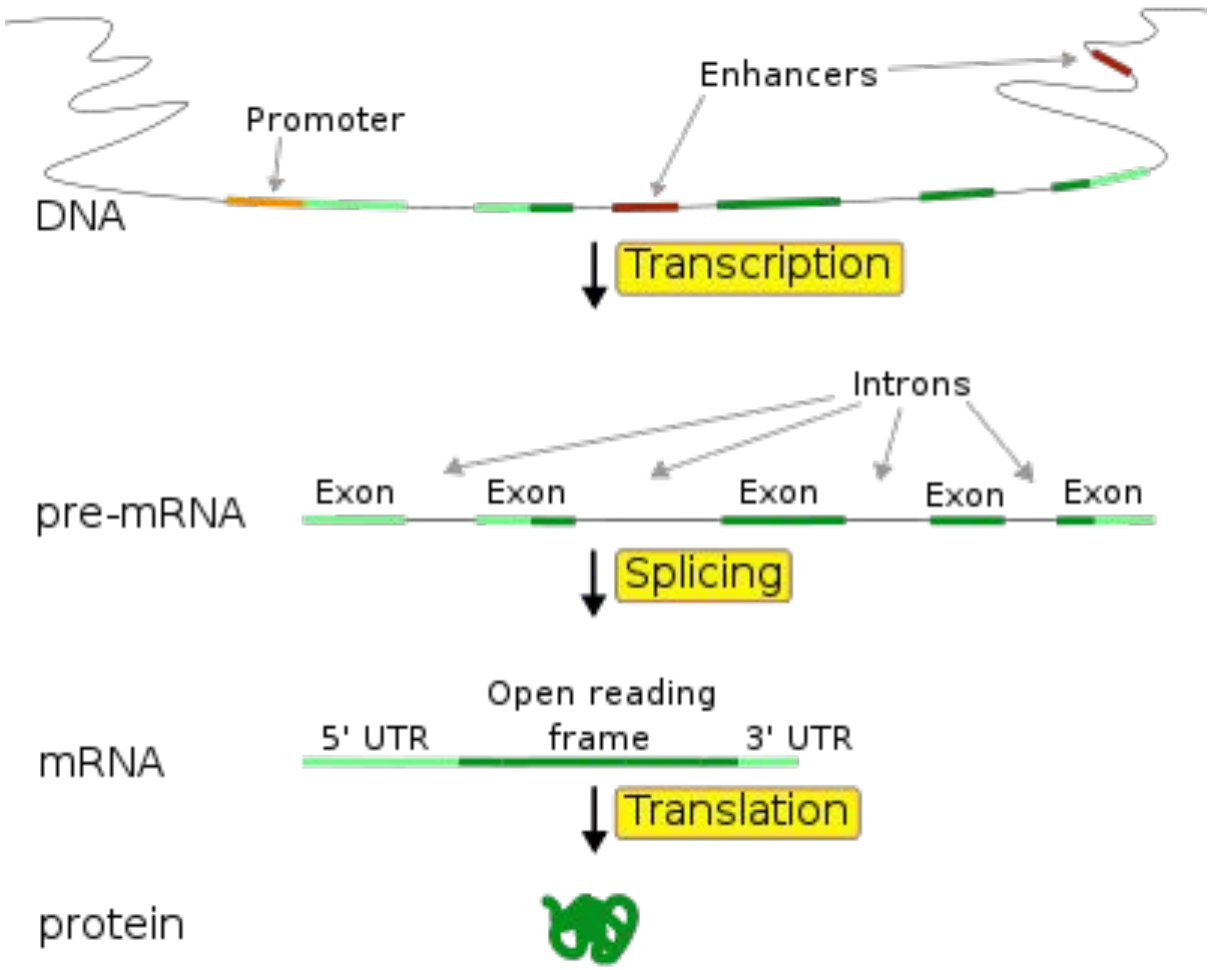
REST Archives

Starting with release 87, there are REST archives (GRCh38 only).

We will continue to provide REST archive services for up to five years, to match the Ensembl website archives.

http://e87.rest.ensembl.org

Ensembl Data Model

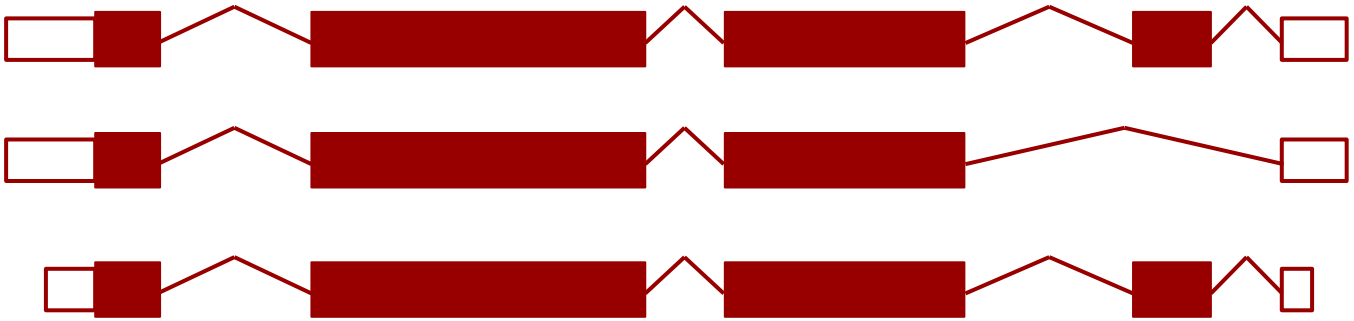


Ensembl Data Model

Primary feature types of Genes, Transcript and Exons.

A Gene is a set of alternatively spliced Transcripts.

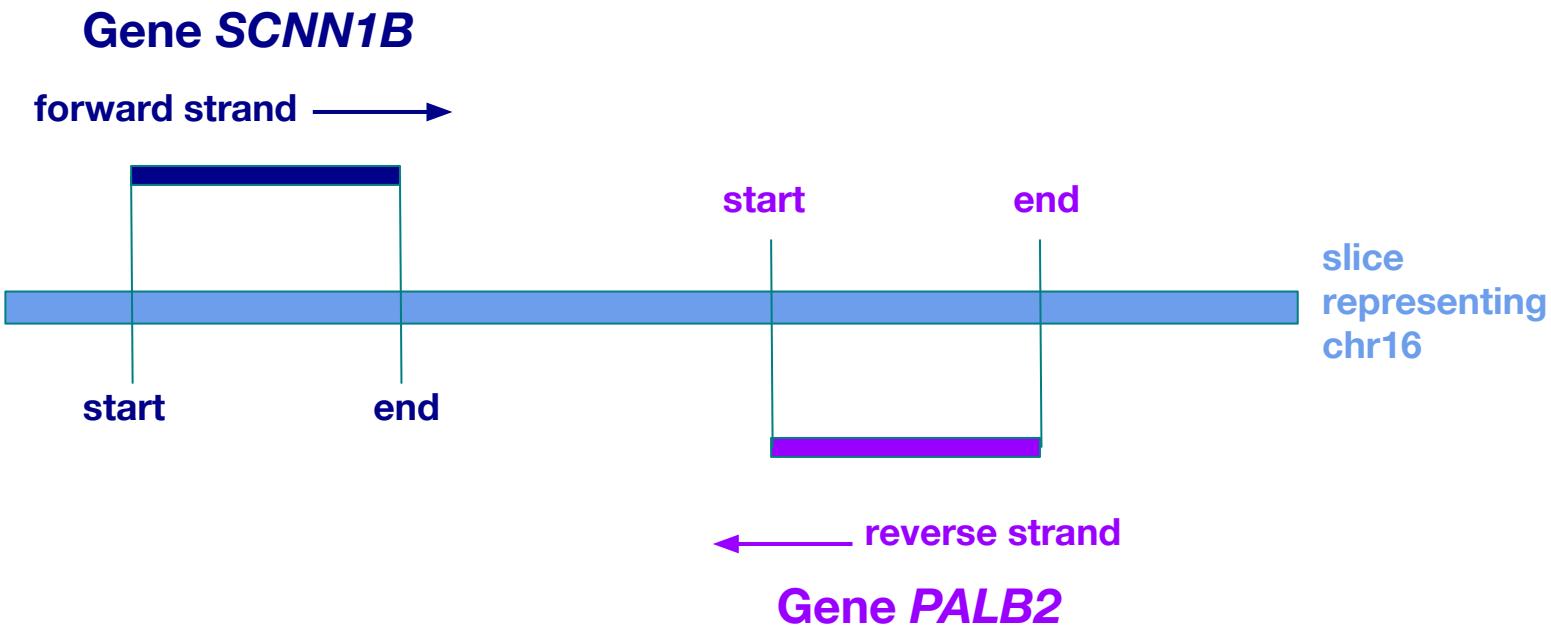
A Transcript is a set of Exons.



Features

Features have a defined location on the genome.

Start and end are always plotted on the forward strand. start < end



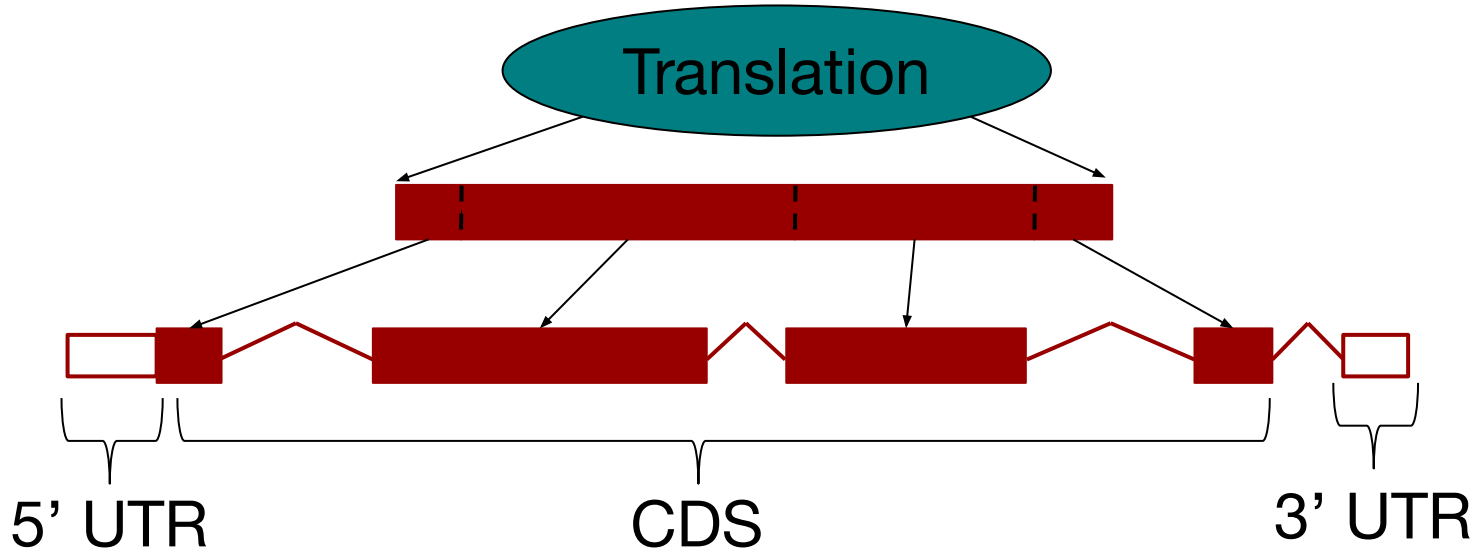
Ensembl Data Model

Translations are not features.

A Translation object defines the UTR and CDS of a Transcript.

Peptides are not stored in the database; they are computed on the fly using Transcript objects.

Not all transcripts have a translation (e.g. ncRNAs).

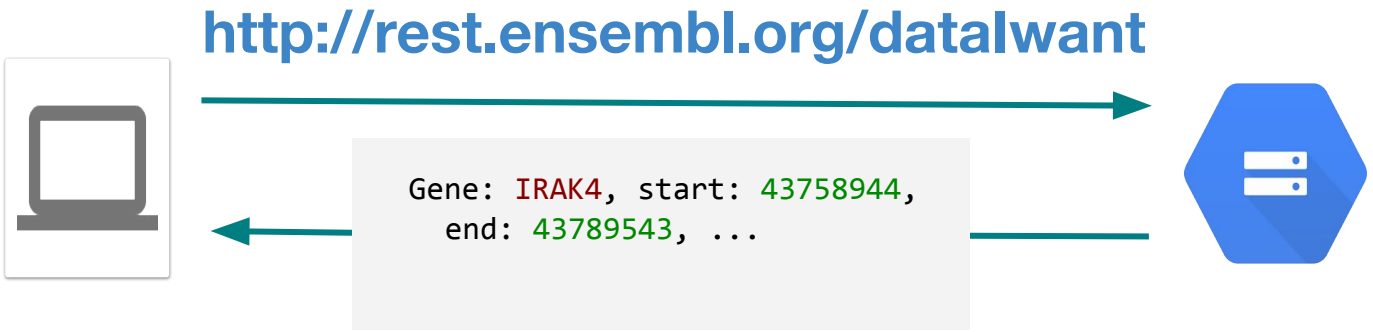


What is a REST API?

REpresentational State Transfer & Application Programming Interface.

It describes how one system can communicate state with another.

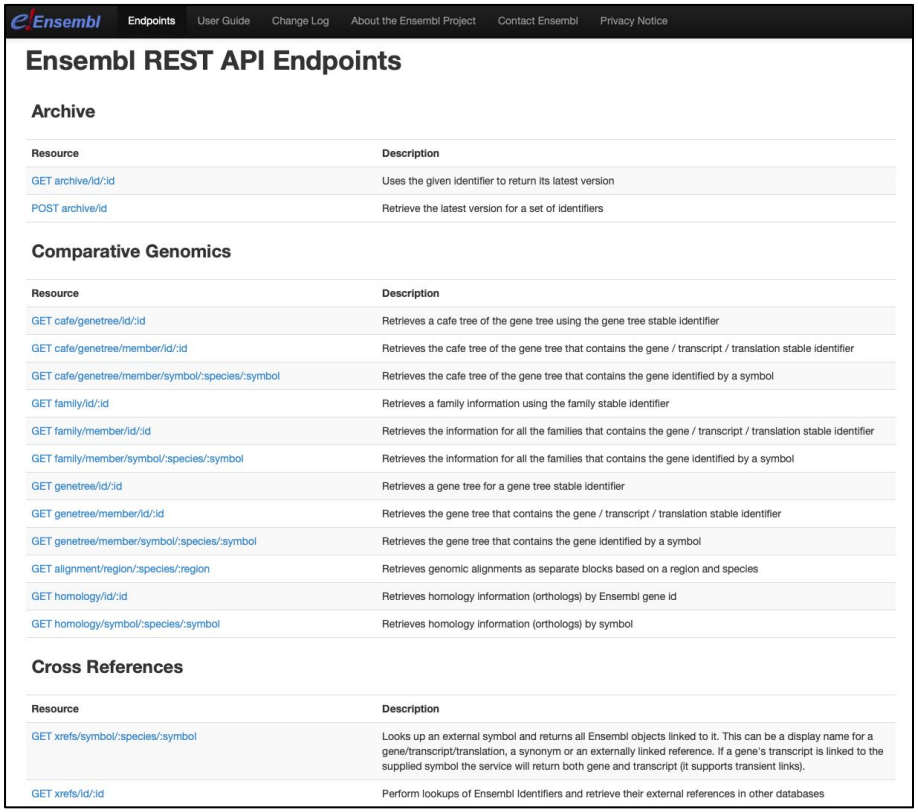
Typically over HTTP(S), providing a machine readable, language agnostic method to access remote data or services.



Ensembl REST API

- Language agnostic access to Ensembl datasets
- Only a fraction of the functionality of the Perl API is exposed

http://rest.ensembl.org



Ensembl | Endpoints | User Guide | Change Log | About the Ensembl Project | Contact Ensembl | Privacy Notice

Ensembl REST API Endpoints

Archive

Resource	Description
GET archive/id/:id	Uses the given identifier to return its latest version
POST archive/id	Retrieve the latest version for a set of identifiers

Comparative Genomics

Resource	Description
GET cafe/genetree/id/:id	Retrieves a cafe tree of the gene tree using the gene tree stable identifier
GET cafe/genetree/member/id/:id	Retrieves the cafe tree of the gene tree that contains the gene / transcript / translation stable identifier
GET cafe/genetree/member/symbol/:species/:symbol	Retrieves the cafe tree of the gene tree that contains the gene identified by a symbol
GET family/id/:id	Retrieves a family information using the family stable identifier
GET family/member/id/:id	Retrieves the information for all the families that contains the gene / transcript / translation stable identifier
GET family/member/symbol/:species/:symbol	Retrieves the information for all the families that contains the gene identified by a symbol
GET genetree/id/:id	Retrieves a gene tree for a gene tree stable identifier
GET genetree/member/id/:id	Retrieves the gene tree that contains the gene / transcript / translation stable identifier
GET genetree/member/symbol/:species/:symbol	Retrieves the gene tree that contains the gene identified by a symbol
GET alignment/region/:species/:region	Retrieves genomic alignments as separate blocks based on a region and species
GET homology/id/:id	Retrieves homology information (orthologs) by Ensembl gene id
GET homology/symbol/:species/:symbol	Retrieves homology information (orthologs) by symbol

Cross References

Resource	Description
GET xrefs/symbol/:species/:symbol	Looks up an external symbol and returns all Ensembl objects linked to it. This can be a display name for a gene/transcript/translation, a synonym or an externally linked reference. If a gene's transcript is linked to the supplied symbol the service will return both gene and transcript (it supports transient links).
GET xrefs/id/:id	Perform lookups of Ensembl Identifiers and retrieve their external references in other databases

What the Ensembl REST API is and is not

- HTTP access to Ensembl data
- Stable service
- Limited by network latency
- Read only
- Versioned with archives

- Not HATEOAS†, or fully RESTful*
- No mirrors
- Not an efficient data mining solution
- Incomplete coverage

† Hypermedia As The Engine Of Application State
* See lengthy debates about Roy Fielding's conception of REST

What is an Endpoint?

In REST, the resource typically refers to some object or set of objects that are exposed at an API endpoint.

```
/api/users/johnny
```

An endpoint by itself is just a reference to a uri that accepts web requests that may or may not be RESTful.

```
/services/service.asmx
```

https://stackoverflow.com/questions/30580562/what-is-the-difference-between-resource-and-endpoint

An endpoint is a particular output that you can get given a particular input.

It is a function that interacts with our database.

Endpoint Documentation

Full documentation of all endpoints is found at:

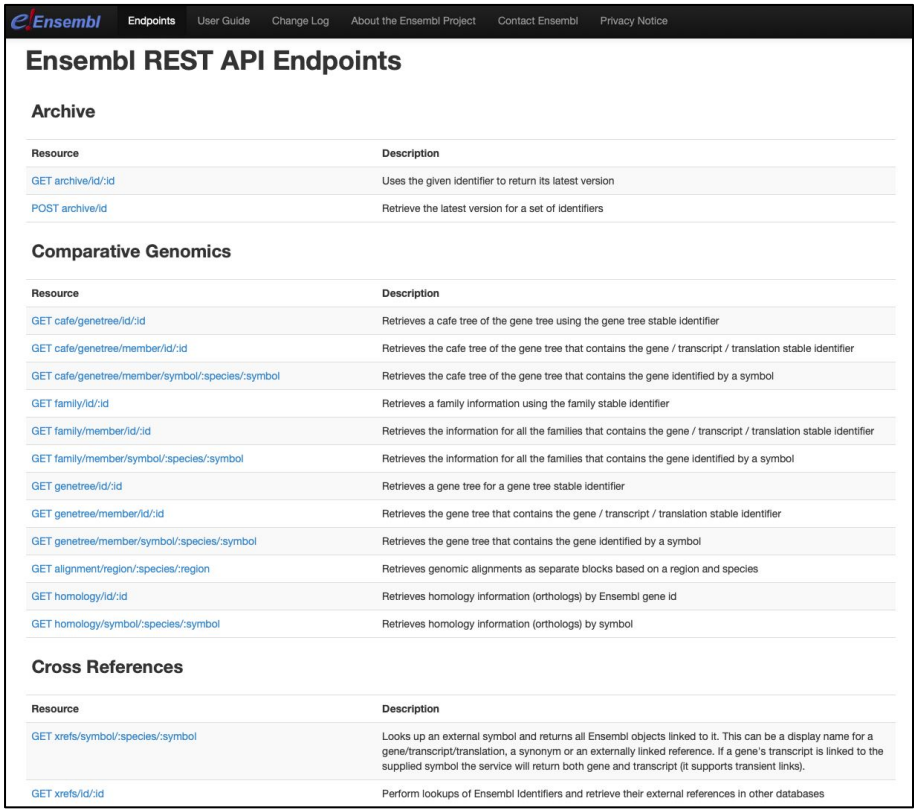
http://rest.ensembl.org

The documentation lists:

- All endpoints grouped by function
- The required parameters for each endpoint
- Optional parameters
- Example code for using the endpoints
- Example output

Functional Groupings

- Archive
- Comparative Genomics
- Cross References
- EQTL
- Information
- Linkage Disequilibrium
- Lookup
- Mapping
- Ontologies & Taxonomy
- Overlap
- Phenotype annotations
- Regulation
- Sequence
- Transcript Haplotypes
- VEP
- Variation
- Variation GA4GH



Tip:

Use Ctrl [Cmd] + F to search the page

Endpoint Documentation

You must include the id in the URL in this position

GET lookup/id/:id

Find the species and database for a single identifier e.g. gene, transcript, protein

Parameters

Required

Name	Type	Description	Default	Example Values
id	*String*	An Ensembl stable ID	-	*ENSG00000157764*

Optional

Name	Type	Description	Default	Example Values
callback	*String*	Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.	-	*randomlygeneratedname*
db_type	*String*	Restrict the search to a database other than the default. Useful if you need to use a DB other than core	-	*core otherfeatures*
expand	*Boolean(0,1)*	Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.	*0*	-

Resource Information

Methods	GET
Response formats	json xml jsonp

You can choose to include these in the URL in the format: **parameter=value**

Making a REST Call in your Web Browser

- The easiest way to make REST calls is to put URLs into a web browser
- It can be used as a quick look-up
- It can help you to test the URLs in your scripts to see:
  - If they work
  - If you have included the correct parameters
  - What the output looks like

Pinging the Database

Ping confirms that you have a connection to the database

http://rest.ensembl.org/info/ping?content-type=application/json


```
{
  ping: 1
}
```

Requesting a Gene by ID

http://rest.ensembl.org/lookup/id/ENSG00000157764?content-type=application/json


```
{
  "source": "ensembl_havana",
  "object_type": "Gene",
  "logic_name": "ensembl_havana_gene",
  "version": 12,
  "species": "homo_sapiens",
  "description": "B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC
Symbol;Acc:HGNC:1097]",
  "display_name": "BRAF",
  "assembly_name": "GRCh38",
  "biotype": "protein_coding",
  "end": 140924764,
  "seq_region_name": "7",
  "db_type": "core",
  "strand": -1,
  "id": "ENSG00000157764",
  "start": 140719327
}
```

Requesting more Info about a Gene by ID

http://rest.ensembl.org/lookup/id/ENSG00000157764?content-type=application/json;expand=1

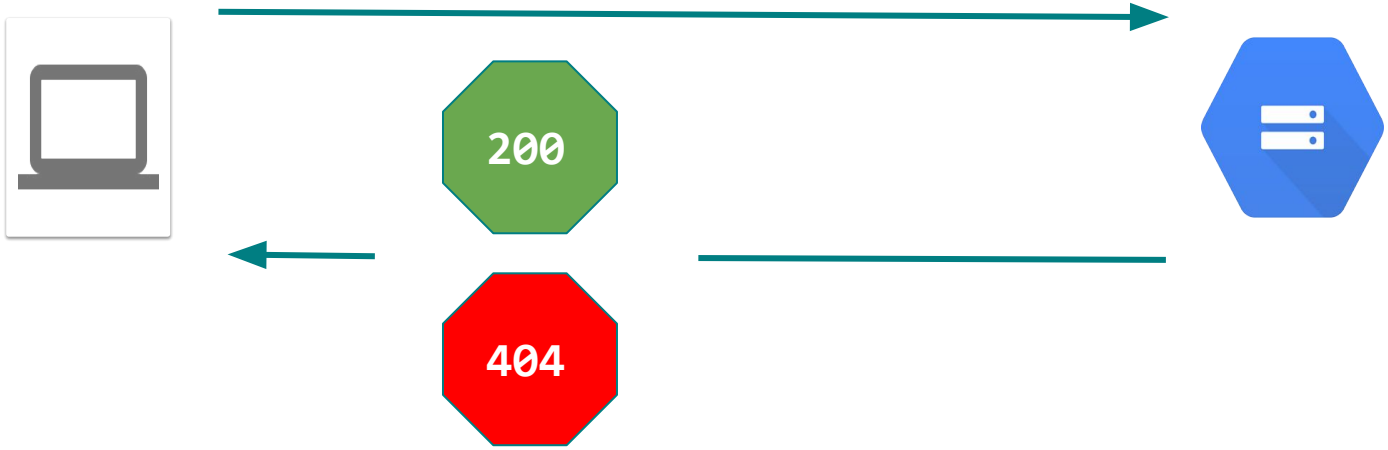
Add an optional parameter


```
{
 "assembly_name": "GRCh38",
 "db_type": "core",
 "Transcript": [
   {
     "assembly_name": "GRCh38",
     "db_type": "core",
     "is_canonical": 0,
     "logic_name": "havana",
     "seq_region_name": "7",
     "strand": -1,
     "start": 140719327,
     "species": "homo_sapiens",
     "version": 7,
     "end": 140924810,
     "source": "havana",
     "Exon": [
       {
         "strand": -1,
         ...
```

HTTP Status Codes

The server uses HTTP status codes to signal the request outcome

http://rest.ensembl.org/thisdoesntexist



HTTP Status Codes

Code	Name	Notes
200	OK	Request was a success. Only process data from the service when you receive this code
400	Bad Request	Occurs during exceptional circumstances such as the service is unable to find an ID. Check if the response Content-type or Accept was JSON. If so the JSON object is an exception hash with the message keyed under error
403	Forbidden	You are submitting far too many requests and have been temporarily forbidden access to the service. Wait and retry with a maximum of 15 requests per second.
404	Not Found	Indicates a badly formatted request. Check your URL

https://github.com/Ensembl/ensembl-rest/wiki/HTTP-Response-Codes

HTTP Status Codes (cont.)

Code	Name	Notes
408	Timeout	The request was not processed in time. Wait and retry later
429	Too Many Requests	You have been rate-limited; wait and retry. The headers `X-RateLimit-Reset`, `X-RateLimit-Limit` and `X-RateLimit-Remaining` will inform you of how long you have until your limit is reset and what that limit was. If you get this response and have not exceeded your limit then check if you have made too many requests per second.
503	Service Unavailable	The service is temporarily down; retry after a pause
418	I'm a teapot	An April Fools joke added in 1998, who said computer scientists don't have a sense of humour?

https://github.com/Ensembl/ensembl-rest/wiki/HTTP-Response-Codes


Questions?

Exercises 1

1. Find an endpoint which you can use to lookup information about a gene using its symbol.
2. Create a URL to find information about the gene *IRAK4* in human.
3. Expand your results to include information about transcripts.

Answers 1

1. GET lookup/symbol/:species/:symbol
2. http://rest.ensembl.org/lookup/symbol/homo_sapiens/IRAK4?content-type=application/json
3. http://rest.ensembl.org/lookup/symbol/homo_sapiens/IRAK4?content-type=application/json;expand=1

Scripting around REST API Calls

Scripting around calls allows you to:

- Output in your preferred format.
- Extract specific bits of data from your REST response.
- Link together calls for more complicated queries.
- Integrate your queries into a larger pipeline.

Language agnostic Access

- REST APIs are designed to be accessed using any programming language.
- Calls can be made and responses decoded within any script.
- We have examples in Python, R and Perl.

Example Code

Example Requests

/lookup/id/ENSG00000157764?content-type=application/json;expand=1

Example output | Perl | Python2 | Python3 | Ruby | Java | R | Curl | Wget

```perl
use strict;
use warnings;

use HTTP::Tiny;

my $http = HTTP::Tiny->new();

my $server = 'http://rest.ensembl.org';
my $ext = '/lookup/id/ENSG00000157764?expand=1';
my $response = $http->get($server.$ext, {
  headers => { 'Content-type' => 'application/json' }
});

die "Failed!\n" unless $response->{success};


use JSON;
use Data::Dumper;
if(length $response->{content}) {
```

Python Modules

- To make requests in Python, you will need the requests package:
  - https://pypi.org/project/requests/ (not needed for this course; this is all set up in your Python Notebook)
- To decode JSON, you will need the JSON package:
  - Should ship with standard Python installations
- To print JSON in an easy to read way, you will need pprint:
  - Should ship with standard Python installations

```
import requests, sys, json
from pprint import pprint
```

R Libraries

- To make requests in R, you will need the httr library
- To decode JSON, you will need the jsonlite package
- To get a more human readable format use the prettify function from the jsonlite package

```
library(httr)
library(jsonlite)
```

Perl Modules

- To make requests in Perl, you will need the HTTP:Tiny module
  - Should ship with standard Perl installations
- To decode JSON, you will need the JSON package:
  - **https://metacpan.org/pod/JSON** (not needed for this course, this is all set up in your Perl Notebook)
- To print JSON in an easy to read way, you will need Data::Dumper:
  - Should ship with standard Perl installations

```
use Data::Dumper;
use HTTP::Tiny;
use JSON;
```

Requesting a Gene by ID

http://rest.ensembl.org/lookup/id/ENSG00000157764?content-type=application/json


```
{
  "source": "ensembl_havana",
  "object_type": "Gene",
  "logic_name": "ensembl_havana_gene",
  "version": 12,
  "species": "homo_sapiens",
  "description": "B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC
Symbol;Acc:HGNC:1097]",
  "display_name": "BRAF",
  "assembly_name": "GRCh38",
  "biotype": "protein_coding",
  "end": 140924764,
  "seq_region_name": "7",
  "db_type": "core",
  "strand": -1,
  "id": "ENSG00000157764",
  "start": 140719327
}
```

Making a Request – Python

- Make a string of the server (you will use this multiple times)
- Make another string of the extension with all the parameters

```
import requests, sys

server = "http://rest.ensembl.org"
ext = "/lookup/id/ENSG00000157764?expand=1"

r = requests.get(server+ext, headers={ "Accept" : "application/json"})

pprint (r)
```

Making a Request – R

- Make a string of the server (you will use this multiple times)
- Make another string of the extension with all the parameters

```
library(httr)
library(jsonlite)

server <- "http://rest.ensembl.org"
ext <- "/lookup/id/ENSG00000157764"

r <- GET(paste(server, ext, sep = ""), accept("application/json"))

r
```

Making a Request – Perl

- Make a string of the server (you will use this multiple times)
- Make another string of the extension with all the parameters

```perl
use HTTP::Tiny;
use JSON;

my $http = HTTP::Tiny->new();
my $server = "http://rest.ensembl.org";
my $ext = "/lookup/id/ENSG00000157764";
my $headers = { 'Accept' => 'application/json' };

my $r = $http->get($server.$ext, {headers => $headers});

print $r;
```

Error Handling – Python

You should never assume that your request has worked.

```
import requests, sys

server = "http://rest.ensembl.org"
ext = "/lookup/id/ENSG00000157764?expand=1"

r = requests.get(server+ext, headers={ "Accept" : "application/json"})

if not r.ok:
    r.raise_for_status()
```

Check the response code returned by the server.

Error Handling – R

You should never assume that your request has worked.

```
library(httr)
library(jsonlite)

server <- "http://rest.ensembl.org"
ext <- "/lookup/id/ENSG00000157764"

r <- GET(paste(server, ext, sep = ""), content_type("application/json"))

r

stop_for_status(r)
```

Check the response code returned by the server.

Error Handling – Perl

You should never assume that your request has worked.

```
use HTTP::Tiny;
use JSON;

my $http = HTTP::Tiny->new();
my $server = "http://rest.ensembl.org";
my $ext = "/lookup/id/ENSG00000157764";
my $headers = { 'Content-Type' => 'application/json' };

my $r = $http->get($server.$ext, {headers => $headers});

die $r->{status}, "\n" unless $r->{success};
```

Check the response code returned by the server.

Content-type and Accept

HTTP allows the serving of different representations of a resource based on client preferences.

Content-type and Accept headers are how servers and clients negotiate what format they will communicate with.

text/html, text/plain, application/json, image/png, etc.

Content-type and Accept

Resource Information

Methods	GET
Response formats	fasta json seqxml text yaml jsonp

- The returned content-types can be specified in the header as accept (you will need to use content-type in URLs)
- Endpoint documentation pages list allowed content-types
- The wiki lists how you specify these

https://github.com/Ensembl/ensembl-rest/wiki/Output-formats

Decoding the Response – Python

- In most cases you will be using JSON formatted responses
- Most languages have JSON parsers that return the data as a structure
- In Python pretty print (pprint) will give you a more human readable format

```
decoded = r.json()

pprint (decoded)
```

Decoding the Response – R

- In most cases you will be using JSON formatted responses
- Most languages have JSON parsers that return the data as a structure
- In R prettify will give you a more human readable format

```
decoded = content(r, "text")

prettify (decoded)
```

Decoding the Response – Perl

- In most cases you will be using JSON formatted responses
- Most languages have JSON parsers that return the data as a structure
- In Perl Data::dumper (dumper) will give you a more human readable format

```
my $decoded = Dumper decode_json($r->{content});

print $decoded;
```

Decoding JSON Response

http://rest.ensembl.org/lookup/id/ENSG00000157764?content-type=application/json


```
{
  "source": "ensembl_havana",
  "object_type": "Gene",
  "logic_name": "ensembl_havana_gene",
  "version": 12,
  "species": "homo_sapiens",
  "description": "B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC
Symbol;Acc:HGNC:1097]",
  "display_name": "BRAF",
  "assembly_name": "GRCh38",
  "biotype": "protein_coding",
  "end": 140924764,
  "seq_region_name": "7",
  "db_type": "core",
  "strand": -1,
  "id": "ENSG00000157764",
  "start": 140719327
}
```

Decoding JSON response

- JSON is essentially a massive dictionary/hash/dataframe with keys and values.
- Sometimes a key contains another nested dictionary or a list
  - Which may contain another
    - And another
      - And another ...
- Look at the JSON to work out what keys you need
- You can cycle through all keys in a dictionary with for loops

GET Helper Function

- The helper function in your script makes your life easier by:
  - Calling the request with the specified server, extension and content-type
  - Getting the status of a failed query
  - Decoding the JSON (if you have used JSON as your content-type)
  - Returning the text (if you use any other content-type)
- Add it to the start of every script; then just call the function when you need to fetch an endpoint

GET Helper Function – Python

```
def fetch_endpoint(server, request, content_type):
    """
    Fetch an endpoint from the server, allow overriding of default content-type
    """
    r = requests.get(server+request, headers={ "Accept" : content_type})

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
        return r.text
```

GET Helper Function – R

```
Fetch_endpoint <- function(server, request, content_type){
    """
    Fetch an endpoint from the server, allow overriding of default content-type
    """
    r <- GET(paste(server, request, sep = ""),  accept(content_type))

    stop_for_status(r)

    if (content_type == 'application/json'){
        return (fromJSON(content(r, "text")))
    } else {
        return (content(r, "text"))
     }
}
```

GET Helper Function – Perl

```perl
# Fetch an endpoint from the server, allow overriding of the default content type
sub fetch_endpoint {
    my $http = HTTP::Tiny->new();
    my ($server, $extension, $content_type) = @_;
    $content_type ||= 'application/json';
    my $response = $http->get($server.$extension, { headers => { 'Accept' =>
$content_type } });
    die "Error: ", $response->{status}, "\n" unless $response->{success};
    if($content_type eq 'application/json') {
        return decode_json($response->{content});
    } else {
        return $response->{content};
    }
}
```


Questions?

Exercises 2

1. Write a script to lookup the gene called *IRAK4* in human and print the results in JSON.

Using Results

Since JSON is a dictionary, you can pull out a single datapoint using the key.


```
{
  "source": "ensembl_havana",
  "object_type": "Gene",
  "logic_name": "ensembl_havana_gene",
  "version": 12,
  "species": "homo_sapiens",
  "description": "B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC
Symbol;Acc:HGNC:1097]",
  "display_name": "BRAF",
  "assembly_name": "GRCh38",
  "biotype": "protein_coding",
  "end": 140924764,
  "seq_region_name": "7",
  "db_type": "core",
  "strand": -1,
  "id": "ENSG00000157764",
  "start": 140719327
}
```

Using Results – Python

Since JSON is a dictionary, you can pull out a single datapoint using the key.

```
server = "http://rest.ensembl.org/"
ext = "lookup/id/ENSG00000157764?"
con = "application/json"
get_gene = fetch_endpoint(server, ext, con)

symbol = get_gene['display_name']
print (symbol)
```

Using Results – R

Since JSON is a dataframe, you can pull out a single datapoint using the key.

```
server <- "http://rest.ensembl.org/"
ext <- "lookup/id/ENSG00000157764?"
con <- "application/json"
get_gene <- fetch_endpoint(server, ext, con)

symbol <- get_gene$display_name
symbol
```

Using Results – Perl

Since JSON is a hash, you can pull out a single datapoint using the key.

```perl
my $server = "http://rest.ensembl.org/";
my $ext = "lookup/id/ENSG00000157764?";
my $con = "application/json";
my $get_gene = fetch_endpoint($server, $ext, $con);

my $symbol = $get_gene->{'display_name'};
print $symbol;
```

Nested JSON Lists

http://rest.ensembl.org/overlap/region/human/7:140424943-140444564?feature=gene;content-type=application/json


```
[
    {
        "gene_id":"ENSG00000146955"
        "Feature_type":"gene",
        "external_name":"RAB19",
        "description":"RAB19, member RAS oncogene family [Source:HGNC
Symbol;Acc:HGNC:19982]",
        "Biotype":"protein_coding",
        "id":"ENSG00000146955",
    },
    {
        "gene_id":"ENSG00000103200",
        "Feature_type":"gene",
        "external_name":"AC069335.1",
        "Description":null,
        "Biotype":"processed_pseudogene"
        "id":"ENSG00000103200"
    }
]
```


List delineated by square brackets [ ] – no keys

Dictionary delineated by curly brackets { } – key-value pairs

Tools to decode JSON

To get the paths to particular items in the JSON, try pasting the full JSON into these tools:

- http://jsonpathfinder.com/
- http://convertjson.com/json-path-list.htm


Questions?

Exercises 3

1. Write a script to lookup the gene called *IRAK4* in human and print the stable ID of this gene.
2. Get all variants that are associated with the phenotype 'Coffee consumption'. For each variant, print
   a. the p-value for the association
   b. the PMID for the publication which describes the association between that variant and ‘Coffee consumption’
   c. the risk allele and the associated gene.
3. Get the mouse homologue of the human *BRCA2* and print the ID and sequence of both.

Exercise 3.2 – Python or Perl

Python

```
for variant in get_phen:
     pmid = variant['attributes']['external_reference']
```

Perl

```
foreach my $var (@{$get_phen}) {
     my %variant = %$var;
     my $pmid = $variant{attributes}{external_reference}; }
```

JSON

```
[{'Variation': 'rs10227393',
'attributes': {'associated_gene': 'NR',
               'beta_coefficient': '0.08 unit decrease',
               'external_reference': 'PMID:25288136',
               'p_value': '9.00e-6',
               'risk_allele': 'C'},
'description': 'Coffee consumption',
'location': '7:145365647-145365647',
'mapped_to_accession': 'EFO:0004330',
'source': 'NHGRI-EBI GWAS catalog'},
...
```

Exercise 3.2 – Python

```
pv = str(variant['attributes'].get('p_value'))
```

`str()` The p_value is a number. Python will not be able to print this easily alongside other objects that are strings. This command converts the number to a string so that it can be printed.

`get('p_value')` Some variant/phenotype associations do not have a p-value. This will return ‘None’ for these. If you do not use it, the script will break.

`get('risk_allele')` Some variant/phenotype associations do not have a risk allele. This will return ‘None’ for these. If you do not use it, the script will break.

Exercise 3.2 – R

R likes to make the data into tables. We can use flatten take advantage of that:

```
flat_get_phen <- flatten(get_phen, recursive = TRUE)
flat_get_phen[, c("Variation",
"attributes.p_value",
"attributes.external_reference",
"attributes.risk_allele",
"attributes.associated_gene")]
```

https://www.rdocumentation.org/packages/jsonlite/versions/1.5/topics/flatten

Exercise 3.3


```
{'data': [{'homologies': [{'dn_ds': 0.4246,
      'method_link_type':'ENSEMBL_ORTHOLOGUES',
      'source': {'align_seq': 'MPIG...',
            'cigar_line': '274M...',
            'id': 'ENSG00000139618',
            'perc_id': 56.9631,
            'perc_pos': 70.0995,
            'protein_id': 'ENSP00000369497',
            'species': 'homo_sapiens',
            'taxon_id': 9606},
      'target': {'align_seq': 'MPVE...',
            'cigar_line': '98M7...',
            'id': 'ENSMUSG00000041147',
            'perc_id': 58.486,
            'perc_pos': 71.9736,
            'protein_id': 'ENSMUSP00000038576',
            'species': 'mus_musculus',
            'taxon_id': 10090},
      'taxonomy_level': 'Euarchontoglires',
      'type': 'ortholog_one2one'}],
'id': 'ENSG00000139618'}]}
```


1. Single key: data
2. List, one entry
3. Keys: homologies and id
4. List, one entry
5. Keys to the homology relationship, including: source and target
6. Source (human) gene features
7. Target (mouse) gene features

Other Content Types – Python

- If you specify another content type (not JSON), the helper function will return text
- This can be used to get:
  - Sequence in FASTA format
  - Gene trees and homologues in various formats
  - Alignments

```
if content_type == 'application/json':
    return r.json()
else:
    return r.text
```

Other Content Types – R

- If you specify another content type (not JSON), the helper function will return text
- This can be used to get:
  - Sequence in FASTA format
  - Gene trees and homologues in various formats
  - Alignments

```
if (content_type == 'application/json'){
    return (fromJSON(content(r, "text")))
} else {
    return (content(r, "text"))
 }
```

Other Content Types – Perl

- If you specify another content type (not JSON), the helper function will return text
- This can be used to get:
  - Sequence in FASTA format
  - Gene trees and homologues in various formats
  - Alignments

```
if($content_type eq 'application/json') {
    return decode_json($response->{content});
} else {
    return $response->{content};
}
```

Other Content Types

Resource Information

Methods	GET
Response formats	fasta json seqxml text yaml jsonp

- Endpoint documentation pages list allowed content-types
- The wiki lists how you specify these

https://github.com/Ensembl/ensembl-rest/wiki/Output-formats

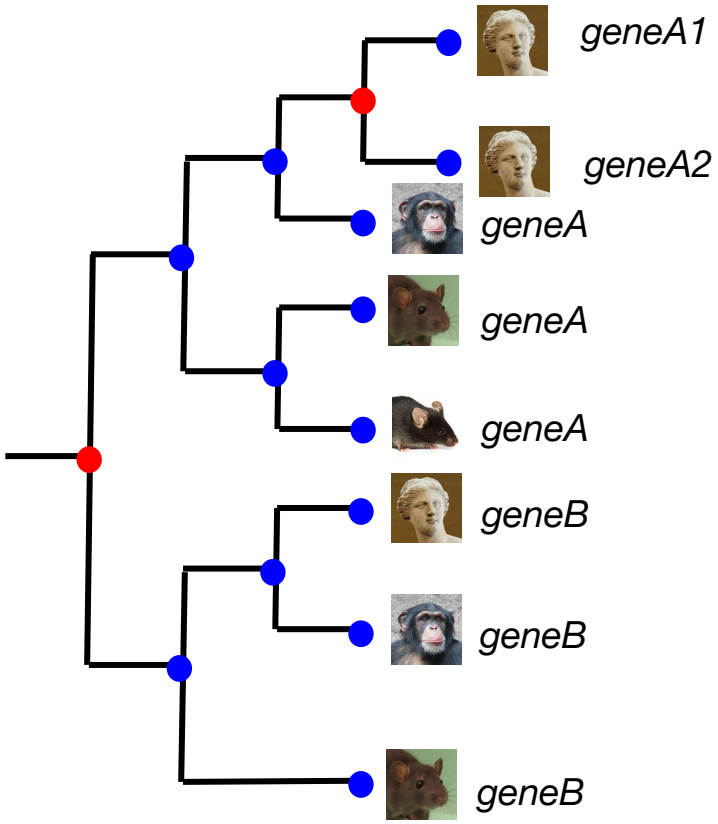
Gene Tree Endpoints

Comparative Genomics

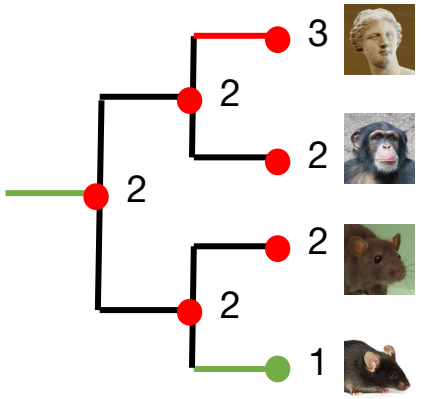
Resource	Description
GET cafe/genetree/id/:id	Retrieves a cafe tree of the gene tree using the gene tree stable identifier
GET cafe/genetree/member/id/:id	Retrieves the cafe tree of the gene tree that contains the gene / transcript / translation stable identifier
GET cafe/genetree/member/symbol/:species/:symbol	Retrieves the cafe tree of the gene tree that contains the gene identified by a symbol
GET family/id/:id	Retrieves a family information using the family stable identifier
GET family/member/id/:id	Retrieves the information for all the families that contains the gene / transcript / translation stable identifier
GET family/member/symbol/:species/:symbol	Retrieves the information for all the families that contains the gene identified by a symbol
GET genetree/id/:id	Retrieves a gene tree for a gene tree stable identifier
GET genetree/member/id/:id	Retrieves the gene tree that contains the gene / transcript / translation stable identifier
GET genetree/member/symbol/:species/:symbol	Retrieves the gene tree that contains the gene identified by a symbol
GET alignment/region/:species/:region	Retrieves genomic alignments as separate blocks based on a region and species
GET homology/id/:id	Retrieves homology information (orthologs) by Ensembl gene id
GET homology/symbol/:species/:symbol	Retrieves homology information (orthologs) by symbol

genetree vs cafe/genetree

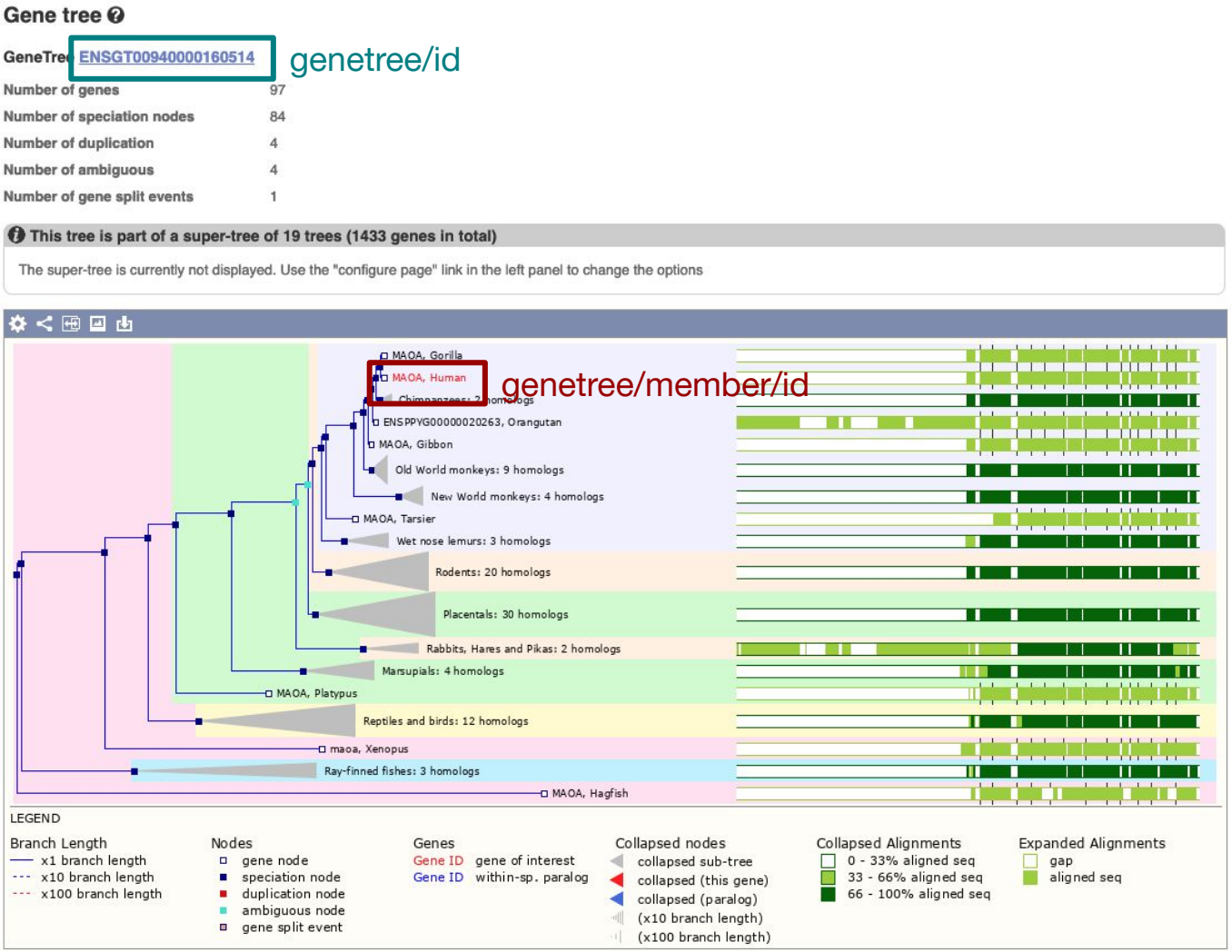
genetree



cafe/genetree



genetree/id vs genetree/member/id




Questions?

Exercises 4

1. Get the gene tree predicted for the gene ENSG00000189221 in full nh format.
2. Get the sequence of the gene ENSG00000157764 in FASTA format.

Gene Tree Endpoints

Comparative Genomics

Resource	Description
GET cafe/genetree/id/:id	Retrieves a cafe tree of the gene tree using the gene tree stable identifier
GET cafe/genetree/member/id/:id	Retrieves the cafe tree of the gene tree that contains the gene / transcript / translation stable identifier
GET cafe/genetree/member/symbol/:species/:symbol	Retrieves the cafe tree of the gene tree that contains the gene identified by a symbol
GET family/id/:id	Retrieves a family information using the family stable identifier
GET family/member/id/:id	Retrieves the information for all the families that contains the gene / transcript / translation stable identifier
GET family/member/symbol/:species/:symbol	Retrieves the information for all the families that contains the gene identified by a symbol
GET genetree/id/:id	Retrieves a gene tree for a gene tree stable identifier
GET genetree/member/id/:id	Retrieves the gene tree that contains the gene / transcript / translation stable identifier
GET genetree/member/symbol/:species/:symbol	Retrieves the gene tree that contains the gene identified by a symbol
GET alignment/region/:species/:region	Retrieves genomic alignments as separate blocks based on a region and species
GET homology/id/:id	Retrieves homology information (orthologs) by Ensembl gene id
GET homology/symbol/:species/:symbol	Retrieves homology information (orthologs) by symbol

Linking Endpoints together

- If you can pull a datapoint from the JSON, you can use it as input for another endpoint.
- You will need to link objects and extensions together.


Questions?

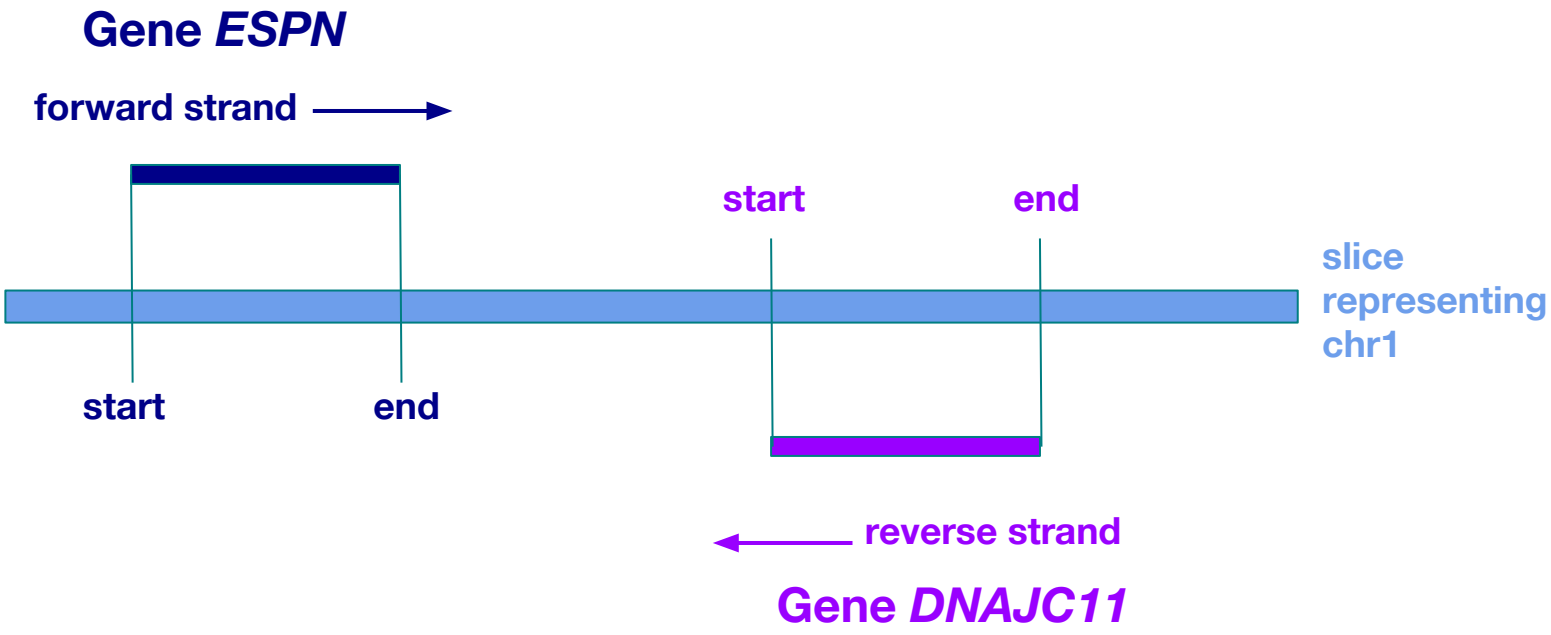
Exercises 5

1. Using the script from 3.1, add a call to fetch and print the sequence for the gene *IRAK4* in FASTA.
2. Print the stable ID of any regulatory features that overlap the region 1000 bp upstream of the *ESPN* gene. (Hints: get the gene ID first, then check the strand of the gene to see which way is upstream.)

Features

Features have a defined location on the genome.

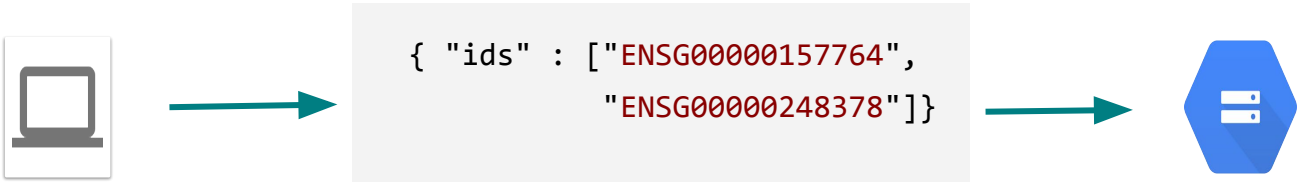
Start and end are always plotted on the forward strand. start < end



HTTP Methods - GET vs POST

GET http://rest.ensembl.org/lookup/ENSG00000157764

POST http://rest.ensembl.org/lookup/



POST allows you to run a query with multiple inputs at once.

POST Documentation

POST archive/id

Retrieve the latest version for a set of identifiers

No :parameter
Your extension is just this

Parameters

Required

No required parameters

Optional

Name	Type	Description	Default	Example Values
callback	*String*	Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.	-	*randomlygeneratedname*

Message

Content-type	Format	Example
application/json	{ "id" : array }	{ "id" : ["ENSG00000157764", "ENSG00000248378"] }

How you should format your input data

Example Requests

/archive/id

Resource Information

Methods	POST
Response formats	json jsonp
Maximum POST size	1000

Maximum list length

POST Input Data

Message

Content-type	Format	Example
application/json	{ "id" : array }	{ "id" : ["ENSG00000157764", "ENSG00000248378"] }

- Create a list of your values
- Convert it into JSON under the key listed in the page
- Python

```
data = json.dumps({ "id" : my_list })
```

- Perl

```
my $data = encode_json( { id => \@mylist } );
```

- R

```
data <- toJSON(list(id=mylist))
```

Using POST - Python

```
import requests, sys

server = "http://rest.ensembl.org"
ext = "/lookup/id"
headers={ "Content-Type" : "application/json", "Accept" : "application/json"}
r = requests.post(server+ext, headers=headers, data='{ "ids" :
["ENSG00000157764", "ENSG00000248378" ] }')

# error checking removed for space

decoded = r.json()
pprint (decoded)
```

Using POST - R

```
library(httr)
library(jsonlite)

server <- "http://rest.ensembl.org"
ext <- "/lookup/id"
genes <- c("ENSG00000157764", "ENSG00000248378")
body_values <- toJSON(list(ids=genes))

r <- POST(paste(server, ext, sep = ""), content_type("application/json"),
accept("application/json"), body = body_values)

prettify(content(r, "text"))
```

Using POST - Perl

```
import requests, sys

server = "http://rest.ensembl.org"
ext = "/lookup/id"
headers={ "Content-Type" : "application/json", "Accept" : "application/json"}
r = requests.post(server+ext, headers=headers, data='{ "ids" :
["ENSG00000157764", "ENSG00000248378" ] }')

# error checking removed for space

decoded = r.json()
pprint (decoded)
```

POST Helper Function

- You can also have a helper function for POST queries
- This is shown in the Jupyter notebooks, and can be defined in a script alongside the GET helper
- The helper function requires four parameters:
  - server
  - request
  - data – this is the JSON you created earlier
  - content_type

POST Helper Function – Python

```python
def fetch_endpoint_POST(server, request, data, content_type='application/json'):

    r = requests.post(server+request,
                      headers={ "Content-Type" : content_type},
                      data=data )

    if not r.ok:
        r.raise_for_status()
        sys.exit()

    if content_type == 'application/json':
        return r.json()
    else:
        return r.text
```

POST Helper Function – R

```
fetch_endpoint_POST <- function(server, request, content_type){
    """
    Fetch an endpoint from the server, allow overriding of default content-type
    """
    r <- POST(paste(server, request, sep = ""),  content_type(content_type),
accept(content_type), body = data)

    stop_for_status(r)

    if (content_type == 'application/json'){
        return (fromJSON(content(r, "text")))
    } else {
        return (content(r, "text"))
     }
}
```

POST Helper Function – Perl

```perl
# Fetch an endpoint from the server, allow overriding of the default content type
sub fetch_endpoint_POST {
    my $http = HTTP::Tiny->new();
    my ($server, $extension, $data, $content_type) = @_;
    $content_type ||= 'application/json';
    my $response = $http->request( "POST", $server.$extension, { headers => {
'Accept' => $content_type }, content => $data });
    die "Error: ", $response->{status}, "\n" unless $response->{success};
    if($content_type eq 'application/json') {
        return decode_json($response->{content});
    } else {
        return $response->{content};
    }
}
```

Decoding POST Queries

POST endpoints return a dictionary of dictionaries.

```
{
  "ENSG00000157764": {
    "source": "ensembl_havana",
    "object_type": "Gene",
...
  },
  "ENSG00000248378": {
    "source": "havana",
    "object_type": "Gene",
...
  }
}
```

Decoding POST Queries

- You can move through the dictionary/hash with:
  - Python

```
for key, value in post_query.items():
```

  - Perl

```
foreach my $hash_reference
(@{$post_query}) {
            my %hash = %$hash_reference;
    }
```

  - R just treats these as dataframes


Questions?

Exercises 6

1. Fetch the all the transcripts of *IRAK4* using the lookup endpoint. Fetch the cDNA sequences of all transcripts using a single POST request, and print in FASTA format.
2. You have the following list of variants:

   `rs1415919662, rs957333053, rs762944488, rs1372123943, rs553810871, rs1451237599, rs751376931`

   Get the variant class, evidence attributes, source and the most_severe_consequence for all variants using the variation POST endpoint.

Rate Limiting

Requests are rate limited to prevent a single user from monopolising the resources.

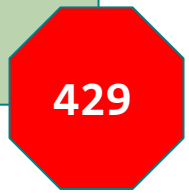
```
X-RateLimit-Limit: 55000
X-RateLimit-Reset: 892
X-RateLimit-Period: 3600
X-RateLimit-Remaining: 54999
Retry-After: 40.0
```

Response headers show we are allowed 55000 requests over an hour (3600 seconds).

I.e. an average 15 requests per second.

1 request used and 892 sec (~15 minutes) from reset.

Wait 40 seconds before sending another request or...




Questions?

Exercises 7

The Jupyter notebook contains a script that queries the *ping* endpoint 25 times, printing the count, the HTTP Status Code, and the X-RateLimit-Remaining header each time.

1. Increase the number of loops, do you start to get 429 errors?
2. Can you add in a step to make it wait a few seconds every iteration? Or every 100 iterations?

Feedback Survey

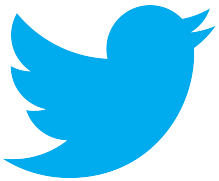
training.ensembl.org/events

Help and Documentation

- Ensembl outreach team: helpdesk@ensembl.org
- Developer mailing list: dev@ensembl.org
- Endpoint documentation: http://rest.ensembl.org
- User guide: https://github.com/Ensembl/ensembl-rest/wiki
- Change log: https://github.com/Ensembl/ensembl-rest/wiki/Change-log

Follow us

www.facebook.com/Ensembl.org



@ensembl
@ensemblgenomes
@AstridGAGall

www.ensembl.info

Publications

Yates A. *et al*
The Ensembl REST API: Ensembl Data for Any Language
Bioinformatics (2015) 31(1): 143–145,
doi.org/10.1093/bioinformatics/btu613
http://europepmc.org/articles/PMC4271150

Cunningham F. *et al*
Ensembl 2019
Nucleic Acids Research (2018) gky1113, doi.org/10.1093/nar/gky1113
https://europepmc.org/abstract/MED/30407521

Kersey P. *et al*
Ensembl Genomes 2018: an integrated omics infrastructure for non-vertebrate species
Nucleic Acids Research (2017) gkx1011, doi.org/10.1093/nar/gkx1011
http://europepmc.org/abstract/MED/29092050

Ensembl 2018

Ensembl Acknowledgements

The Entire Ensembl Team

Fiona Cunningham, Premanand Achuthan, Wasiu Akanni, James Allen, M. Ridwan Amode Irina Armean, Ruth Bennett, Jyothish Bhai, Konstantinos Billis, Sanjay Boddu, Carla Cummins, Claire Davidson, Kamalkumar Jayantilal Dodiya, Astrid Gall, Carlos García Girón, Laurent Gil, Tiago Grego, Leanne Haggerty, Erin Haskell, Thibaut Hourlier, Osagie Izuogu, Sophie Janacek, Thomas Juettemann, Mike Kay, Matthew Laird, Ilias Lavidas, Zhicheng Liu, Jane Loveland, José Marugán, Thomas Maurel, Aoife McMahon, Benjamin Moore, Joannella Morales, Jonathan Mudge, Michael Nuhn, Denye Ogeh, Anne Parker, Andrew Parton, Mateus Patricio, Ahamed Imran Abdul Salam, Bianca Schmitt, Helen Schuilenburg, Dan Sheppard, Helen Sparrow, Eloise Stapleton, Marek Szuba, Kieron Taylor, Glen Threadgold, Anja Thormann, Alessandro Vullo, Brandon Walts, Andrea Winterbottom, Amonida Zadissa, Marc Chakiachvili, Adam Frankish, Sarah Hunt, Myrto Kostadima, Nick Langridge, Fergal Martin, Matthieu Muffato, Emily Perry, Magali Ruffier, Daniel Staines, Stephen Trevanion, Bronwen Aken, Andrew Yates, Daniel Zerbino, Paul Flicek

Funding

wellcome

EMBL

National Human Genome Research Institute

Open Targets

SAVE THE TASMANIAN DEVIL PROGRAM APPEAL

BBSRC bioscience for the future

TRANSFORMING GENETIC MEDICINE INITIATIVE

Co-funded by the European Union

Training Materials

- Ensembl training materials are protected by a CC BY license
- http://creativecommons.org/licenses/by/4.0/
- If you wish to re-use these materials, please credit Ensembl for their creation
- If you use Ensembl for your work, please cite our papers
- http://www.ensembl.org/info/about/publications.html