




SEAVE

*A Comprehensive Variant Filtration
Platform for Clinical Genomics*



ADMINISTRATION GUIDE

v1.1

Dr. Velimir Gayevskiy

This guide is for Seave administrators and describes how to maintain and manage a Seave server.

It assumes that are you comfortable with cloud computing, Linux, MySQL and server administration.

Contents

About	1
Acknowledgements	1
Installation	2
Installation Options	2
Amazon Machine Image (AMI)	2
Important Initial Configuration	3
SSL	3
User Accounts	3
Seave Tokens and MySQL Password	3
SSH Keys	4
Importing Data	4
GEMINI Databases	4
Genome Block Store	6
Modifying Gene Lists	7
Add a Gene List	7
Add Gene(s) to List(s)	7
Remove Gene(s) from List(s)	8
Remove Gene List	8
Rename Gene List	8
View Genes in List	8
Modifying Users and Groups	8
Add a Group	9
Add a User	9
Remove a Group	9
Manage User-Group Membership	9
View Users in a Group	10

Manage Administrator Priveleges	10
Managing GEMINI Databases	10
Add Database	10
Delete Database	11
Modify Database Pedigree Information with a PED File	11
Rename Database	11
Generate Database Summary	12
Move Database to Different Group	12
View Annotation History	12
Managing GBS Data	12
Adding GBS Data	13
Deleting GBS Data	13
Advanced	13
Going From VCF to GEMINI Database for Import Into Seave . . .	13
VCF Creation	14
VCF Decomposition and Normalisation	14
VCF Genomic Impact Annotation	15
Creating a GEMINI Database	16
Automatically Importing Databases or GBS Variants Into Seave	16
Installing Seave from the Source Code on a Private Server or Local Computer	17
Viewing Variants in IGV	18
Setting Up SSL	18
Using a Separate GEMINI Databases Volume	19
Using a Separate MySQL Database Server	19
Automated Testing Using Selenium	19
Keeping Seave Up To Date Using Automated Deployments	19
Updating Annotation Sources	20
OMIM	21

COSMIC	23
MITOMAP	25
RVIS	26
ClinVar	27
COSMIC CGC	29
Seave Logging	30
Gene Name Validation	30

About

Seave is a web-based platform for genetic variant filtration, primarily for use in clinical genomics. It stores short variants in the form of single nucleotide polymorphisms (SNPs) and insertions/deletions (Indels). Long genetic variants such as copy number variants (CNV), structural variants (SV) and losses of heterozygosity (LoH) are stored in the form of genomic blocks of any size. Variants can be filtered using a variety of parameters and the results are annotated with a large number of external annotation databases.

The development of Seave began in January 2015 to provide the Kinghorn Centre for Clinical Genomics (KCCG) at the Garvan Institute a place to store genomic data and make it accessible to clinicians and researchers without bioinformatics backgrounds. It grew out of a simple front end to GEMINI to include many advanced features. It has gone on to be used commercially within Genome.One, a spin-off from KCCG, and for a variety of germline and somatic research projects within and outside the Garvan Institute.

Seave was designed and developed by Dr. Velimir Gayevskiy. Dr. Tony Roscioli contributed early advice for improving inheritance logic and usability for clinicians. Dr. Mark Cowley has overseen development and contributed significantly to feature requests and code reviews.

Acknowledgements

We thank the Translational Genomics group (KCCG) and Genome.One for their comments and questions that lead to new features and bug fixes: Lisa Ewans, Eric Lee, Marie Wong, Kishore Kumar, Clare Puttick and André Minoche.

Seave would not be possible without the existence of GEMINI, a free academic tool created by Dr. Aaron Quinlan at the University of Utah. His work allowed us to rapidly prototype Seave and use GEMINI in any way we desire due to his generous MIT Licensing.

Installation

Installation Options

As Seave is a web-based platform, it is designed to be installed on a web-server under your control and used by multiple users by navigating to the URL/IP of the server.

The recommended method of creating such a server is to use [Amazon Web Services](#), a cloud platform for which a server image (AMI) of Seave is supplied. This image contains the Seave code as well as all software packages required for Seave to run correctly — *most of the setup has been taken care of for you*. Please make sure to configure your Seave server immediately after creating it or it is extremely vulnerable; you can find information for how to do so in the [Important Initial Configuration](#) section.

Advanced users may wish to run Seave on their own hardware either on a private server or locally on their computer. This is possible but requires quite a bit more configuration, it is not recommended and not supported. Nonetheless, some instructions to do so can be found in the [Installing Seave from the Source Code on a Private Server or Local Computer](#) section of this documentation.

Amazon Machine Image (AMI)

The Seave AMI can be found in 2 AWS regions: Asia Pacific (Sydney) and US East (N. Virginia). To find it, navigate to the AMIs section of your AWS interface and change "Owned by me" to "Public images" in the dropdown to the left of the search box. After this, search for "SeaveAMI" in the search box and the image should appear. You can now start your own instance (server) using this image. A t2.micro instance is sufficient to run Seave on a smaller scale, but you may want to provision more resources if you will have many concurrent users and/or very large databases. You should open up ports 80 and 443 (http/https) to the world and port 22 (SSH) to your or your institution's IP range for logging in to the server. All outbound connections should be allowed.

Once your instance is running, it will be provisioned a public IP address by AWS. You can further provision a permanent public IP address under "Elastic IPs" and assign this to the instance. This permanent public IP can then be assigned to a domain name via DNS settings so users simply navigate to "seave.myinstitute.org" to visit the institute's Seave. At this point you can now open your browser and navigate to either the IP address of the server or the domain, if you have set that up, to see your fresh Seave

server.

Important Initial Configuration

This section is very important if you have just started a new Seave server. Please make the changes here immediately or your Seave is very insecure.

SSL

By default, Seave will redirect all requests to https (SSL) using a self-signed certificate. Most browsers will complain loudly about this as it is an insecure practice to use a self-signed certificate. While it can be forcibly bypassed, you should deal with this instead by either adding in your own SSL certificate or disabling this redirection. Indeed, the reason we configure it this way by default is that adding your own certificate is easy and results in users securely communicating with the server — see the [Setting Up SSL](#) section for how to do this. To disable the redirection to https, SSH in to the server and navigate to `/var/www/html`. Open the `.htaccess` in your editor of choice and remove all lines in the "# When the request is using HTTPS" and "# When the request is using HTTP" blocks.

User Accounts

A fresh Seave server will have the following two user accounts already created:

1. **Administrator** Username: `admin@seave.bio` Password: `yA0s8KHF`
2. **User** Username: `default@seave.bio` Password: `u0FKn5DA`

Since these passwords are public, anyone can log in with these details into your new Seave server. To fix this, log in to the Seave website under the administrator account and navigate to the User Administration page. Then, either change the password for both accounts or create new ones to replace them (deleting the old ones), using correct email addresses for your organisation.

Seave Tokens and MySQL Password

Next, you will need to SSH in to the server and navigate to `/var/www/html`. Here you will find the file `config.ini`. Open it in your editor of choice and

change the tokens on the following lines:

1. `gbs_import_token= ...`
2. `gemini_db_import_token= ...`

While it is not strictly necessary provided you have secured the webserver (i.e. not opened port 3306 used by MySQL to the world), it's nonetheless a good idea to also change the `[mysql] password=` value in `config.ini` and the corresponding password in the MySQL server running on the Seave webserver. This will protect against accidental misconfiguration where port 3306 is opened to external traffic and the public password is used to log in to the MySQL server by a third party.

SSH Keys

When you launch an instance, AWS will ask you which SSH key you would like to use for connecting to it — a new one or an existing one. It will then dutifully add this SSH key to the `authorized_keys` file on the instance.

The Seave AMI already comes configured with an SSH key inserted by the developer (Vel). If you need support for your Seave, this means it can be connected to without any configuration on your part (besides firewall access). However, this is a back door, and if you'd rather not have this option, you can delete this key from having access to your Seave. You can do this by navigating to `/home/ubuntu/.ssh` and removing the line ending in `SeaveAMI` in the `authorized_keys` file.

Importing Data

Seave stores short variants (SNPs and Indels) in **GEMINI** databases as files and long variants (CNVs and SVs) in the Genome Block Store within the internal MySQL database. There are multiple ways to import data into each.

GEMINI Databases

GEMINI databases must be produced with **GEMINI** and are files with a `.db` file extension. These files can be imported into Seave in one of three ways:

1. Through the Database Administration page when you are logged in as an administrator in Seave. Navigate to this page and you will see

a “Add a new database” section which invites you to enter a URL to a database and select a Seave group to import the database into. You will need to create a URL pointing to your [GEMINI](#) database through something like Dropbox, DNAnexus, AWS S3 or your institution’s HPC cluster. After you click “Add database”, Seave will verify the URL and begin downloading the file into the selected group. During this time it will be unusable in the web browser from which you launched the import. This process can take minutes for panel/exome databases up to hours for larger databases containing many whole genomes, depending on the connection speed between your Seave server and the server hosting the database. After the import is successful, you will see a message to the effect, except if the import took a long time and your connection to Seave was severed in the mean time.

2. You can directly copy the database file into the right directory containing the databases on your Seave server. This directory is specified in `config.ini` file in the web root under the `[gemini]` heading in `db_dir=`. Within this directory are each of the Seave groups you have set up as sub-directories, copy the database into the correct sub-directory to automatically assign it to that group. Copying to the server can be performed with the standard `scp` or `rsync` commands.
3. Databases can be imported automatically using the Seave import API. To use this method, you must navigate to a specific URL containing information that Seave uses to import your database. This method is most suited to automated systems that import data at the end of a bioinformatics pipeline so are easily capable of constructing the URL required. The URL is to the `dx_import.php` page of your Seave server with a number of GET variables to populate:
 - **token** (*required*) — Token for authentication, specified in `config.ini` under the `[dx_import]` heading in `gemini_db_import_token=`
 - **url** (*required*) — URL to your [GEMINI](#) database, must end with `.db`
 - **import_type** (*required*) — Always a value of `gemini_db`
 - **group** (*required*) — The group to import the database into, the group must exist
 - **test** (*optional*) — A value of `bamboo` to import the database completely and then automatically delete it; this is optionally used for testing purposes only
 - **md5** (*optional*) — An optional MD5 hash of the database file, Seave will generate an MD5 hash after downloading the database

and compare it against the submitted value to ensure data integrity

An example of an import URL is:

```
https://www.seave.bio/dx_import.php?token=d8e8fca2dc0f896fd7cb4cb0031ba249&
url=https://dl.dnanex.us/F/D/yvJgVbQqx59VBXFyykykVv8y26RPxQp0BiPqXZbY/
321773.G1.hc.vep.db&md5=d41d8cd98f00b204e9800998ecf8427e&import_
type=gemini_db&group=KCCG
```

Genome Block Store

Large variants must be produced using one of the allowed tools and can be imported into the Genome Block Store in one of two ways:

1. Through the GBS Administration page when you are logged in as an administrator in Seave. Navigate to this page and you will see a “Import genomic blocks” section which invites you to select a method, enter a sample (for some methods) and upload a genomic blocks file from the method selected. The file you must upload differs per method and is specified under the upload button. Upon specifying this information and clicking “Import genomic blocks”, Seave will parse the file you selected for genomic blocks and display them to you in a table. Carefully inspect this table to make sure the information matches what is in the input file and once you are satisfied click “This data is correct, import it”. After a short period, your data will be saved in the GBS and will be immediately linked to any GEMINI databases with the same sample name(s).
2. GBS data can be imported automatically using the Seave import API. To use this method, you must navigate to a specific URL containing information that Seave uses to import your GBS data file. This method is most suited to automated systems that import data at the end of a bioinformatics pipeline so are easily capable of constructing the URL required. The URL is to the `dx_import.php` page of your Seave server with a number of GET variables to populate:
 - **token** (*required*) — Token for authentication, specified in `config.ini` under the `[dx_import]` heading in `gbs_import_token=`
 - **method** (*required*) — The software used to generate the GBS import file, must be one of CNVnator, LUMPY, Sequenza, ROHmer, VarpipeSV, Manta or CNVkit
 - **url** (*required*) — URL to your GBS import file, must end with the relevant extension for the method specified

- **import_type** (*required*) — Always a value of GBS
- **sample_name** (*required sometimes*) — If the sample name is not specified in the data file for the method selected, it must be specified here
- **test** (*optional*) — A value of bamboo to import the GBS data completely and then automatically delete it; this is optionally used for testing purposes only
- **md5** (*optional*) — An optional MD5 hash of the import file, Seave will generate an MD5 hash after downloading the file and compare it against the submitted value to ensure data integrity

An example of an import URL is:

```
https://www.seave.bio/dx_import.php?token=d8e8fca2dc0f896fd7cb4cb0031ba249&
url=https://dl.dnanex.us/F/D/PzJ1z7PbkW2p85gG9P1v9J3FnzvQS12DJqHP3bbv/
321773.D1vsG1.manta.somaticSV.vcf.gz&md5=be616328d6eb1c36ac8d218de0e4ef51&
import_type=GBS&method=Manta&sample_name=321773_T
```

Modifying Gene Lists

Seave allows the creation, modification and deletion of gene lists on the Gene List Administration page when you are logged in as an administrator. You can find a link to the Gene List Administration page on the home page or by hovering over “Databases” on the top menu and clicking “Gene List Administration”.

Add a Gene List

The “Add gene list” section of the page allows you to enter a gene list name which will be created as an empty list — indicated by the (0) after the list name anywhere it appears.

Add Gene(s) to List(s)

To add genes to this new list, or an existing one, select one or more lists in the “Add gene(s) to list(s)” section and type in semicolon-delimited gene names in the box below. Seave will validate the genes you enter against a list of valid gene names and add the ones that pass to the gene list(s) selected. If any genes you enter are not found to be valid, you will get a list of these and you will need to find their synonyms and add them to the list(s) again until you are successful. For more information on this validation see the Gene Name Validation section.

Remove Gene(s) from List(s)

Removing genes from one or more gene lists works the same way as adding them, this time in the “Delete gene(s) from list(s)” section. The only validation on gene names entered will be whether they exist in the gene list(s) you select.

Remove Gene List

An entire gene list can be removed in the “Delete gene list” section by selecting a gene list in the dropdown menu. This will immediately delete the gene list from Seave so it no longer appears on any query page. Deletions of gene lists are logged in the Seave database in case a mistake is made, but are not readily able to be restored. If you need to see the genes that were in a gene list that was deleted, log in to the MySQL server and run the following query, filling in the deleted gene list name:

```
1 SELECT gene_name, date_deleted FROM KCCG_GENE_LISTS.  
   gene_lists_deletions WHERE list_name = '<gene list  
   name>';
```

Rename Gene List

A gene list can be renamed in the “Rename gene list” section, first select the gene list you wish to rename in the dropdown menu and then enter the new list name in the input box below.

View Genes in List

The “View genes in gene list” section allows you to view all genes in the gene list you select from the dropdown menu. This includes the date and time when each gene was added to the list and also includes a download link to a tab-separated TSV file for importing the gene list into a spreadsheet like Excel.

Modifying Users and Groups

Seave contains a user and group management system that allows administrators to manage access to data. Every **GEMINI** database on Seave belongs to a single group and users can have access to any number of groups. If a

user is in multiple groups, they will see (and be able to query) all databases within the groups they have access to on the Databases page. To manage users and groups, you must log in as an administrator and click the “User administration” button on the home page, or alternatively you can find a link on the top menu by hovering over “Databases” and clicking “User Administration”.

Add a Group

The first section titled “Add a group” allows you to create a new group in Seave. Enter a group name in the “Group name” box, this is the **user-facing** name for the group and must only contain letters and numbers (no spaces) not exceeding 50 characters in length. Add a description of the group underneath in the “Short description” box, this is just a long form expansion of what/who the group is for.

Add a User

Adding a user to Seave can be done in the next section titled “Add a user”. Enter the new user’s email address and a password for their account and click “Add user” to add them to Seave. The password must be managed by administrators, there is currently no functionality for non-administrator users to change their own passwords. Passwords are stored as salted hashes in Seave so there is no way to extract a user’s password from the database.

Remove a Group

The next two sections titled “Remove a group” and “Remove a user” are for removing groups or users from Seave. Each contains a dropdown menu of all groups or users and a button to remove the selected option. Groups can only be removed when there are no databases belonging to them. Users can be removed without any restrictions.

Manage User-Group Membership

User-group membership is managed in the next two sections titled “Add a user to a group” and “Remove a user from a group”. Each section first contains a dropdown list of all users in Seave followed by either the groups they *are not* in (for adding a user to a group) or the groups they *are* in (for removing a user from a group). Select the user and group combination for

the task you want to perform and the user will be added or removed from the group.

The next section titled “Change a user’s password” allows you to change any user’s password. Select the user whose password you want to change from the dropdown and enter a new password into the box below. Pressing “Change password” will immediately change that user’s password, make sure to give them the new password and store it securely!

View Users in a Group

To interrogate current group memberships, the next section titled “View all users in a group” lets you select a group from the dropdown menu and view all users within it. The resulting table of users in the group includes the user’s email address, whether they are an administrator, the time when they were added to the group and the time when they were added to Seave as a user.

Manage Administrator Privileges

The final two sections control administrator access to Seave. In the “Make a user an administrator” section, you can select a non-administrator user and make them an administrator. In the “Remove administrator access” section you can select an administrator user and remove their administrator access. You cannot remove the last administrator’s access as there must be at least one administrator.

Managing GEMINI Databases

Short variants (SNPs and Indels) are stored in GEMINI databases within Seave. These databases are discrete .db files produced by GEMINI and each belong to a group within Seave. The “Database administration” page allows administrators to manage the databases. The page can be found by clicking “Data administration” from the home page or by clicking “Database administration” after mousing over “Databases” in the top menu.

Add Database

One of the ways to import GEMINI databases into Seave is via the “Add a new database” section. The requirements are a URL to the database (ending

in .db) and a group to import the database into. For more information on this process see the [GEMINI Databases](#) section.

Delete Database

Databases can be deleted in the next section “Delete a database”. This contains a dropdown list of all databases in all groups, select the one you wish to delete and press “Delete database”. Note that deletion is immediate and permanent, **there is no way to restore a deleted database**. You will need to re-import any database that is accidentally deleted.

Modify Database Pedigree Information with a PED File

The “Annotate a database” section allows you to select a database from the dropdown list and upload a .ped pedigree file to apply to it. The pedigree file must consist of 6 tab-separated columns in the following order:

1. **Family name** — to be displayed in Seave
2. **Sample name** — must be the same as in the database
3. **0** — normally used for father sample name but not used by Seave
4. **0** — normally used for mother sample name but not used by Seave
5. **Gender** — 1 for male, 2 for female and 3 for unknown
6. **Affected status** — -9 for unknown, 1 for unaffected and 2 for affected

This is an example of a pedigree file in this format:

1	NA12878Trio	NA12878	0	0	2	2
2	NA12878Trio	NA12891	0	0	1	1
3	NA12878Trio	NA12892	0	0	2	1

Seave will run the [GEMINI](#) command `gemini annotate` to apply your pedigree file to the database. If this returns a non-zero exit code, Seave will report an error.

Rename Database

The “Rename a database” section lets you select a database and enter a new filename for it. This filename must end in .db and not already exist in the group. The new filename will be immediately displayed on the databases page for all users who have access to it.

Generate Database Summary

Seave automatically generates a variant counts summary when a database is imported using the API or via the “Add a new database” section. However, if a database is imported as a file, this report won’t be generated and must be generated manually in the “Re-generate a summary” section. Select the database for which to generate the report and click “Re-generate summary” to begin the process. Generating the summary involves running a number of [GEMINI](#) queries so can take quite some time for a large database with millions of variants. Seave will be unusable in your current browser once you start this process, switch to another browser to keep using it if needed.

Move Database to Different Group

Databases within Seave must belong to one group. The “Move database” section allows you to reassign a database from one group to another. Select the current group and database from the dropdown and then select the group to move the database to and click “Move database” to move it. This will immediately reassign the database and it will be visible and queryable to users in the new group. A database with the same filename must not already exist in the new group.

View Annotation History

Seave contains a number of annotation databases (such as ClinVar) that are used to improve variant annotation. Advanced users will want to update these annotation databases as they improve over time (see the section on [Updating Annotation Sources](#)). As part of the update process, each update should be logged in Seave to track the current version of each annotation and when it was updated. The “View annotation history” section lets you select an annotation from all of the annotation sources used in Seave and view the history of updates to that annotation. This is useful for auditing purposes, such as when you need to know what version of an annotation was used on a specific date.

Managing GBS Data

Variants affecting large parts of the genome such as CNVs, SVs, LoHs are stored in the Genome Block Store (GBS) within Seave. These are stored in a normalised schema in Seave’s MySQL database — also used for storing

annotation databases, users/groups, gene lists, etc. The “GBS Administration” page allows administrators to manage GBS data. The page can be found by clicking “GBS administration” from the home page or by clicking “GBS Administration” after mousing over “Databases” in the top menu.

Note: the GBS feature of Seave is the least developed and, as such, is not as feature-rich as the short variant filtration.

Adding GBS Data

One of the ways to import GBS data into Seave is via the “Import genomic blocks” section. As output files containing these events are generally small, you can directly upload them to Seave. For more information on this process see the [Genome Block Store](#) section.

Deleting GBS Data

GBS data is stored in Seave linked to sample names only (i.e. not belonging to groups). The “Delete genomic blocks” section thus allows you to select a sample name with GBS data and view the methods for which GBS data has been imported. Select the sample-method combination you want to delete and click “Delete blocks” to remove it. Once the last method for a sample has been removed, the sample will no longer appear in the samples list.

Advanced

Going From VCF to GEMINI Database for Import Into Seave

Seave manages [GEMINI](#) databases to store and query SNV and Indel variants. For a number of reasons, we have opted to decouple the variant annotation and [GEMINI](#) database creation from Seave:

- We wanted to handle WGS-sized data, which means >3Gb VCF files that are cumbersome to upload via a webpage.
- Variant annotation using VEP and [GEMINI](#) is computationally expensive for WGS data, where we typically use 16- to 32-core servers. We felt that it makes sense to keep this heavy compute as part of the production genomics pipeline.
- We have a production analysis pipeline, so adding an analysis module at the end to push data into Seave was straightforward.

To make it easier to adopt Seave, this section will go through the process of starting with a VCF file through to creating a GEMINI database that is ready for import into Seave.

We use VEP and the b37d5 (1000 genomes + decoy) reference genome in our production pipeline so the process outlined here will follow this. Other options may work too, but this is what works well for us.

VCF Creation

Germline DNA — VCF files are created by a Best Practices (GATK) production pipeline. Our primary germline variant caller is GATK HaplotypeCaller v3.3. We use gVCF mode, so each sample has a g.vcf.gz extension. These are joint-called into cohorts using GATK GenotypeGVCFs. For WGS data we then apply VQSR.

Somatic DNA — Our primary somatic variant caller is Strelka2. By default, the VCF files are incompatible with GEMINI as they lack the required GT field, the optional AD, DP, GQ fields, and always name the samples NORMAL and TUMOR. We have a script to post-process Strelka VCF files to make them compatible with GEMINI and provide fields that are useful for Seave. See scripts/strelka_add_to_FORMAT.py in the repository for this guide. To combine multiple Strelka VCFs for multi-sample or cohort analysis of somatic variants, we use GATK CombineVariants

VCF Decomposition and Normalisation

Multi-allelic variants (e.g. REF: A, ALT: T,C) are output by most variant callers, but downstream tools often don't handle them well. The problem lies in having to deal with conflicting or extra data for each allele. The standard solution is to first decompose variants such that each of the multiple alleles becomes a separate variant sharing the same position and reference allele. The next step is to normalise these decomposed variants to adjust the position and reference allele if needed (usually in the case of SNP+Indel multi-allelic variants).

We use the tool Vt (v0.5) for variant decomposition and normalisation, following the process documented in the GEMINI documentation [here](#). GEMINI expects variants to be decomposed and normalised and will issue a warning if it finds any that aren't. The reason for this is that the annotations GEMINI adds to variants have been decomposed and normalised to enable maximum ability to annotate variants.

We use the following command for Vt:

```

1 zcat ~/in/vcfgz/$input_filename | sed 's/ID=AD,Number=./
  ID=AD,Number=R/' | vt decompose -s - | vt normalize
  -r genome.fa - | vt sort -o $output_file -
2
3 tabix -p vcf $output_file

```

VCF Genomic Impact Annotation

We use [Variant Effect Predictor \(VEP\)](#) for annotating variants. Seave works well with variants annotated by VEP v74, v79 and v87; presumably the intermediate and following versions work too.

There is a [newer version](#) of VEP which we have not thoroughly tested, so we recommend using the older [ensembl-tools-vep](#). See `scripts/Make-file.vep87` in the repository for this guide for directions on installing VEP 87 along with the required plugins to make the most of Seave.

We do not find the `upstream_gene_variant` and `downstream_gene_variant` annotations useful for coding or regulatory variant analyses, so we delete them. See `scripts/filter_vep.py` in the repository for this guide for how this is done.

To run VEP on a VCF we use the following parameters:

```

1 # Annotate the VCF with VEP
2 /vep/variant_effect_predictor.pl -i ./in/vcfgz/* --
  species homo_sapiens --vcf -o output.vcf --stats_file
  "$vcfgz_prefix".vep.html --offline --fork 'nproc' --
  no_progress --canonical --polyphen b --sift b --
  symbol --numbers --terms so --biotype --total_length
  --plugin LoF,human_ancestor_fa:false --fields
  Consequence,Codons,Amino_acids,Gene,SYMBOL,Feature,
  EXON,PolyPhen,SIFT,Protein_position,BIOTYPE,CANONICAL
  ,Feature_type,cDNA_position,CDS_position,
  Existing_variation,DISTANCE,STRAND,CLIN_SIG,LoF_flags
  ,LoF_filter,LoF,RadialSVM_score,RadialSVM_pred,
  LR_score,LR_pred,CADD_raw,CADD_phred,
  Reliability_index,HGVSc,HGVSp --fasta /vep/
  homo_sapiens/87_GRCh37/Homo_sapiens.GRCh37.75.dna.
  primary_assembly.fa.gz --hgvs --shift_hgvs 1 --dir /
  vep
3
4 # VEP79 introduced spaces into the INFO field, for the
  sift & polyphen2 annotations. fix them.
5 perl -ne 'if ($_ !~ /\#/) { $_ =~ s/ /\_ /g; print $_; }
  else { print $_; }' output.vcf > output.clean.vcf
6 mv output.clean.vcf output.vcf

```

```

7
8 # Replace upstream_gene_variant/downstream_gene_variant
  with intergenic_variant
9 python ./filter_vep.py --vcf output.vcf | vcf-sort |
  bgzip > output.sorted.vcf.gz
10 tabix -p vcf output.sorted.vcf.gz

```

Creating a GEMINI Database

GEMINI installation is straightforward from the [documentation](#). We have tested Seave extensively with v0.11.*, and more recently with 0.17.*, 0.18.* and 0.19.1.

Databases are created from annotated VCF files with:

```

1 gemini load -v "$vcfgz_path" --cores 'nproc' --skip-gerp-
  bp -t VEP out.db

```

Automatically Importing Databases or GBS Variants Into Seave

Our production pipeline runs on [DNA Nexus](#) and we have created an app there that allows the import of GEMINI databases and GBS (long) variants into any group on multiple Seave servers.

The logic of the app for importing GEMINI databases can be distilled to:

```

1 TOKEN=xxxxxxxxxxxxxxxx
2 MD5=$(md5sum "$filename" | cut -d"_" -f1)
3 SEAVE="https://www.seave.bio"
4 SEAVE_URL="${SEAVE}/dx_import.php?token=${TOKEN}&url=${
  DX_URL}&md5=${MD5}"
5 SEAVE_URL="${SEAVE_URL}&import_type=gemini_db"
6 SEAVE_URL="${SEAVE_URL}&group=${group}"
7 OUTCOME=$(curl '-s' '-S' '-L' "${SEAVE_URL}")
8 echo "Seave_reported:_$OUTCOME"

```

The logic of the app for importing GBS (long) variants can be distilled to:

```

1 TOKEN=xxxxxxxxxxxxxxxx
2 MD5=$(md5sum "$filename" | cut -d"_" -f1)
3 SEAVE="https://www.seave.bio"
4 SEAVE_URL="${SEAVE}/dx_import.php?token=${TOKEN}&url=${
  DX_URL}&md5=${MD5}"
5

```

```

6 all_methods=(CNVnator LUMPY Sequenza ROHmer VarpipeSV
  Manta CNVkit)
7 if [[ ! "${all_methods[@]}" =~ "${method}" ]]; then
8     echo "A valid 'method' must be specified if data is
  being imported into the GBS."
9     exit 1
10 fi
11
12 # VCF files contain the sample names in the header. Many
  CNV/SV callers produce TSV files, so you also need to
  specify the sample name.
13 sample_name_methods=(CNVnator Sequenza ROHmer CNVkit)
14
15 # If the method specified doesn't include a sample name
  in the file to import, make sure the user specified
  one
16 if [[ "${sample_name_methods[@]}" =~ "${method}" ]];
  then
17     if [[ "$sample_name" == "" ]]; then
18         echo "You must specify a sample name for GBS
  import methods that don't include a sample
  name in the file to import"
19         exit 1
20     fi
21     SEAVE_URL="${SEAVE_URL}&sample_name=${sample_name}"
22 fi
23
24 SEAVE_URL="${SEAVE_URL}&import_type=GBS&method=${method}"
25 SEAVE_URL="${SEAVE_URL}&group=${group}"
26 OUTCOME=$(curl -s -S -L "${SEAVE_URL}")
27 echo "Seave reported: $OUTCOME"

```

Installing Seave from the Source Code on a Private Server or Local Computer

Seave is written in PHP and uses MySQL for its database and as such, it is meant to run in a LAMP (Linux, Apache, MySQL, PHP) stack. If you would like to run Seave on your own hardware rather than on AWS, you have two options.

The easier option is to download the AMI from AWS and run it as an image. This can be accomplished by:

1. Launch the AMI as an instance
2. Take a snapshot of the instance

3. Create a new EBS volume of the snapshot and attach it to the instance
4. SSH download the entire volume using dd, e.g. `ssh ubuntu@<instance ip> "sudo dd if=/dev/xvdf | gzip -1 -" | dd of=seave-image.gz`
5. Run the image as a virtual machine

The more difficult option is to clone the Seave source code repository from [GitHub](#) and then proceed to set up Apache, MySQL and PHP on a Linux or macOS machine. You will need to populate your MySQL server with each of the databases that Seave uses — you can find the SQL files required to do so in this repository under the Database Structures folder. Make sure to install **GEMINI** and **bedtools**, along with any other dependencies that are required to fix features that are broken. It's recommended to look at the file permissions and Apache/PHP configuration files in the AMI for making sure Seave has enough resources and is adequately secured.

Viewing Variants in IGV

All variants reported by Seave should be manually validated using IGV as variant callers and aligners often get it wrong. To help with this, Seave has a "Navigate in IGV" button below the results tables which, when clicked, brings up an "IGV" button for each variant on the far right of the results table. If the user has an open IGV window open with data loaded, clicking this button will open a new browser tab which will load a URL in an `<iframe>` that will force IGV to navigate to the location of the variant, the browser tab is closed after a second. This should work with a default installation of IGV but if it doesn't, ensure that the user has "Enable port 60151" ticked under the Advanced tab in the IGV preferences.

Setting Up SSL

By default, Seave is configured for SSL. The Seave AMI comes with a self-signed certificate which raises red flags in modern browsers and should be modified upon initial set-up of the server. The easiest way to do this is to obtain your own certificate from a trusted certificate authority and install it on your Seave server. To do this, copy the certificate file (ending in .pem) to `/etc/ssl/certs` and the key file (ending in .key) to `/etc/ssl/private`. Then, modify the following two lines in `/etc/apache2/sites-available/default-ssl.conf` to point to your certificate files:

```
1 SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
```

```
2 SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.  
key
```

Restart apache with `sudo service apache2 restart` and the new certificate should be applied immediately.

Alternatively, you may wish to use [Let's Encrypt](#) to obtain a free SSL certificate that can be automatically renewed by [certbot](#). You can find guides for how to do this with Apache on their websites.

Using a Separate GEMINI Databases Volume

It's often a good idea to house GEMINI databases on a different volume to the Seave webserver, such as on a separate EBS volume when using AWS. To do this, set up your volume and mount it on the webserver, then go to the `config.ini` file in the web root and modify the path in `db_dir=` under the `[gemini]` heading to point to the mount point.

Using a Separate MySQL Database Server

If you'd like to use a separate MySQL database server for Seave, such as an RDS server on AWS, go to the `config.ini` file in the web root and modify the particulars in `server=`, `user=` and `password=` under the `[mysql]` heading. If you do this, make sure that the latency between your webserver and MySQL server are as small as possible, do not use a MySQL server located on a different continent to the webserver!

Automated Testing Using Selenium

The Seave code repository contains a 'test' folder containing a `Selenium.pl` Perl script for conducting automated end-to-end tests of Seave. The "Testing Documentation.md" file in the same directory lists details for how to run automated tests in either a headless browser or locally. This code is hard coded to the test cases used for internal development so will need significant modification to be useful externally.

Keeping Seave Up To Date Using Automated Deployments

If you'd like to keep Seave updated to make use of new features, you can use a product like Atlassian Bamboo to build Seave on a development server to be deployed to a production server. We use the following set of tasks on the public GitHub code base to build Seave:

1. Checkout the Seave code repository from [GitHub](#)
2. Tar the current deployment on the dev server to the home directory, delete the current deployment and recreate the web root with the correct permissions
3. Tar and gzip the source code checkout and copy it to the development server
4. Extract the source code checkout to the web root and set permissions
5. Overwrite the default config.ini with your own details
6. Run tests
7. Delete the new code and restore the previous deployment from the tar in the home directory to get back to where you started

Updating Annotation Sources

Seave contains a number of annotation sources for linking variants to observations and predictions of pathogenicity stored in databases and *in silico* scores. As these databases and scores are constantly improving, they need to be periodically updated so Seave contains the most recent information. Annotation sources live in the MySQL database and are updated in a semi-automatic way using SQL files, a Perl script and a few MySQL queries. These files are provided for you in a separate Seave annotations repository [here](#). You will need the correct download from the annotation source you are updating, as will be outlined in annotation-specific sections below. The general procedure is:

1. SSH in to the webserver and run: `mysql -h localhost -P 3306 -u root -p < /path/to/<annotation>_create_database.sql` — this will create a new temporary database with the correct structure of tables to house the annotation
2. Run the relevant Perl script on the annotation download file to import it into the new temporary database — this is typically in the format of `perl /path/to/<annotation>.pl /path/to/<annotation download file> <db host> <db username> <db password>`
3. Once the script reports success, log in to the MySQL database and a run few queries to check that the new annotation is as expected with regards to the total number of entries and to check that a few specific well-known values are still present

4. Run the second SQL file ending in <annotation>_replace_database.sql in the same way as in the first step — this will archive the current production annotation database into a backup “_OLD” database set the new database as the one active in production
5. From the MySQL database, run a query to add the annotation update to the record of Seave’s annotation updates — this will bump the version of the annotation reported for all queries hereafter and on the “Data Sources” page and also add to the log of annotation updates for auditing purposes

OMIM

Create the new temporary database

```
mysql -h <db host> -P 3306 -u root -p < /path/to/OMIM_create_database.sql
```

Import the OMIM download file

```
perl /path/to/OMIM.pl /path/to/genemap.txt <db host> <db username>
<db password>
```

Check for number of rows per OMIM table, new vs old

```

1 SELECT
2     (SELECT COUNT(*) FROM OMIM.omim_disorders) AS
3     CURRENT_omim_disorders_count,
4     (SELECT COUNT(*) FROM OMIM_NEW.omim_disorders) AS
5     NEW_omim_disorders_count,
6     (SELECT COUNT(*) FROM OMIM.
7     omim_disorders_to_omim_numbers) AS
8     CURRENT_omim_disorders_to_omim_numbers_count,
9     (SELECT COUNT(*) FROM OMIM_NEW.
10    omim_disorders_to_omim_numbers) AS
11    NEW_omim_disorders_to_omim_numbers_count,
12    (SELECT COUNT(*) FROM OMIM.omim_genes) AS
13    CURRENT_omim_genes_count,
14    (SELECT COUNT(*) FROM OMIM_NEW.omim_genes) AS
15    NEW_omim_genes_count,
16    (SELECT COUNT(*) FROM OMIM.omim_number_to_gene) AS
17    CURRENT_omim_number_to_gene_count,
18    (SELECT COUNT(*) FROM OMIM_NEW.omim_number_to_gene)
19    AS NEW_omim_number_to_gene_count,
20    (SELECT COUNT(*) FROM OMIM.omim_numbers) AS
21    CURRENT_omim_numbers_count,
22    (SELECT COUNT(*) FROM OMIM_NEW.omim_numbers) AS
23    NEW_omim_numbers_count
24 ;
```

Check the results for 2 very well known genes match

```
1 SELECT
2     OMIM.omim_genes.gene_name ,
3     OMIM.omim_numbers.omim_number ,
4     OMIM.omim_numbers.omim_title ,
5     OMIM.omim_numbers.omim_status ,
6     OMIM.omim_disorders.omim_disorder
7 FROM
8     OMIM.omim_numbers
9 INNER JOIN OMIM.omim_number_to_gene ON OMIM.
    omim_number_to_gene.omim_number = OMIM.omim_numbers.
    omim_number
10 INNER JOIN OMIM.omim_genes ON OMIM.omim_genes.gene_id =
    OMIM.omim_number_to_gene.gene_id
11 INNER JOIN OMIM.omim_disorders_to_omim_numbers ON OMIM.
    omim_disorders_to_omim_numbers.omim_number = OMIM.
    omim_numbers.omim_number
12 INNER JOIN OMIM.omim_disorders ON OMIM.
    omim_disorders_to_omim_numbers.disorder_id = OMIM.
    omim_disorders.disorder_id
13 WHERE
14     OMIM.omim_genes.gene_name IN('BRCA1', 'BRCA2');
15
16 SELECT
17     OMIM_NEW.omim_genes.gene_name ,
18     OMIM_NEW.omim_numbers.omim_number ,
19     OMIM_NEW.omim_numbers.omim_title ,
20     OMIM_NEW.omim_numbers.omim_status ,
21     OMIM_NEW.omim_disorders.omim_disorder
22 FROM
23     OMIM_NEW.omim_numbers
24 INNER JOIN OMIM_NEW.omim_number_to_gene ON OMIM_NEW.
    omim_number_to_gene.omim_number = OMIM_NEW.
    omim_numbers.omim_number
25 INNER JOIN OMIM_NEW.omim_genes ON OMIM_NEW.omim_genes.
    gene_id = OMIM_NEW.omim_number_to_gene.gene_id
26 INNER JOIN OMIM_NEW.omim_disorders_to_omim_numbers ON
    OMIM_NEW.omim_disorders_to_omim_numbers.omim_number =
    OMIM_NEW.omim_numbers.omim_number
27 INNER JOIN OMIM_NEW.omim_disorders ON OMIM_NEW.
    omim_disorders_to_omim_numbers.disorder_id = OMIM_NEW
    .omim_disorders.disorder_id
28 WHERE
29     OMIM_NEW.omim_genes.gene_name IN('BRCA1', 'BRCA2');
```

Archive and replace the current production version

mysql -h <db host> -P 3306 -u root -p < /path/to/OMIM_replace_production.sql

Insert a row into the annotation database

```
1 INSERT INTO ANNOTATIONS.annotation_updates
2     (annotation_id, version, update_method_id,
3      update_time)
4 VALUES (
5     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.
6      annotations WHERE ANNOTATIONS.annotations.name =
7      'OMIM'),
8     '<OMIM release date>',
9     (SELECT ANNOTATIONS.update_methods.id FROM
10      ANNOTATIONS.update_methods WHERE ANNOTATIONS.
11      update_methods.method_name = 'automatic'),
12     now())
13 );
```

If something goes wrong and old annotations need to be restored

```
1 RENAME TABLE OMIM_OLD.omim_disorders TO OMIM.
2   omim_disorders, OMIM_OLD.
3   omim_disorders_to_omim_numbers TO OMIM.
4   omim_disorders_to_omim_numbers, OMIM_OLD.omim_genes
5   TO OMIM.omim_genes, OMIM_OLD.omim_number_to_gene TO
6   OMIM.omim_number_to_gene, OMIM_OLD.omim_numbers TO
7   OMIM.omim_numbers;
```

COSMIC

Create the new temporary database

```
mysql -h <db host> -P 3306 -u root -p < /path/to/COSMIC_create_database.sql
```

Import the COSMIC download files

```
perl /path/to/COSMIC.pl /path/to/CosmicCodingMuts.vcf
/path/to/CosmicMutantExport.tsv <db host> <db username> <db password>
```

Check for number of rows per COSMIC table, new vs old

```
1 SELECT
2     (SELECT COUNT(*) FROM COSMIC.variants) AS
3     CURRENT_variants,
4     (SELECT COUNT(*) FROM COSMIC_NEW.variants) AS
5     NEW_variants,
6     (SELECT COUNT(*) FROM COSMIC.cosmic_numbers) AS
7     CURRENT_cosmic_numbers,
8     (SELECT COUNT(*) FROM COSMIC_NEW.cosmic_numbers) AS
9     NEW_cosmic_numbers,
```

```

6      (SELECT COUNT(*) FROM COSMIC.
      cosmic_number_to_variant) AS
      CURRENT_cosmic_number_to_variant,
7      (SELECT COUNT(*) FROM COSMIC_NEW.
      cosmic_number_to_variant) AS
      NEW_cosmic_number_to_variant
8 ;

```

Check the results for 2 variants match

```

1 SELECT
2     COSMIC.variants.chr,
3     COSMIC.variants.pos,
4     COSMIC.variants.ref,
5     COSMIC.variants.alt,
6     COSMIC.cosmic_numbers.cosmic_number,
7     COSMIC.cosmic_numbers.cosmic_count,
8     COSMIC.cosmic_numbers.cosmic_primary_site,
9     COSMIC.cosmic_numbers.cosmic_primary_histology
10 FROM
11     COSMIC.variants
12 INNER JOIN COSMIC.cosmic_number_to_variant ON COSMIC.
      cosmic_number_to_variant.variant_id = COSMIC.variants
      .id
13 INNER JOIN COSMIC.cosmic_numbers ON COSMIC.cosmic_numbers
      .cosmic_number = COSMIC.cosmic_number_to_variant.
      cosmic_number
14 WHERE
15     chr = '7' AND pos = '140453136' AND ref = 'A' AND
      alt = 'T'
16 ;
17
18 SELECT
19     COSMIC_NEW.variants.chr,
20     COSMIC_NEW.variants.pos,
21     COSMIC_NEW.variants.ref,
22     COSMIC_NEW.variants.alt,
23     COSMIC_NEW.cosmic_numbers.cosmic_number,
24     COSMIC_NEW.cosmic_numbers.cosmic_count,
25     COSMIC_NEW.cosmic_numbers.cosmic_primary_site,
26     COSMIC_NEW.cosmic_numbers.cosmic_primary_histology
27 FROM
28     COSMIC_NEW.variants
29 INNER JOIN COSMIC_NEW.cosmic_number_to_variant ON
      COSMIC_NEW.cosmic_number_to_variant.variant_id =
      COSMIC_NEW.variants.id
30 INNER JOIN COSMIC_NEW.cosmic_numbers ON COSMIC_NEW.
      cosmic_numbers.cosmic_number = COSMIC_NEW.
      cosmic_number_to_variant.cosmic_number

```

```

31 WHERE
32     chr = '7' AND pos = '140453136' AND ref = 'A' AND
        alt = 'T'
33 ;

```

Archive and replace the current production version

mysql -h <db host> -P 3306 -u root -p < /path/to/COSMIC_replace_production.sql

Insert a row into the annotation database

```

1 INSERT INTO ANNOTATIONS.annotation_updates
2     (annotation_id, version, update_method_id,
        update_time)
3 VALUES (
4     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.
        annotations WHERE ANNOTATIONS.annotations.name =
        'COSMIC'),
5     '<COSMIC version>',
6     (SELECT ANNOTATIONS.update_methods.id FROM
        ANNOTATIONS.update_methods WHERE ANNOTATIONS.
        update_methods.method_name = 'automatic'),
7     now())
8 );

```

MITOMAP

Create the new temporary database

mysql -h <db host> -P 3306 -u root -p < /path/to/MITOMAP_create_database.sql

Import the MITOMAP download file

perl /path/to/MITOMAP.pl /path/to/disease.vcf /path/to/polymorphisms.vcf
 <db host> <db username> <db password>

Check for number of rows, new vs old

```

1 SELECT
2     (SELECT COUNT(*) FROM MITOMAP.mitomap) AS CURRENT\
        _mitomap_count,
3     (SELECT COUNT(*) FROM MITOMAP\_NEW.mitomap) AS NEW\
        _mitomap_count
4 ;

```

Check the results for 2 variants match

```

1 SELECT * FROM MITOMAP.mitomap WHERE chr = 'MT' AND pos =
   '961' AND ref = 'T' AND alt = 'C';
2
3 SELECT * FROM MITOMAP_NEW.mitomap WHERE chr = 'MT' AND
   pos = '961' AND ref = 'T' AND alt = 'C';

```

Archive and replace the current production version

```
mysql -h <db host> -P 3306 -u root -p < /path/to/MITOMAP_replace_production.sql
```

Insert a row into the annotation database

```

1 INSERT INTO ANNOTATIONS.annotation_updates
2     (annotation_id, version, update_method_id,
3      update_time)
4 VALUES (
5     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.
6      annotations WHERE ANNOTATIONS.annotations.name =
7      'MITOMAP'),
8     '<MITOMAP release date>',
9     (SELECT ANNOTATIONS.update_methods.id FROM
10      ANNOTATIONS.update_methods WHERE ANNOTATIONS.
11      update_methods.method_name = 'automatic'),
12     now())
13 );

```

RVIS

Create the new temporary database

```
mysql -h <db host> -P 3306 -u root -p < /path/to/RVIS_create_database.sql
```

Import the RVIS download file

```
perl /path/to/RVIS.pl /path/to/RVIS.txt <db host> <db username> <db
password>
```

Check for number of rows, new vs old

```

1 SELECT
2     (SELECT COUNT(*) FROM RVIS.rvis) AS
3     CURRENT_RVIS_count,
4     (SELECT COUNT(*) FROM RVIS_NEW.rvis) AS
5     NEW_RVIS_count
6 ;

```

Check the results for 2 genes match

```

1 SELECT * FROM RVIS.rvis WHERE gene = 'TET2';
2
3 SELECT * FROM RVIS_NEW.RVIS WHERE gene = 'TET2';

```

Archive and replace the current production version

mysql -h <db host> -P 3306 -u root -p < /path/to/RVIS_replace_production.sql

Insert a row into the annotation database

```

1 INSERT INTO ANNOTATIONS.annotation_updates
2     (annotation_id, version, update_method_id,
3      update_time)
4 VALUES (
5     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.
6      annotations WHERE ANNOTATIONS.annotations.name =
7      'RVIS'),
8     '<RVIS release version/date>',
9     (SELECT ANNOTATIONS.update_methods.id FROM
10      ANNOTATIONS.update_methods WHERE ANNOTATIONS.
11      update_methods.method_name = 'automatic'),
12     now())
13 );

```

ClinVar

Create the new temporary database

mysql -h <db host> -P 3306 -u root -p < /path/to/CLINVAR_create_database.sql

Import the ClinVar download file

perl /path/to/CLINVAR.pl /path/to/clinvar_<date>.vcf <db host> <db
username> <db password>

Check for number of rows, new vs old

```

1 SELECT
2     (SELECT COUNT(*) FROM CLINVAR.clinvar) AS
3     CURRENT_clinvar_count,
4     (SELECT COUNT(*) FROM CLINVAR_NEW.clinvar) AS
5     NEW_clinvar_count
6 ;

```

Check the results for 2 variants match

```

1 SELECT
2     CLINVAR.clinvar.chr,

```



```

3      CLINVAR.clinvar.position,
4      CLINVAR.clinvar.ref,
5      CLINVAR.clinvar.alt,
6      CLINVAR.clinvar.clinvar_rs,
7      CLINVAR.clinvar.clinsig,
8      CLINVAR.clinvar.clintrait
9 FROM
10     CLINVAR.clinvar
11 WHERE
12     CLINVAR.clinvar.chr = '1' AND CLINVAR.clinvar.
        position = '201331068' AND CLINVAR.clinvar.ref =
        'A' AND CLINVAR.clinvar.alt = 'G'
13 ;
14
15 SELECT
16     CLINVAR_NEW.clinvar.chr,
17     CLINVAR_NEW.clinvar.position,
18     CLINVAR_NEW.clinvar.ref,
19     CLINVAR_NEW.clinvar.alt,
20     CLINVAR_NEW.clinvar.clinvar_rs,
21     CLINVAR_NEW.clinvar.clinsig,
22     CLINVAR_NEW.clinvar.clintrait
23 FROM
24     CLINVAR_NEW.clinvar
25 WHERE
26     CLINVAR_NEW.clinvar.chr = '1' AND CLINVAR_NEW.
        clinvar.position = '201331068' AND CLINVAR_NEW.
        clinvar.ref = 'A' AND CLINVAR_NEW.clinvar.alt =
        'G'
27 ;

```

Archive and replace the current production version

mysql -h <db host> -P 3306 -u root -p < /path/to/CLINVAR_replace_production.sql

Insert a row into the annotation database

```

1 INSERT INTO ANNOTATIONS.annotation_updates
2     (annotation_id, version, update_method_id,
3      update_time)
4 VALUES (
5     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.
        annotations WHERE ANNOTATIONS.annotations.name =
        'ClinVar'),
6     '<ClinVar release date>',
7     (SELECT ANNOTATIONS.update_methods.id FROM
        ANNOTATIONS.update_methods WHERE ANNOTATIONS.
        update_methods.method_name = 'automatic'),
8     now())
9 ;

```

COSMIC CGC

Create the new temporary database

```
mysql -h <db host> -P 3306 -u root -p < /path/to/COSMIC_CGC_create_database.sql
```

Import the COSMIC CGC download file

```
perl /path/to/COSMIC_CGC.pl /path/to/<COSMIC CGC export>.tsv <db  
host> <db username> <db password>
```

Check for number of rows, new vs old

```
1 SELECT  
2     (SELECT COUNT(*) FROM COSMIC_CGC.cosmic_cgc) AS  
3     CURRENT_cosmic_cgc_count,  
4     (SELECT COUNT(*) FROM COSMIC_CGC_NEW.cosmic_cgc) AS  
5     NEW_cosmic_cgc_count  
6 ;
```

Check the results for 2 genes match

```
1 SELECT * FROM COSMIC_CGC.cosmic_cgc WHERE gene = 'BRCA1';  
2  
3 SELECT * FROM COSMIC_CGC_NEW.cosmic_cgc WHERE gene = '  
4     BRCA1';
```

Archive and replace the current production version

```
mysql -h <db host> -P 3306 -u root -p < /path/to/COSMIC_CGC_replace_production.sql
```

Insert a row into the annotation database

```
1 INSERT INTO ANNOTATIONS.annotation_updates  
2     (annotation_id, version, update_method_id,  
3     update_time)  
4 VALUES (  
5     (SELECT ANNOTATIONS.annotations.id FROM ANNOTATIONS.  
6     annotations WHERE ANNOTATIONS.annotations.name =  
7     'COSMIC CGC'),  
8     '<COSMIC CGC release date>',  
9     (SELECT ANNOTATIONS.update_methods.id FROM  
10    ANNOTATIONS.update_methods WHERE ANNOTATIONS.  
11    update_methods.method_name = 'automatic'),  
12    now())  
13 ;
```

Seave Logging

Seave logs all user interactions where either a state change occurs (e.g. adding, removing or changing something) or a query is performed. This allows access auditing/statistics, debugging issues and interrogation of malicious events. Every event is logged with the ID and email of the user who performed it, a description of the event with relevant details, the IP address of the user and the time when the event occurred.

This logging occurs to Seave's MySQL LOGGING database in the table `website_events`. There is no interface for viewing these logs within Seave itself so you will need to run a database query by connecting to the MySQL database directly. If your MySQL database is on the same server as the webserver (e.g. if you used the AMI), SSH into the webserver and connect to the MySQL database with `mysql -h localhost -P 3306 -u root -p`, otherwise connect to the MySQL database server directly with the same command but with "localhost" replaced with the URL/IP of your server. Once connected, execute the following query to get the latest 100 events:

```
1 SELECT * FROM LOGGING.website_events ORDER BY id DESC
   LIMIT 100;
```

Further to the above, Seave also stores the results of all queries. When a GEMINI or GBS query is executed, the results are saved to a .tsv file; all the results pages do is read this TSV file and display it in a table. This is why sending the URL from the results page to someone else will mean they can see the the same results but can't query the database without logging in. You can find the query history by connecting to the webserver via SSH and navigating to `/var/www/html/temp`. Here, you will see a large number of files starting with the year, month, day and time of the query and ending in `*.tsv`, `*.gem` and `*.err`. The first is the variants results file for the query, the second is the GEMINI command executed if the query was for short variants and the third is for any errors that occurred. You may want to periodically tar/gzip these and archive them so they don't take up too much space.

Gene Name Validation

When genes are added to lists (see the [Add Gene\(s\) to List\(s\)](#) section), or when a query is performed with a custom gene list, Seave will validate the genes entered against an internal list of valid genes. The purpose of this is to prevent genes being searched that cannot be annotated as impacted in the GEMINI databases of variants. The process of creating the GEMINI database is going to dictate the genes that can be annotated, specifically, the

upstream annotation of your VCF file by one of the two supported tools Variant Effect Predictor (VEP) or snpEff. Different versions of these tools will use different gene names for annotation so it will be up to you to determine the list used by your bioinformatics pipeline and add the same list to your Seave installation.

The gene list Seave uses to validate genes is located in:

```
/var/www/html/assets/validation_gene_list.txt
```

By default, Seave ships with the Ensembl 75 list. To change this list, copy a new text file to the same location in your Seave installation. Each gene should be on a separate line with no other whitespace.