

GECKO-MGV

Combining strengths for multi-genome visual
analytics comparison

User Manual

Version v2: 17th December 2018.



UNIVERSIDAD
DE MÁLAGA



Developed by:

Sergio Diaz-Del-Pino
Pablo Rodriguez-Brazzarola
Esteban Perez-Wohlfeil
Oswaldo Trelles

Report incidences to:

ortrelles@ac.uma.es

Or contact:

www.bitlab.es

Introduction

GeckoMGV is visual display for pairwise-comparisons capable of working with more than one sequence at the same time. It works through the web and allows the user to navigate dynamically through the comparisons, filtering and consulting information.

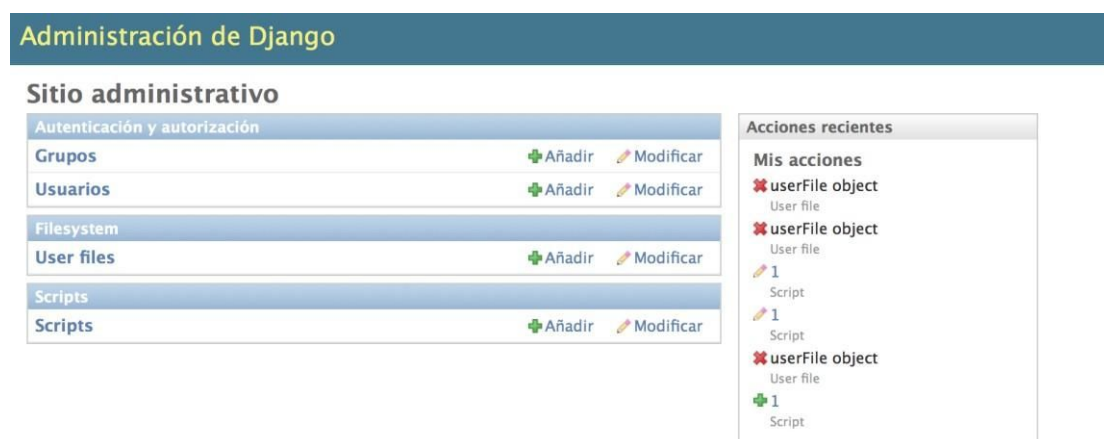
In this new versión, due to a backend implementation, it is available the possibility to execute services that process the working data.

We proceed to explain how to register and execute this services.

1.- Backend Administration

To access the administration panel of this application go to the following link: <http://pistacho.ac.uma.es/admin>

There we can see the administrator panel that contains the models within our database. At first, both users and scripts will be added through this interface.



Administration panel


1.1 Adding users

To add a user we must access the panel '*Usuarios*' and press on the '*Añadir usuario*' button.



The screenshot shows the 'Administración de Django' interface. The top navigation bar includes 'Inicio > Autenticación y autorización > Usuarios'. The main heading is 'Escoja usuario a modificar'. Below it is a search bar and a table of users. The table has columns: 'Nombre de usuario', 'Dirección de correo electrónico', 'Nombre', 'Apellidos', and 'Es staff'. One user, 'yeyo', is listed with the email 'sergiodiazdp@gmail.com' and is marked as staff. A sidebar on the right contains filters for 'Por es staff', 'Por es superusuario', and 'Por activo', each with 'Todo', 'Sí', and 'No' options. A button 'Añadir usuario +' is in the top right.

Users panel



The screenshot shows the 'Añadir usuario' form in the Django administration interface. The form has three main input fields: 'Nombre de usuario:' with the value 'guest', 'Contraseña:', and 'Contraseña (confirmación):'. Below the first field is a note: 'Requerido. 30 caracteres o menos. Letras, dígitos y @/./+/-/_ solamente.' Below the password fields is a note: 'Introduzca la misma contraseña que arriba, para verificación.' At the bottom are three buttons: 'Grabar y añadir otro', 'Grabar y continuar editando', and 'Grabar'.

Adding user 'guest/guest'



This screenshot is similar to the first one, showing the 'Usuarios' panel. The table now lists two users: 'guest' and 'yeyo'. The 'guest' user is marked as not staff (indicated by a red minus icon), while 'yeyo' is marked as staff (indicated by a green plus icon). The sidebar filters and the 'Añadir usuario +' button remain the same.

User added

1.2 Adding scripts

A script is composed of:

- An executable (if possible, to be compiled on the server)
- A form defined in Django (Python)
- A database registry

1.2.1 Executables

In this example we will use the program '*kmersFreq*' that returns a list of kmers given a sequence.

We must add the script to the path '*/var/www/GeckoMGV/media/scripts/*' of the server.

1.2.2 Form in Django – Python

Our applications requires to have a form for each added script. This form must have a field for each parameter needed by the executable.

To do that we must access the file '*forms.py*' of our app '*scripts*' folder located at: '*GeckoMGV/scripts/forms.py*' and define the fields according to the Django standard that can be reviewed here:

<https://docs.djangoproject.com/en/1.8/topics/forms/> In our example, the program '*kmersFreq*' requires of:

- A file field of a sequence.
- An integer field for the 'K' size of the kmer.
- A text field 'Fullout' that indicates the output format.

In our case we have defined the form '*kmersForm*' with the previously described fields:

```
class kmersForm(forms.Form):
    filename = forms.ChoiceField(label="Filename", widget=forms.Select(attrs={'class': 'selector'}))
    K = forms.CharField(label='K', max_length=1)
    fullOut = forms.CharField(label='fullOut', max_length=1)
```

An init method will also need to be declared if we want to use file type fields, to exclusively retrieve the logged user's files.

```
def __init__(self, *args, **kwargs):
    self.user = kwargs.pop('user', None)
    self.request = kwargs.pop('request', None)
    super(kmersForm, self).__init__(*args, **kwargs)
    print self.user
    print userFile.objects.filter(user=self.user)
    self.fields['filename'] = forms.ChoiceField(choices=[(file.file.name, file.file.name)
                                                         for file in userFile.objects.filter(user=self.user)])
```

1.2.3 Database registry

Once we are done with the executable and the form, we can register the service in the database.

To achieve that, we must go navigate from the admin panel to '*Scripts*' > '*Añadir Nuevo Script*' and fill out the form that contains the following fields:

- **Name:** Service name
- **exeName:** Executables name
- **Path:** Executables path
- **Help:** Help/Information
- **Form:** Name of the defined Django form For our example, it should

look like this:

Administración de Django

Inicio > Scripts > Scripts > 1

Modificar script

| | |
|----------|---------------------------------------|
| Name: | Calculate Kmers |
| ExeName: | kmersFreq |
| Path: | GeckoMGV/media/scripts/ |
| Help: | Calculate Kmers from a sequence given |
| Form: | kmersForm |

✖ Eliminar

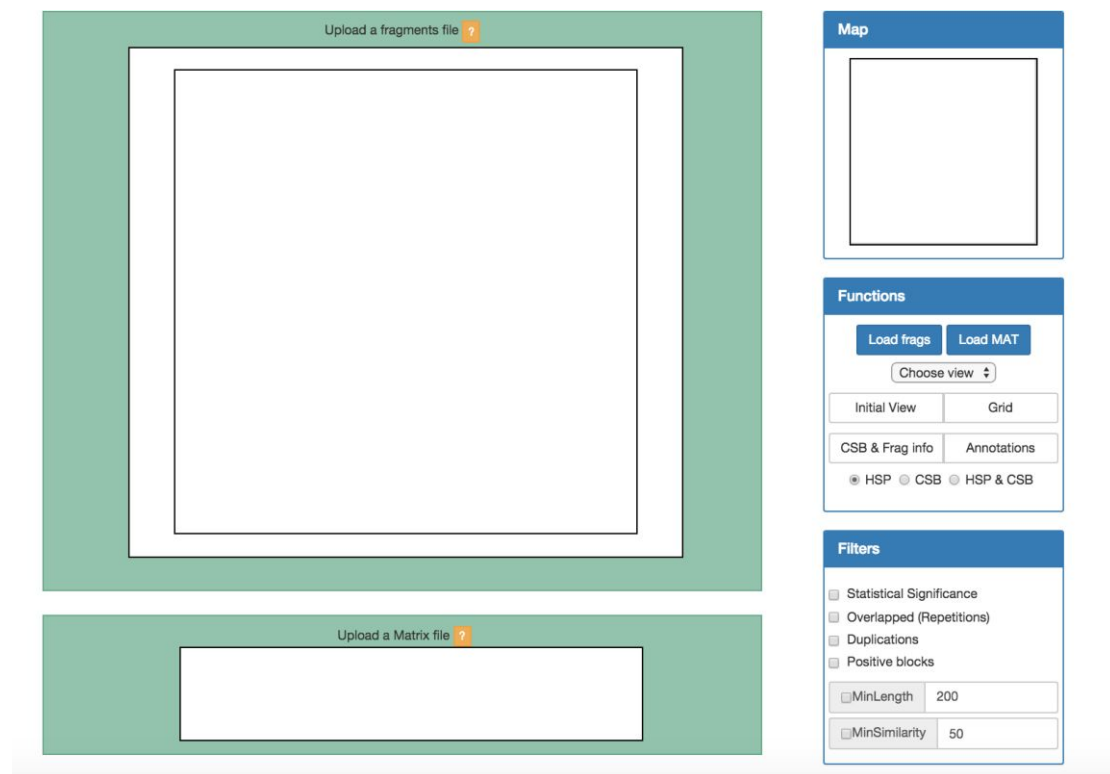
New service registry

Once this is done, the script can be executed by any registered user.

2. Client execution

To access the canvas display we must go into the following URL:
<http://pistacho.ac.uma.es>

The main page is the canvas display accompanied by its main functionalities.

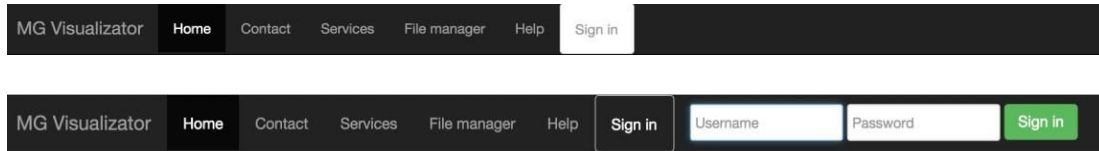


To make use of the new functions, the user must do the following:

- Login to the system
- Upload the necessary files for the execution of a service.
- Access the service and execute it

2.1 Login to the system

In the navigation bar we can find the *'Sign in'* button. When clicked a form drops down within the bar that allows us to introduce the login data:



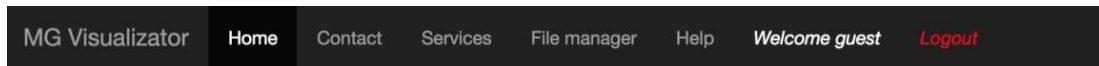
The screenshot shows a dark navigation bar with the following items: 'MG Visualizator', 'Home', 'Contact', 'Services', 'File manager', 'Help', and 'Sign in'. Below the 'Sign in' button, a form is displayed with two input fields labeled 'Username' and 'Password', and a green 'Sign in' button.

Upon completion, in this case as:

user: guest

pass: guest.

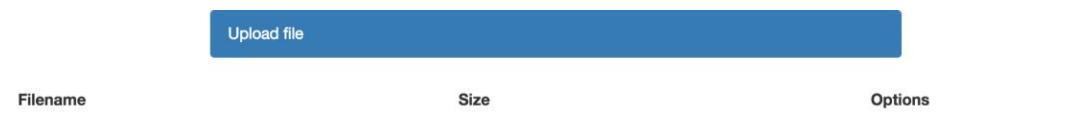
The navigation bar will display that we are logged into system, next to a *'Logout'* button:



The screenshot shows the same dark navigation bar, but now it includes 'Welcome guest' and a red 'Logout' button next to the 'Sign in' button.

2.2 File uploading

Once logged in the system, we access the *'File Manager'* tab where we can find our file system. There is also a button with the text *'Upload file'* that deploys a small menu that allows us to upload files.



The screenshot shows the 'File Manager' tab view. At the top, there is a blue 'Upload file' button. Below it is a table with three columns: 'Filename', 'Size', and 'Options'.



'File Manager' tab view



The screenshot shows the 'Upload file' dropdown menu. It has a blue header with the text 'Upload file'. Below the header, there is a section titled 'Select a file:' with a text input field containing 'Seleccionar archivo' and 'Ningún archivo seleccionado'. Below the input field is an 'Upload' button. The table with columns 'Filename', 'Size', and 'Options' is visible at the bottom of the dropdown.

Dropdown menu

We proceed by selecting our desired file to upload, for example '*sequence.fasta*' and we upload it to our server. After it is done, the file manager tab should look like this:

| Upload file | | |
|--------------------|-----------|---|
| Filename | Size | Options |
| sequence.fasta.txt | 436 bytes |   |


Once our file is in the server, we can work with it.

2.3 Service execution

Now we will executed the previously registered service. To do that we access the '*Services*' tab to view the list of our services. This list is dynamically constructed every time we access this tab. Therefore, new services will appear right after their registry:

Services

Calculate Kmers
Calculate Kmers from a sequence given



After pressing the '*Start*' button, a new view is generated with a form whose fields have been previously defined when the service was registered.

Filename:

/var/www/GeckoMGV/media/files/users/guest/sequence.fasta.txt

K: **fullOut:**

After filling out the fields and selecting our file, we can execute the service to obtain the results.

It is necessary to give a new format to the results since these are sent as plain/text even though they might be processed within the program.

Printing k-mers with at least one occurrence in alphabetical order AA 1 AC 0 AG 0 AT 1 CA 0 CC 0
CG 0 CT 0 GA 0 GC 1 GG 3 GT 1 TA 2 TC 1 TG 3 TT 1 Average frequency: 0.88, Standard
deviation: 0.99

Client-side obtained results

```
Printing k-mers with at least one occurrence in alphabetical order
AA\t1
AC\t0
AG\t0
AT\t1
CA\t0
CC\t0
CG\t0
CT\t0
GA\t0
GC\t1
GG\t3
GT\t1
TA\t2
TC\t1
TG\t3
TT\t1
Average frequency: 0.88, Standard deviation: 0.99
```

Server-side obtained results

Future work:

- Improved point & zoom.
- Allow file saving after executing a script:
 - o At the moment it is only possible to copy the results into a local machine.
- Make a list of useful predetermined scripts.
- Work with fragments
 - o Add the option to work directly from the contextual menú of the fragments.

