

# AvatoICV

[Introduction](#)

[Dependencies](#)

[Configuring Python](#)

[Getting the Installer](#)

[Installing AvatoICV](#)

[Updating AvatoICV](#)

[How To Run AvatoICV](#)

[An AvatoICV Session](#)

[High Level Overview](#)

[Start Screen](#)

[AvatoICV Scoring Session](#)

[Data Sources](#)

[Log in to Data Source](#)

[Choose Matrix or Dataset](#)

[Select Scoring Approach](#)

[Select Item To Score](#)

[Filtering Data](#)

[Exclude Poor Quality Images Screen](#)

[Segmentation](#)

[Orientation](#)

[Scoring Configuration](#)

[Partially or Fully Scored?](#)

[Training Set vs Scoring Set Nuances](#)

[Scoring](#)

[AvatoICV Results Review Screen](#)

[True Scoring Results](#)

[Algorithm Evaluation Results](#)

[Sorting Columns](#)

[Zooming In and Out of an Image](#)

[Confidence Threshold Control](#)

[Uploading Scores](#)

[Caveats, Limitations, and Assumptions](#)

[Dataset or Matrix Name Uniqueness](#)

[Training Data Sufficiency](#)

[Training Data Consistency](#)

[Image Orientation Consistency](#)

[How To Run AvatoICV Using Different Versions of Matlab](#)

[AvatoICV Internal Data Format](#)

[Adding New Algorithms into AvatoICV](#)

## Introduction

AvatolCV is a system that supports running computer vision and machine learning algorithms on images of biological specimens. The goal is to automate scoring biological characters, leveraging training data provided by the biologist.

AvatolCV is integrated via web services with two web-based biological image repositories:

Morphobank ([morphobank.org](http://morphobank.org)) – a taxon vs character matrix based system

BisQue ([http://bisque.iplantcollaborative.org/client\\_service](http://bisque.iplantcollaborative.org/client_service)) – an image centric repository for plant images

## Dependencies

64 bit Windows / 64 bit OSX (Mac)  
python 2.7.x  
Java 8 64 bit  
MATLAB R2015b  
8G RAM

## Configuring Python

Your system may come with python 2.7.x pre-installed.

To check on Mac, open up a terminal (Finder->Applications->Utilities->Terminal) and type "python --version". If present you will see something like : (\$ is the Mac prompt)

```
$python --version  
Python 2.7.10
```

If not present, install it from <https://www.python.org/downloads/release/python-2711/>. If "python --version" does not yield an answer, you will need to set the PATH directory to contain the directory with the python executable using this command:

```
$export PATH=$PATH:<location where python was installed>
```

for example

```
$export PATH=$PATH:/usr/bin/python
```

To check on a Windows machine, type "python --version"

```
C:\>python --version  
Python 2.7
```

If not present, install it from <https://www.python.org/downloads/release/python-2711/>. If "python --version" does not yield an answer, you will need to set the PATH directory to contain the directory with the python executable. (Control Panel -> System and Security -> System -> Advanced System Settings -> Environment Variables. In the System Variables window, select Path and click edit. Click once in the Variable Value field to ensure it is no longer highlighted. Right arrow to the end and type a semicolon followed by the path to where python was installed, for example if python was installed at C:\, you would append this to the path: ";C:\Python27". Then click OK, click OK, click OK. Now open up a new command shell and "python --version" should yield "Python 2.7")

## Getting the Installer

1. Got to <https://github.com/AVATOL/AvatolCVInstaller>
2. Click the button that says "Download ZIP"
3. unzip the file and then proceed with the instructions in the next section

## Installing AvatolCV

To run it, open up a command shell on Mac and type:

```
$cd whereTheFileWasUnzippedTo  
$python updateAvatolCV.py install_root
```

or on Windows

```
C:>cd whereTheFileWasUnzippedTo  
C:\whereTheFileWasUnzippedTo>python updateAvatolCV.py install_root
```

...where install\_root is replaced by your choice of what directory you want to install into. The installer will create a subdirectory called 'avatol\_cv' and place the system under that. So, for example, you specify

```
C:\whereTheFileWasUnzippedTo>python updateAvatolCV.py C:\someDir
```

...then c:\someDir\avatol\_cv will contain all the files for AvatolCV.

## Updating AvatolCV

If there is a bug fix and you need to pull in the updated system, merely run the same command that you used to install it initially.

```
python updateAvatolCV.py install_root
```

## How To Run AvatolCV

Once the files are in place, you would do:

```
cd C:\someDir\avatol_cv\java\lib
java -jar avatol_cv.jar
```

...to start the system

or on Mac

```
cd /someDir/avatol_cv/java/lib
java -jar avatol_cv.jar
```

NOTE - before starting an AvatolCV Session, it is recommended that you read the sections titled:

- An AvatolCV Session
- AvatolCV Scoring Session
- AvatolCV Results Review Screen
- Caveats, Limitations, and Assumptions

...Iso that you will know what to expect and are aware of assumptions made by AvatolCV and certain limitations of the system



# An AvatolCV Session

## High Level Overview

Before going into the details on each screen, here is an overview of the algorithm pipelines (sequences of algorithms) that work together as of this writing (May 17 2016). Since AvatolCV is a research project, the pipelines reflect the proof-of-concept nature of the work.

| Pipeline   | Platform        | Segmentation        | Orientation   | Scoring             |
|------------|-----------------|---------------------|---------------|---------------------|
| leaf       | Mac             | basicSegmenter      | basicOrienter | shapeTextureScoring |
| batskull   | Mac,<br>Windows | skip                | skip          | partsScoring        |
| nematocyst | Mac,<br>Windows | higClutterSegmenter | NA            | NA                  |

AvatolCV has been designed to allow other algorithms to be plugged into it without modifying the system. (See “Adding New Algorithms to AvatolCV”) This means depending on which platform you are running on, certain algorithms will be available.

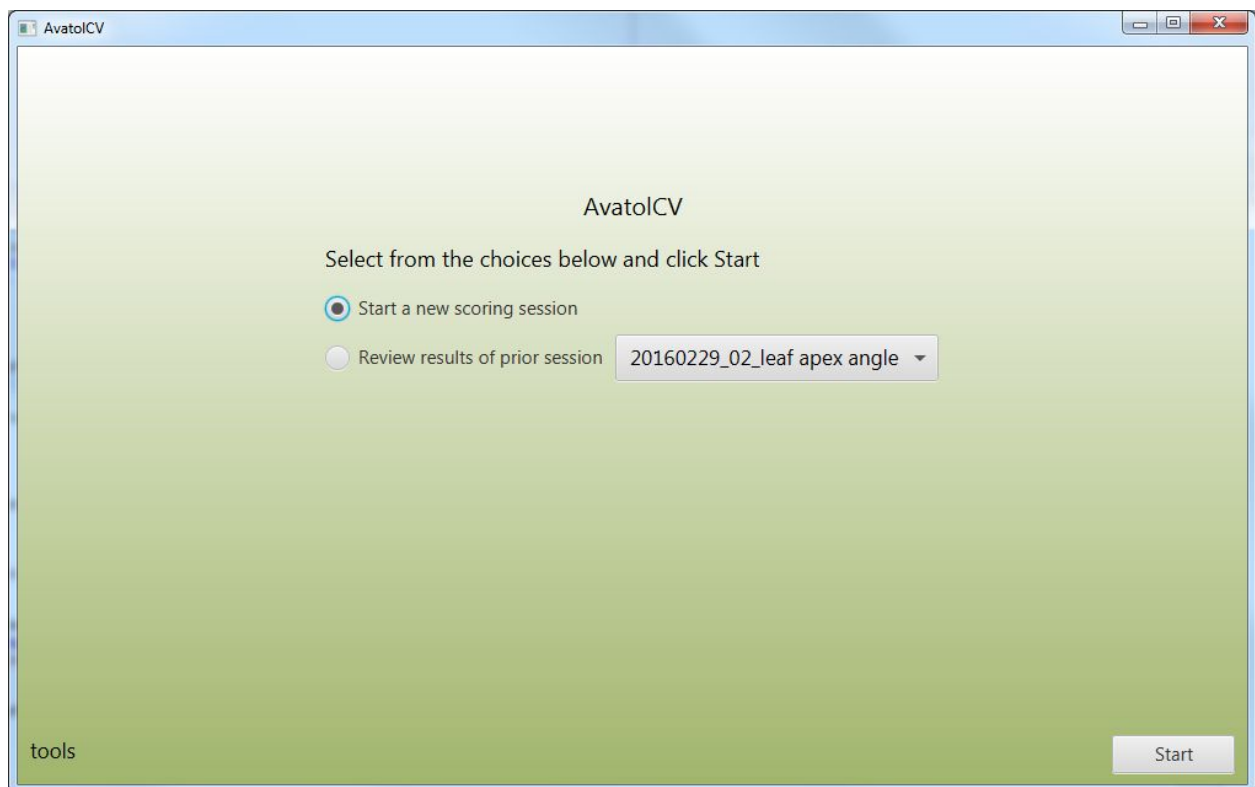
## Start Screen

The start screen of AvatoICV offers two choices: starting a new scoring session or reviewing results from a prior session. Prior sessions are named by the following pattern:

<date>\_<one-up number>\_<character name>

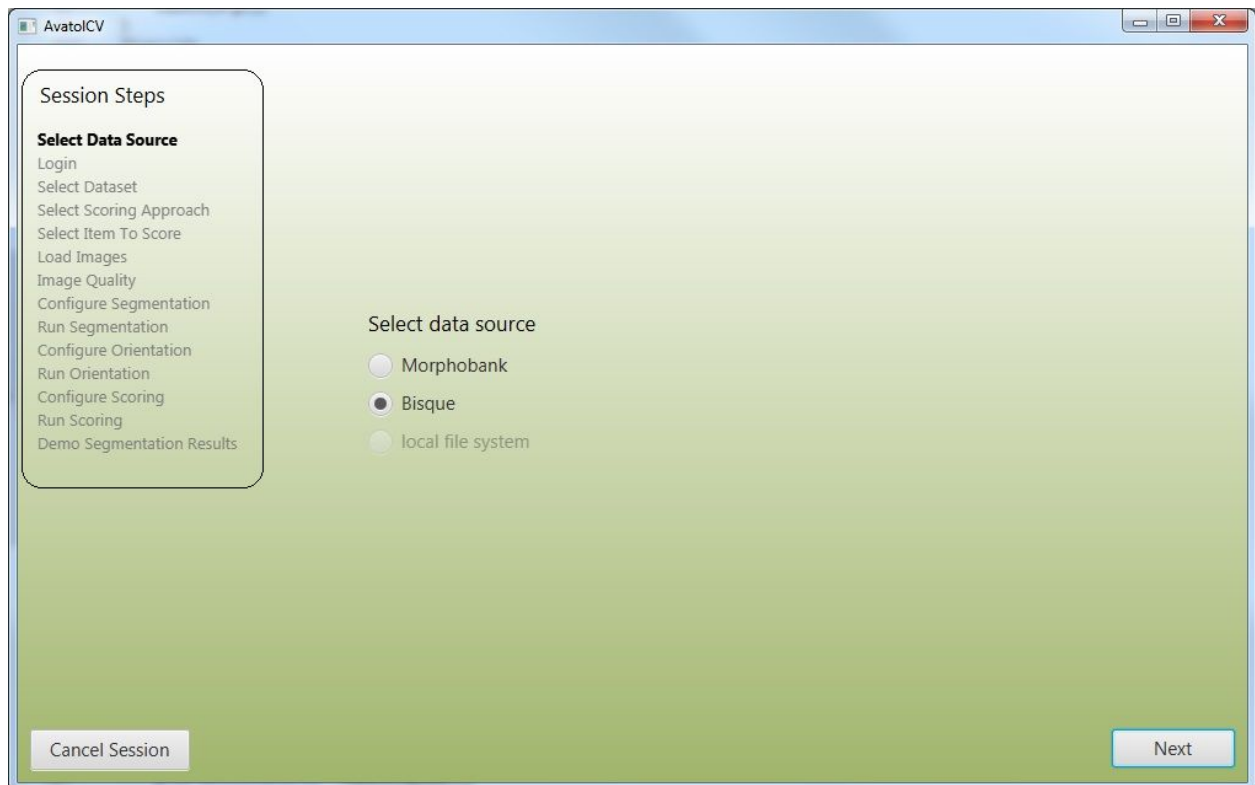
Date is of the form year-month-day. Date is assigned when the run is started, not when finished (in case it runs past the midnight threshold). One-up number is needed in case there are more than one scoring sessions for a particular character on a particular day.

For example, “20160415\_02\_leaf apex angle” refers to the 2nd scoring run of the character “leaf apex angle” on April 15, 2016.



# AvatoICV Scoring Session

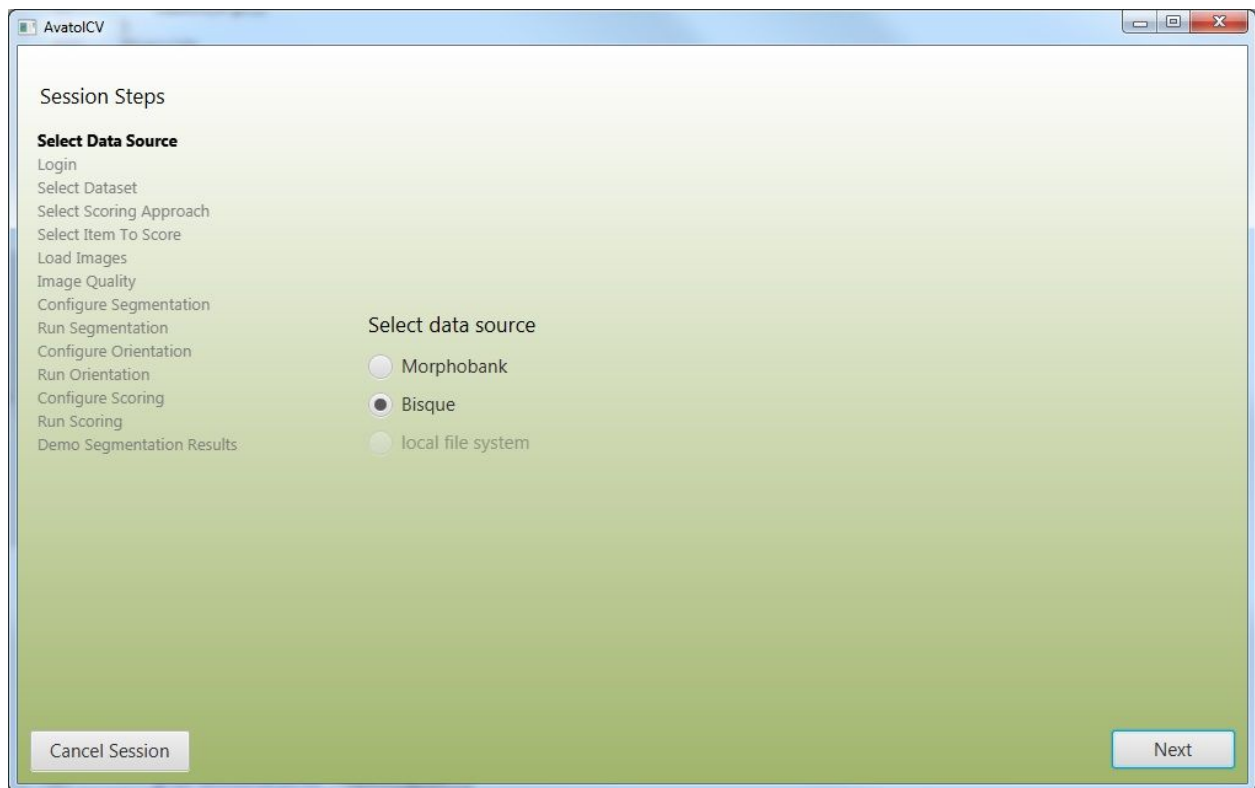
Once “new scoring session” is chosen, the remaining screens show a list of steps (circled below) for the session on the left, and where in that sequence of steps the current screen lies. The greyed out items have not been reached, the black items have been completed, and the bolded item is the current screen.





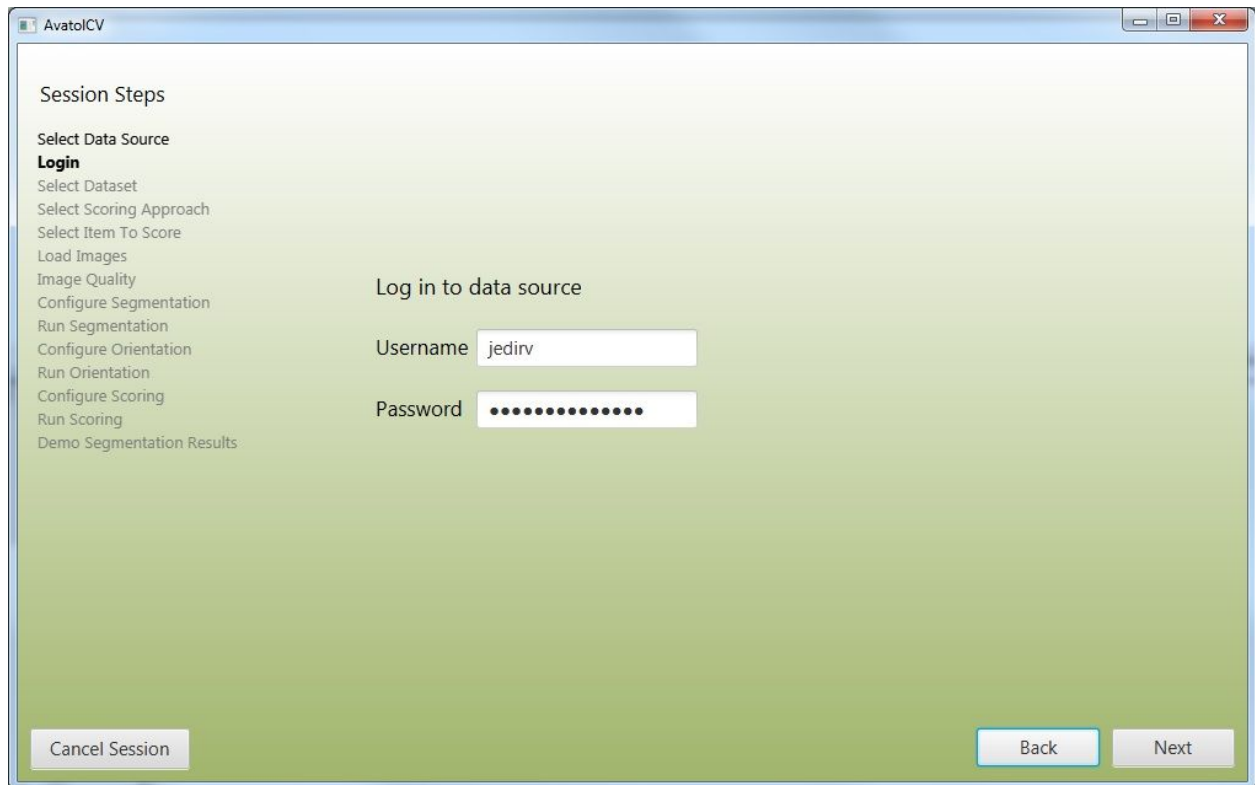
## Data Sources

AvatolCV is integrated using web services apis with Morphobank and BisQue. If either of these is selected, the DataSource component of AvatolCV will be interacting with the appropriate system. There is also an option for choosing “local file system”. “local file system” will work only if the dataset (images and metadata) are formatted and arranged in AvatolCV’s internal data format. (See AvatolCV Internal Data Format section for details). *As of this writing (5/17/2016) a “local file system” run should work except that the functionality to save results has not yet been implemented.*



## Log in to Data Source

At this screen, the user should specify their username and password for either Morphobank or BisQue, depending on which they chose.



The screenshot shows the AvatoICV application window. On the left, a 'Session Steps' list includes: Select Data Source, **Login**, Select Dataset, Select Scoring Approach, Select Item To Score, Load Images, Image Quality, Configure Segmentation, Run Segmentation, Configure Orientation, Run Orientation, Configure Scoring, Run Scoring, and Demo Segmentation Results. The 'Login' step is highlighted. The main area is titled 'Log in to data source' and contains two input fields: 'Username' with the text 'jedirv' and 'Password' with masked characters. At the bottom, there are three buttons: 'Cancel Session', 'Back', and 'Next'.

AvatoICV

Session Steps

- Select Data Source
- Login**
- Select Dataset
- Select Scoring Approach
- Select Item To Score
- Load Images
- Image Quality
- Configure Segmentation
- Run Segmentation
- Configure Orientation
- Run Orientation
- Configure Scoring
- Run Scoring
- Demo Segmentation Results

Log in to data source

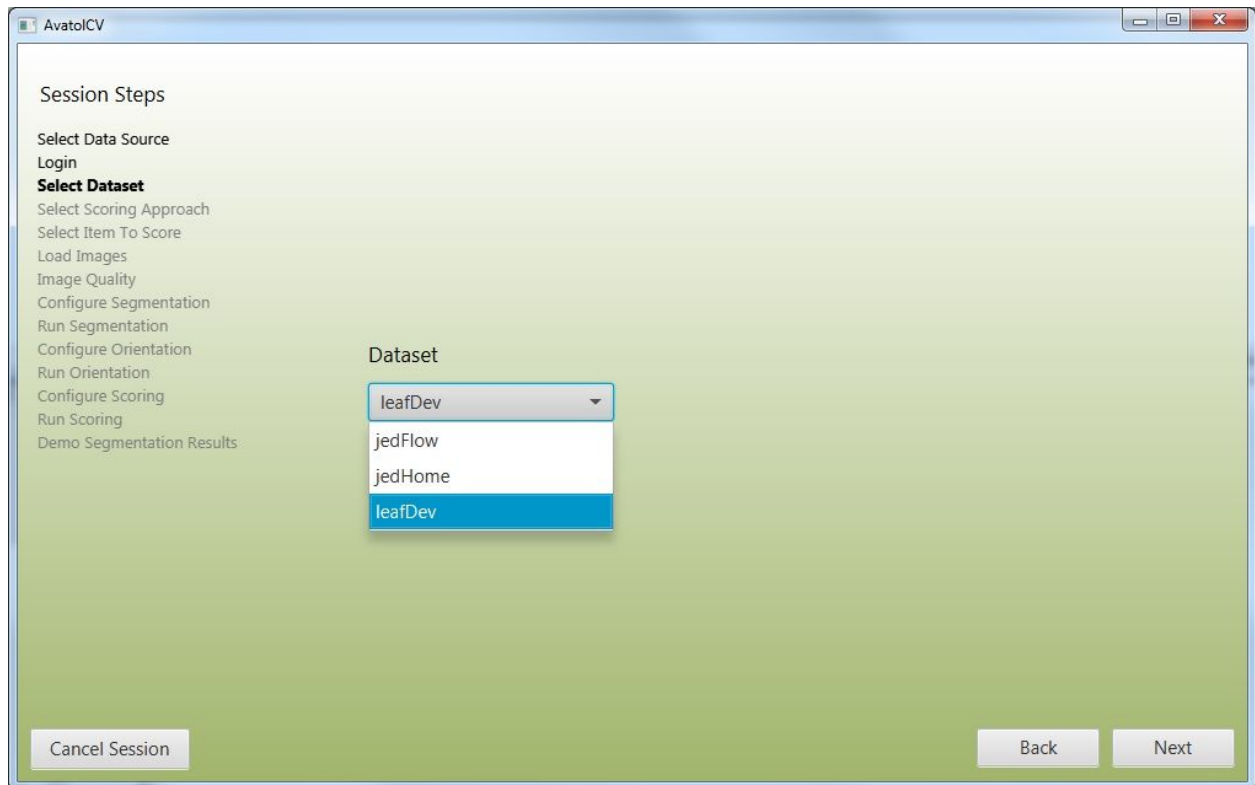
Username

Password

Cancel Session Back Next

## Choose Matrix or Dataset

For Morphobank sessions, the user is presented with the list of the matrix names they have available at their Morphobank account. For BisQue users, they are presented with the names of the Datasets they have access to in BisQue.



Once the user makes a selection, enough metadata is pulled in from the data source to enable the system to pose the next question

## Select Scoring Approach

The choice of scoring algorithm will determine whether the character choice question will allow single or multiple answers (for scoring single characters vs multiple characters at once). Thus, the scoring approach must be selected as this next step. Two general choices are provided:

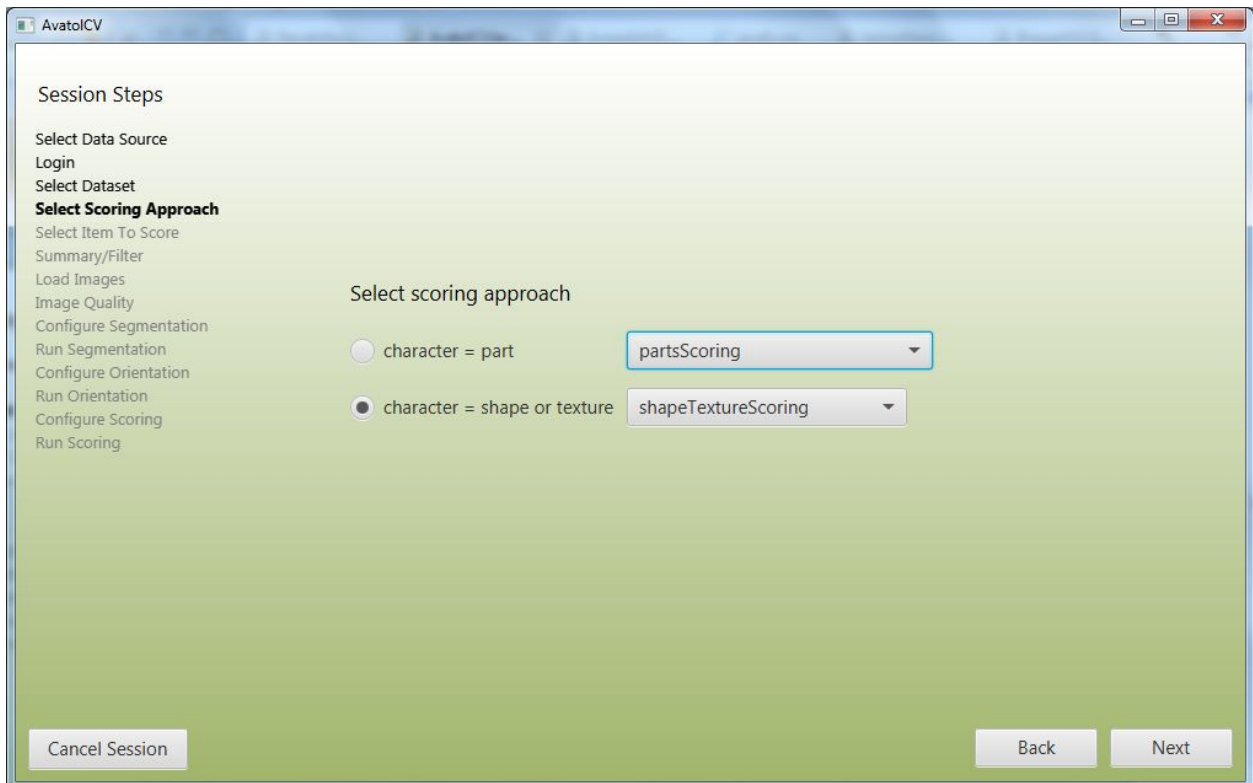
character = part

and

character = shape or texture

An example of character as part would be a tooth in a batskull. It is a visual region of the image which represents a distinct entity in the image. An example of “character as shape or texture” would be the angle of the apex of a leaf.

Depending on the choice, particular scoring algorithms are available, depending on the platform. The partsScoring algorithm is available on both Mac and Windows. The shapeAndTextureScoring algorithm is available on the Mac only. The system has been designed to accomodate additional algorithms being added later.



The screenshot shows a software window titled "AvatoICV" with a sidebar on the left listing "Session Steps". The steps are: Select Data Source, Login, Select Dataset, **Select Scoring Approach** (highlighted), Select Item To Score, Summary/Filter, Load Images, Image Quality, Configure Segmentation, Run Segmentation, Configure Orientation, Run Orientation, Configure Scoring, and Run Scoring. The main area of the window is titled "Select scoring approach" and contains two radio button options. The first option is "character = part" with an unselected radio button and a dropdown menu showing "partsScoring". The second option is "character = shape or texture" with a selected radio button and a dropdown menu showing "shapeTextureScoring". At the bottom of the window, there are three buttons: "Cancel Session" on the left, and "Back" and "Next" on the right.

AvatoICV

Session Steps

- Select Data Source
- Login
- Select Dataset
- Select Scoring Approach**
- Select Item To Score
- Summary/Filter
- Load Images
- Image Quality
- Configure Segmentation
- Run Segmentation
- Configure Orientation
- Run Orientation
- Configure Scoring
- Run Scoring

Select scoring approach

☐ character = part partsScoring

☒ character = shape or texture shapeTextureScoring

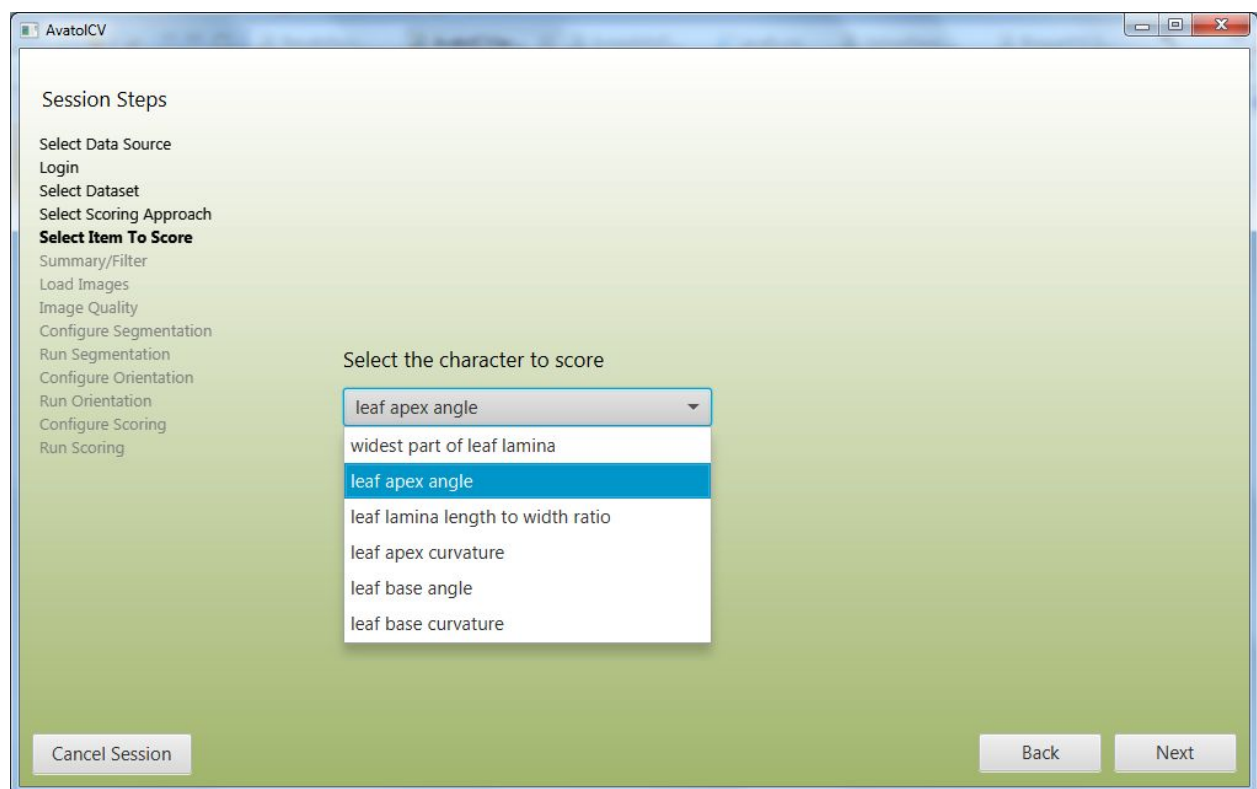
Cancel Session Back Next

## Select Item To Score

This screen allows the user to select one or more characters to be scored.

NOTE - The partsScoring algorithm performs best when the all the presence/absence parts are scored at the same time – they help disambiguate each other. So, when multiple characters are selected, the scoring of those, in a single pass, generates multiple outputs, one for each character. For example, if the characters called X, Y, and Z were selected, then there would be three results generated for that single run: 20160516\_1\_X, 20160516\_1\_Y, and 20160516\_1\_Z. So reviewing this single run would require visiting three result pages. Results reviewing is organized by focusing on one character at a time. (See AvatolCV Results Review for more details on reviewing results)

If the algorithm doesn't support multiple character scoring, then the character choices appear in a dropdown list.



Once the character(s) is selected, the remaining metadata (i.e. for those characters) is pulled in.

## Filtering Data

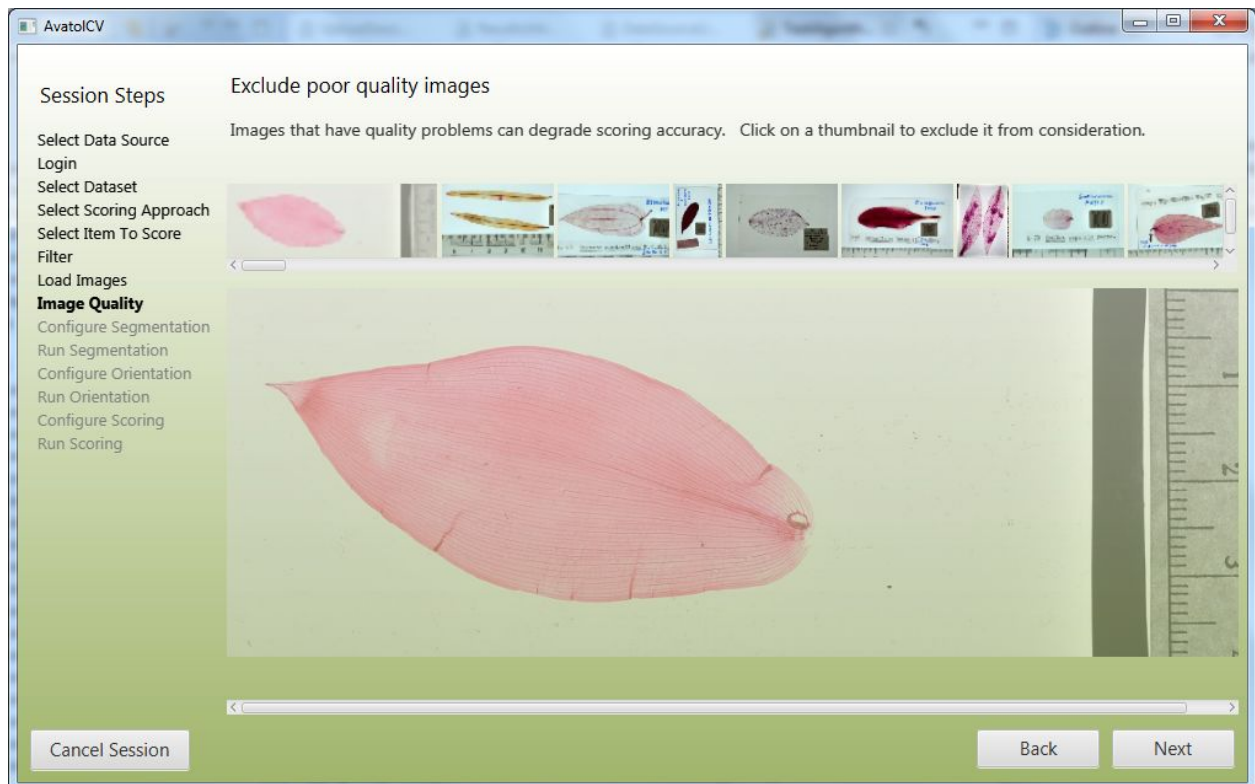
For Morphobank matrices, each cell may contain multiple images for the taxon+character combination. These images might be taken from different viewpoints (ventral vs dorsal). Since (for some algorithms) AvatolCV requires images to be from a consistent aspect, the filtering screen is provided as a way to let the user exclude images if they are not of the desired view. However, this screen is generic to any dataset's situation. If there was a BisQue project, for example, where the images that had a certain attribute/value pair that should not to be included in the later processing, that attribute/value pair could be selected here to screen those images out.

<screen shot>

Once filtering is complete, the system can now pull in images. Only the images that are relevant to the selected character(s) and which have not been filtered out, are pulled in. Also, if the images had been pulled in by a prior session, they are not pulled in again.

## Exclude Poor Quality Images Screen

All the images in play for the run are shown in thumbnail form. Hovering over one of the images shows a larger version of the image. If the image has quality issues as described shortly, it can be excluded so that it does not adversely affect the quality of the later processing. Clicking on a thumbnail will grey it out (and the larger version of it). This signifies that this image will be excluded. Clicking a greyed-out thumbnail will delete the exclusion marker and the image will once again be back in play.

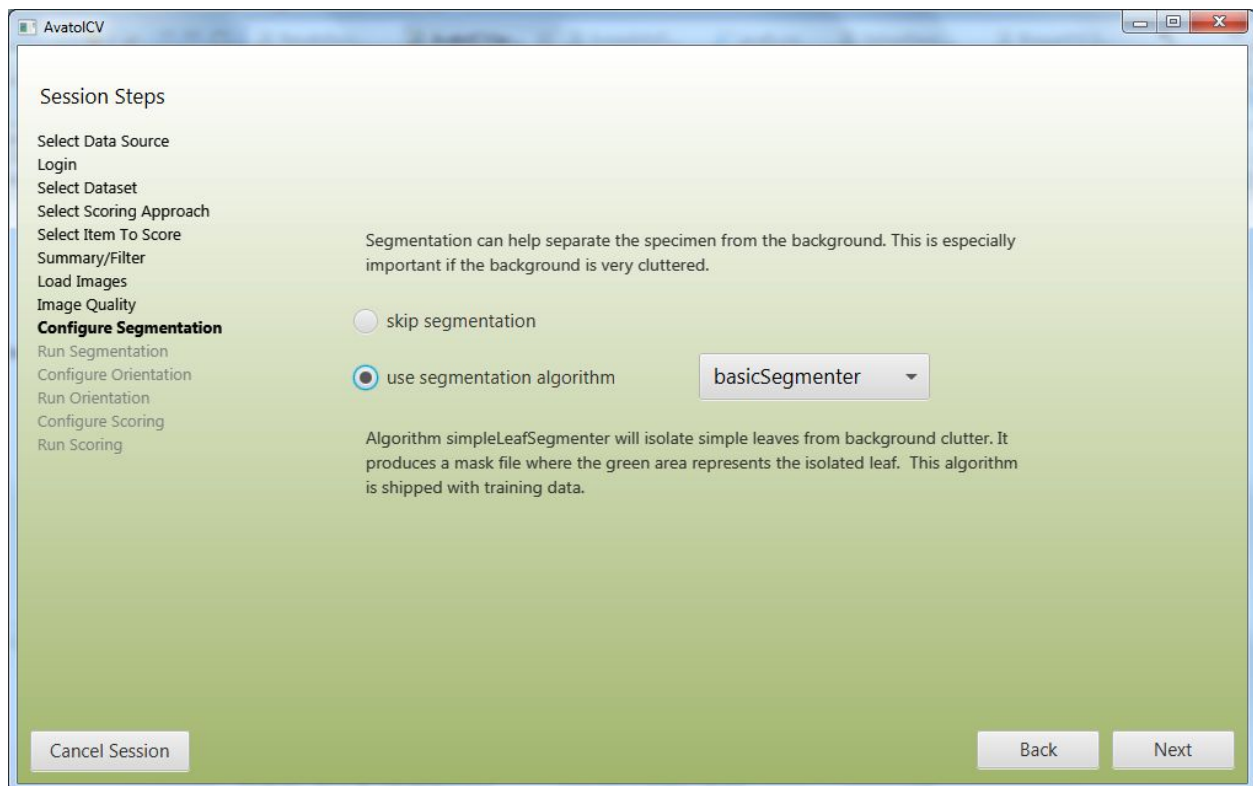


## Segmentation

Segmentation can help separate the specimen from the background. This is important if the background is very cluttered. It may or may not be needed depending on the images.

The screen allows the user to skip segmentation or select a segmentation algorithm. On Windows, the only segmentation algorithm that has been implemented is highClutterSegmenter. However, this algorithm generates output formats that aren't usable by either of the scoring algorithms in play as of this writing (May 17, 2016), so it's presence is just for demonstration purposes. highClutterSegmenter was developed to try to segment nematocysts in images where multiple specimens were overlapping.

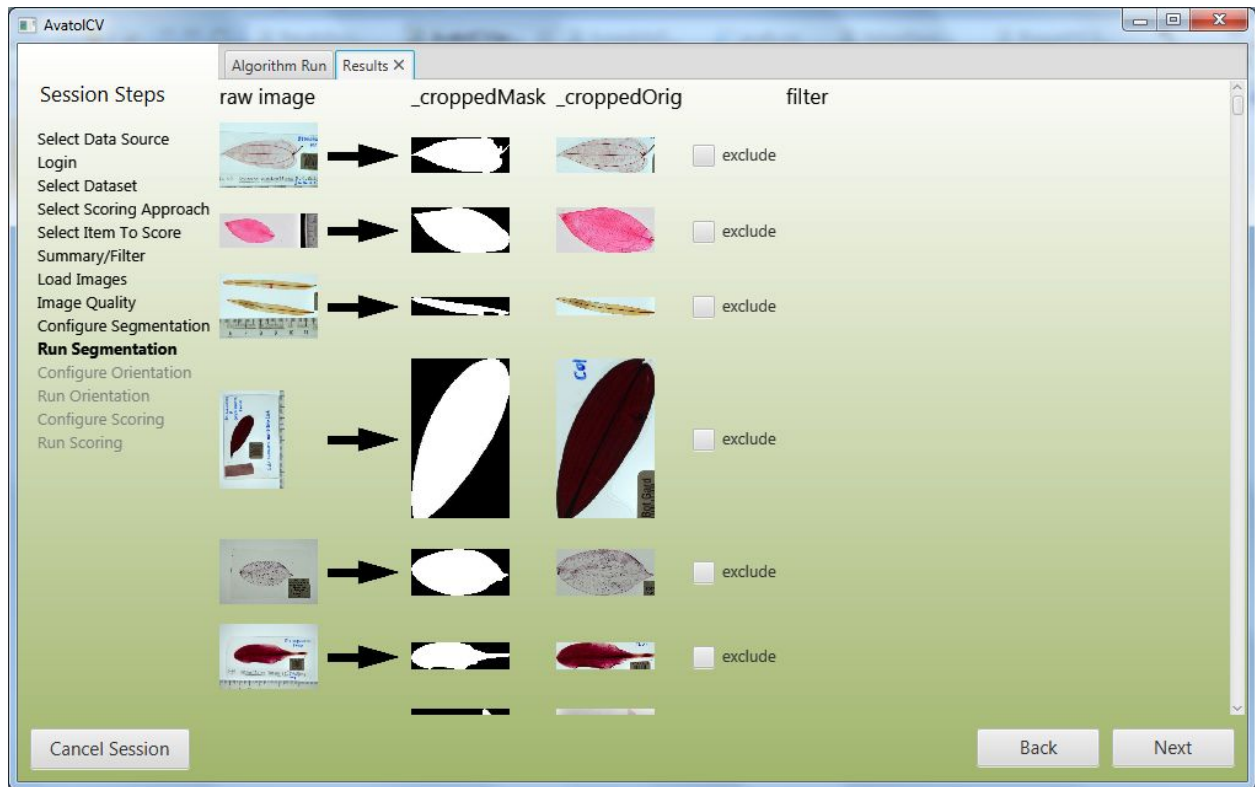
On the Mac, basicSegmenter, part of the leaf pipeline, is supported. It is trained on a leaf dataset. If other types of images are in play, the algorithm would perform better if training was on more relevant images. For instructions on how to retrain basicSegmenter, see the section called "User Supplied Training Examples"



On the Mac, when segmentation is running, the output is directed to AvatoICV Run Segmentation window. When the algorithm has finished, the AvatoICV automatically switches to the segmentation results tab on the same screen. This view shows the input and output images and provides a checkbox next to each for excluding that image from further processing.



The reason this would be done would be if the segmentation results were really bad, then there's no point in having later processing working with that segmentation output.



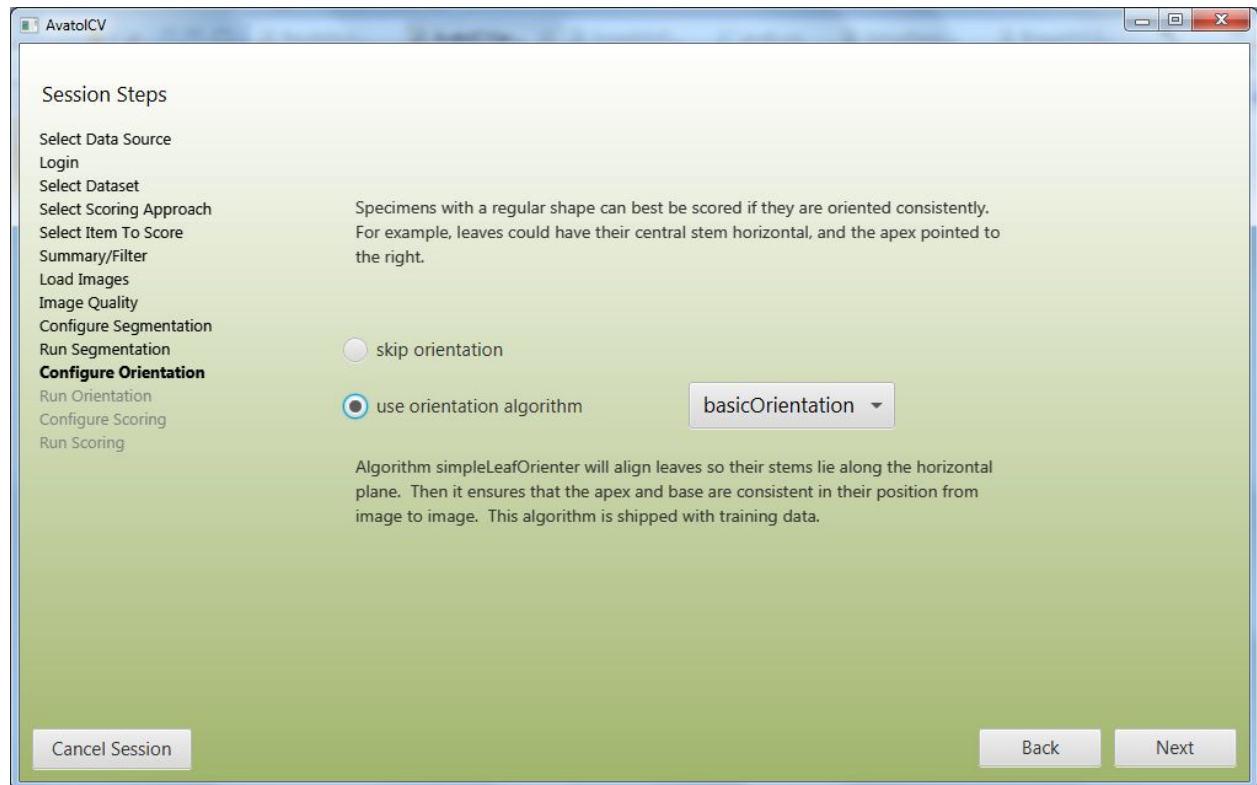
On Windows, then segmentation is running, the output is not directed to the AvatoICV Run Segmentation window, due to how Matlab (which is used by the algorithms) handles input/output on windows. To allow some visibility into how the algorithm is running, the system instead routes stdout (the console output) into a log file located at :

avatoicv\sessions\<matrix\_or\_dataset\_name>\<run\_id>\logs\<algorithmType>

Where run\_id is of the form date\_number\_characterName (20160415\_01\_leaf apex angle) and algorithmType is either segmentation,orientation or scoring.

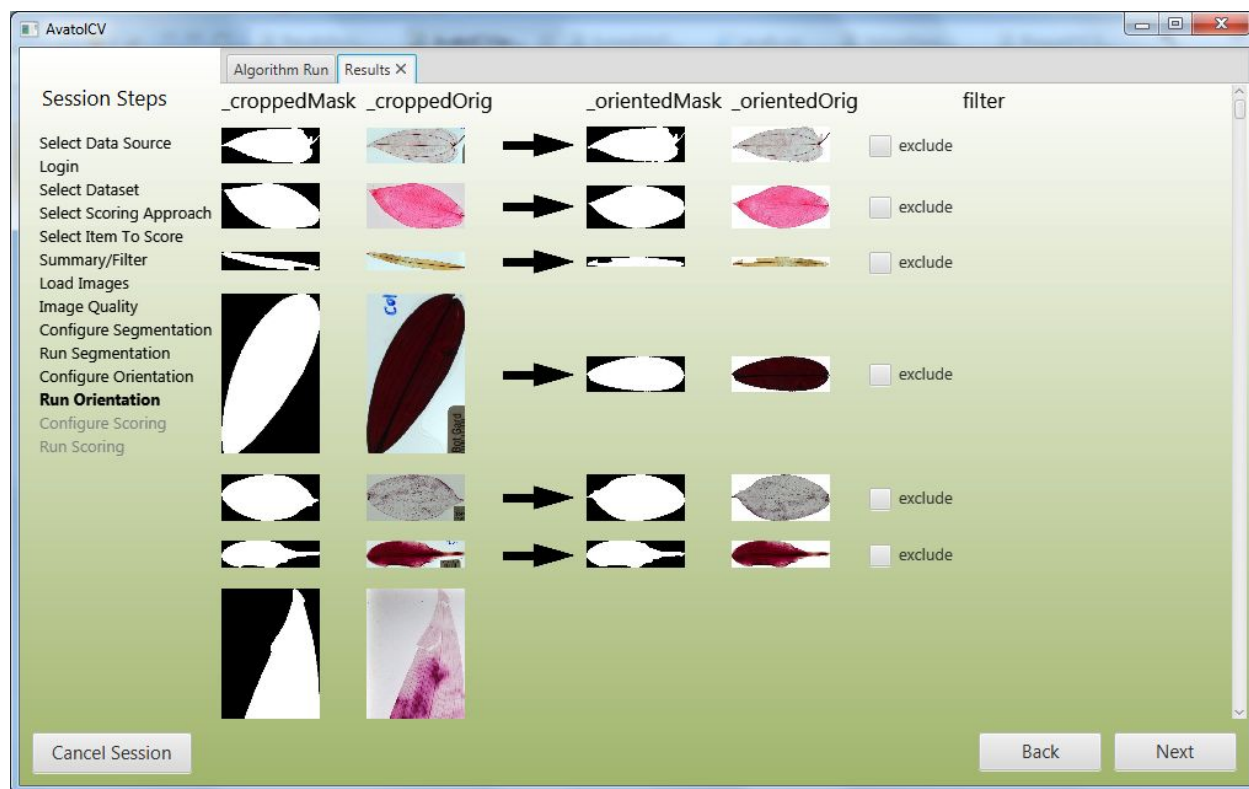
# Orientation

Orientation is to ensure that specimens are oriented in a consistent manner, for algorithms where that is important. The one orientation algorithm currently implemented is simpleOrienter (Mac only) which is trained using leaf images.



See the Segmentation section for a description of why algorithm output is not rendered into the AvatoICV window on Windows.

Orientation Results are shown in the same fashion as segmentation results - input images on the left and output images to the right of the arrow, with a checkbox to exclude those with poor results.



# Scoring Configuration

The scoring configuration screen controls how data is divided between training and scoring. There are different ways in which the data can be divided, depending on the nature of the data.

## Partially or Fully Scored?

First, the dataset might be partially scored, or fully scored already. If the dataset is completely scored (there is a character state specified for the character(s) in question for all images in play) then this positions AvatoICV into “algorithm evaluation” mode. If the dataset is not completely scored, then this positions AvatoICV into “true scoring” mode.

In “algorithm evaluation” mode, AvatoICV will divide the data into training and scoring sets. It will pretend that the images in the scoring set are not yet scored and will score them after training on the training set. The user may adjust which images are selected for training vs scoring. Then, at the results screen, the score values can be compared to the user’s prior scores.

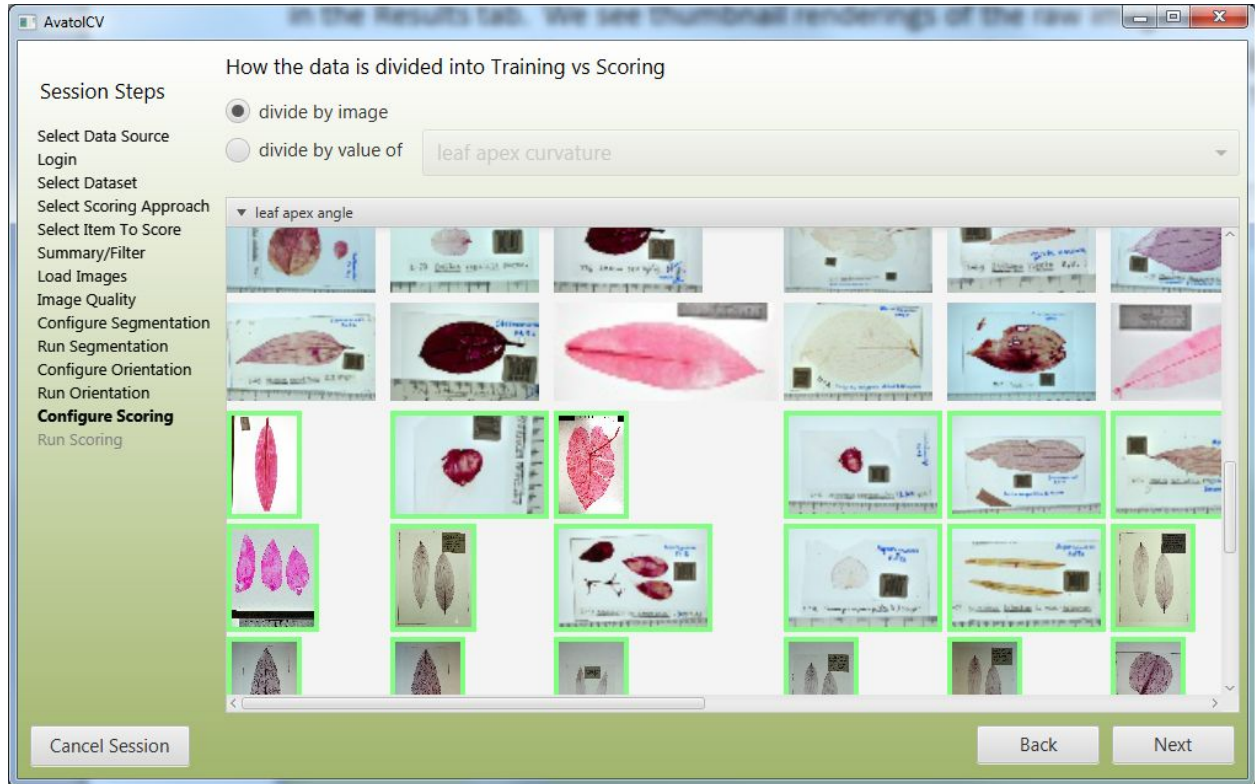
In “true scoring” mode, the training and scoring sets are predetermined by some of them having been scored and some of them not. The user is not given the opportunity to adjust which items are to be scored and trained as it wouldn’t make sense to do so in this situation.

## Training Set vs Scoring Set Nuances

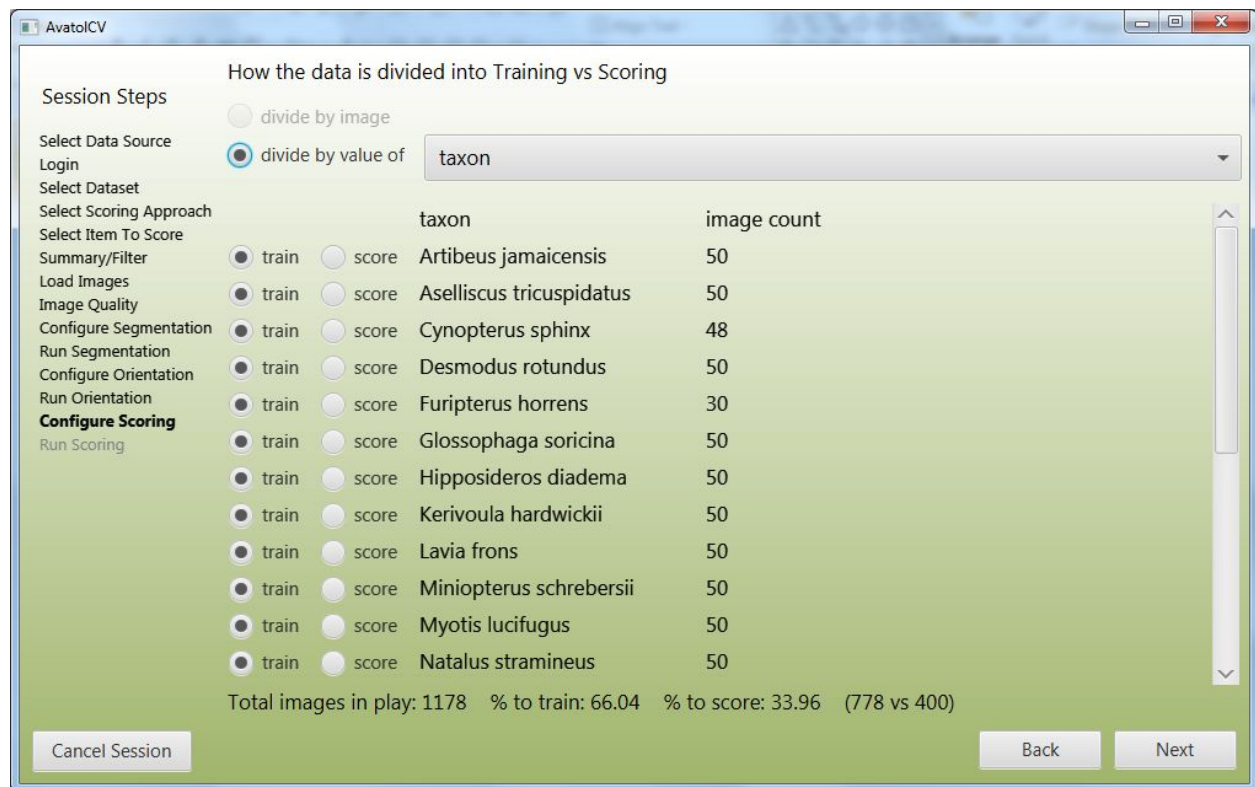
Another issue relevant to scoring configuration is whether there is a “trainVsTestKey” in play. In Morphobank, matrices have taxa on one axis, and character on the other. AvatoICV will want to train on some taxa and score on others. This makes the notion of “taxon” the trainVsTestKey for all Morphobank projects, in the the value of the taxon will determine whether an image is in the scoring set or training set.

So, the screen gives the choice of “divide by image” or “divide by value of”.

“Divide by image” is used when there is no trainVsTestKey in play, which will likely be the case for Bisque runs. The screen shot below shows this mode. The thumbnails with the green outlines are the ones that are in the scoring set. The ones lacking outlines are in the training set.



For Morphobank runs, “divide by image” is always disabled because taxon will always be the trainVsTestKey. Below is a screen shot where “divide by value of” is enforced because it is a Morphobank run, which will train on certain taxa and score on others.



The notion of trainVsTestKey is also supported for BisQue runs, but might not ever be relevant to the dataset in question. Where it would be relevant (for a BisQue run) would be if there was an attribute associated with images in the dataset where the user wanted to train on all the images where that attribute had a certain range of values, and score the ones where that attribute had a different range of values.

In summary, for a BisQue run, “divide by image” will always be enabled. “Divide by value of” will be enabled if there are any image properties other than the character(s) in play. (The automatically generated BisQue attributes of name and timestamp are ignored) If the user decides that one of these other attributes should be a trainVsTestKey, then the user can select the “sort by value” option, and AvatoICV will divide the dataset by finding

**IMPORTANT NOTES FOR MORPHOBANK USERS:** please refer to the section called Training Data Consistency.

## Scoring

The scoring screen shows progress of the scoring algorithm, just as Segmentation and Orientation do. When completed, AvatoICV automatically moves the user to the Results Review screen.(see next section) See the Segmentation section for a description of why algorithm output is not rendered into the AvatoICV window on Windows.

# AvatoICV Results Review Screen

This screen appears when scoring is complete or when the user chooses to review results from a prior scoring session at the Start Screen. There are two flavors of Results Review Screen. They are distinguished by what information is presented.

“True Scoring” mode runs are runs done with datasets that have only been partially scored by the user. This means that when AvatoICV has completed its automatic scoring, there are no “true labels” to compare these results to. In “Algorithm Evaluation” mode, the dataset has been completely scored prior and AvatoICV has divided the data up into a training and scoring set. Then, when scoring is complete, we do have “true labels” to compare the scores to.

## True Scoring Results

Below is a screenshot of the Results Screen for a True Scoring mode run.

The screenshot shows the AvatoICV Results Review Screen. The left sidebar displays the 'Run Overview' for the selected run '20160415\_01\_leaf apex angle'. The main panel shows a table of results for 'leaf apex angle - SCORED images'.

**Run Overview:**

- Run ID: 20160415\_01
- Dataset: leafDev
- Data Source: bisque
- Algorithm: debugCheatLeafScore
- Scoring Target: leaf apex angle
- Values: acute, straight, obtuse, reflex

**Results Table:**

| image | score  | saved? | confidence | name                     |
|-------|--------|--------|------------|--------------------------|
|       | acute  | no     | 0.75       | Asparagus-falcatus-var   |
|       | acute  | no     | 0.95       | clearings-019            |
|       | obtuse | no     | 0.62       | Chamaelirium-luteum-0146 |
|       | acute  | no     | 0.92       | Alstoemeria-mcl73-mp     |
|       | obtuse | no     | 0.63       | Chamaelirium-luteum-0147 |
|       | acute  | no     | 0.93       | clearings-021            |
|       | obtuse | no     | 0.63       | Dioscorea-communis-0040  |

Adjust confidence threshold to control which scores are kept: 0 25 50 75 100

Buttons: Upload Scores, Undo Upload 1, Done

The left side of the screen shows which run is selected, and this can be changed using that control. Under the run selector, we see the Run ID, the matrix or dataset name, the datasource used, and the scoring algorithm that was used.



Below that we see the character chosen for scoring specified as the scoring target. Below that are the range of values possible for that character. (the character states in Morphobank parlance).

Remember that if the partsScoring run involved multiple characters, each character would have its own results screen that can be chosen in the selector at the top.

On the right side of the screen we see two tabs - one for SCORED images and one for TRAINING images. The training images are there for reference with their user assigned scores. More interesting is the scored image view which shows:

Image - a thumbnail for each image scored

Score - the score generated by the scoring algorithm

Saved? - an indication of whether this score has already been uploaded

Confidence - The algorithm's confidence in that score

Name - the original name of the image, if available

## Algorithm Evaluation Results

The screen shot below is from an Algorithm Evaluation run so we can show all the columns that might be present..

The screenshot displays the AvatoCV software interface. On the left, the 'Run Overview' panel shows the following details:

- Run ID: 20160202\_01
- Dataset: AVAToL Computer Vision
- Data Source: morphobank
- Algorithm: partsScoring
- Scoring Target: M3 presence
- Values: M3 present, M3 absent

The main panel shows a table of results under the 'M3 presence - SCORED images' tab. The table has the following columns: image, truth, score, saved?, confidence, taxon, and name. The data rows are as follows:

| image | truth      | score      | saved? | confidence | taxon                 | name   |
|-------|------------|------------|--------|------------|-----------------------|--------|
|       | M3 present | M3 present | no     | 1.00       | Mormoops megalophylla | 380465 |
|       | M3 present | M3 present | no     | 1.00       | Saccopteryx bilineata | 380472 |
|       | M3 present | M3 present | no     | 1.00       | Trachops cirrhosus    | 380474 |
|       | M3 present | M3 present | no     | 1.00       | Mormoops megalophylla | 380489 |
|       | M3 present | M3 present | no     | 0.98       | Saccopteryx bilineata | 380495 |

At the bottom, there is a 'Filtering' section with a slider to 'Adjust confidence threshold to control which scores are kept' (ranging from 0 to 100) and buttons for 'Upload Scores', 'Undo Upload', and 'Done'.

Notice these additional columns:

Truth - the true label (if it was an evaluation run)

Taxon - this is the name of the trainVsTestKey that is relevant for Morphobank runs

Note also that the Morphobank batskull images have yellow crosshairs. In the training tab, these crosshairs indicate where the user has specified that the part exists. In the scores tab, they indicate where AvatoICV thinks the part is.

In Morphobank, image names are not available in the web service api used to access the information so the internal Morphobank image ID is used to stand in for the image name.

## Sorting Columns

Clicking on any column header (image, truth, score, etc) will sort the table on that column.

Clicking on the same column header will then reverse the sort. Clicking on the image column has the same effect as clicking on the name column.

## Zooming In and Out of an Image

To see an image in more detail, click the thumbnail and a large version of the image will appear below it. Click again and the large image will be removed.

## Confidence Threshold Control

Below the scrolling image list, there is a slider for adjusting the confidence threshold. Any images with scores lower than the confidence threshold will be greyed out and ignored when scores are uploaded (saved to Morphobank or BisQue).

## Uploading Scores

Once the user has set the threshold in such a way that they are happy with the scores above it, they can click the Upload Scores button at the bottom of the screen. Once pressed, the user will be presented with an authentication screen, unless they have already authenticated with Morphobank or BisQue already during that session. After that, the progress bar along the bottom of the screen will show how far in the upload process the system is. If the user decides

they made a mistake and wants to undo the upload, they can click the Undo Upload button and scores at the data source will be restored to their prior value.

As scores are uploaded, the associated rows are highlighted and the value of the “saved?” column is changed from “no” to “yes”.

Clicking Done will take the user back to the Start Screen.

# Caveats, Limitations, and Assumptions

## Dataset or Matrix Name Uniqueness

AvatolCV assumes that for a given user account in either BisQue or Morphobank, that the matrix or dataset names will be unique. Since AvatolCV places all the information about for a run under a directory named after the project or matrix, two same-named projects pulled into AvatolCV would have their files co-mingled, with corrupting results. Further, if a user has both a BisQue account and a Morphobank account, if they reuse the project name in both places, and pull those both into AvatolCV, those will also co-mingle and corrupt each other. An enhancement request has been filed to include the dataset or matrixID in the pathnames to avoid this problem.

## Training Data Sufficiency

An issue in general for CV algorithms is having sufficient training data. For Morphobank matrices, in order for partsScoring to function well, there must either be a large enough number of taxa involved or multiple images per taxon if there are fewer taxa. Development was done using a Morphobank matrix with around ten images per taxon and 24 taxa.

BisQue has no formal notion of taxon as part of its data model, though in theory it could be a property of the image and if so could be specified as the trainVsTestKey at the “divide by value of” control in the UI. For development we used a Bisque dataset with around 200 images.

## Training Data Consistency

For Morphobank users, the following situation has not been rigorously accounted for in AvatolCV and will likely encounter errors: some images for a taxon (of the desired view) scored and some not. This is true also if multiple characters are in play for the run. Let’s say you’ve selected two presence/absence characters called X and Y for a partsScoring run. If all of character X images are scored for taxonA, but not all of character Y are scored for taxonA, then you are in the likely-to-have-errors situation.

The same issue applies to BisQue users for the case where a trainVsTestKey is specified at the “divide by value of” control at the Scoring Configuration Screen, whether that trainVsTestKey is an attribute called “taxon” or something else. Let’s say the images in the dataset have attributes charX and charY that represent presence/absence characters and you want to run the partsScoring algorithm on them. For partsScoring, you need a trainVsTestKey, so let’s say the trainVsTestKey you choose is an attribute called “species”. That means that for every image

where the value of attribute species is SpeciesA, you'll want all the images to have charX and charY either both already scored (for training) or both not yet scored (for scoring). Because BisQue does not impose a conception of a matrix around its images, it is much more difficult in BisQue to ensure the healthy training data consistency condition.

It is possible in Morphobank for an image for a specimen to be reused for different characters. There is a restriction in AvatoICV that the same image not be used for both training and testing. This is another reason why all characters for a given taxon should either be scored or not scored.

Since the same image won't appear twice in a BisQue dataset (at least on purpose), there is no way that the same image can wind up being used for both training and testing.

## **Image Orientation Consistency**

For partsScoring, specimens must be consistently oriented, for example, images of skulls should all be pointed in the same direction. As of this writing (May 17, 2016), this orientation must be done prior to images being pulled into AvatoICV, since the basicOrienter algorithm has been trained on leaf or leaf-like images so will not perform well on other types of images.

# How To Run AvatolCV Using Different Versions of Matlab

Each of the following scripts refers to R2015b so you will need to adjust them before you try to run the algorithms:

For running on the MAC:

avatar\_cv/modules/segmentation/yaoSeg/segmentationRunner.sh

(the line that says : /Applications/MATLAB\_R2015b.app/bin/matlab -nodisplay -r "\$matlab\_func"  
quit)

avatar\_cv/modules/orientation/yaoOrient/orientationRunner.sh

(the line that says : /Applications/MATLAB\_R2015b.app/bin/matlab -nodisplay -r "\$matlab\_func"  
quit)

avatar\_cv/modules/scoring/leafScore/leafScore.sh

(the line that says : /Applications/MATLAB\_R2015b.app/bin/matlab -nodisplay -r "\$matlab\_func"  
quit)

avatar\_cv/modules/scoring/batskullDPM/batSkullScore.py

(the line that says : MAC\_MATLAB\_PATH = "/Applications/MATLAB\_R2015b.app/bin/matlab")

For running on WINDOWS:

avatar\_cv/modules/scoring/batskullDPM/batSkullScore.py

(the line that says : WIN\_MATLAB\_PATH = "C:\\Program  
Files\\MATLAB\\R2015b\\bin\\matlab.exe")

# AvatolCV Internal Data Format

To be written...

# Adding New Algorithms into AvatolCV

To be written...

# User Supplied Training Examples

To be written...

# What the Installer Does In Detail

AvatolCV is broken up into the following bundles:

```
docs
modules_3rdParty
modules_osu
java
```

usage: `python updateAvatolCV.py <installRoot>`

Here is the process `updateAvatolCV.py` uses:

1. ensures specified installation root dir exists
2. creates an `avatol_cv` dir under that installation root dir
3. determines a platform code - what system we are running on : win | mac | unsupported
4. generates the name of the downloadManifest file that it needs to download (from <http://web.engr.oregonstate.edu/~irvineje/AvatolCV/>) using:

```
downloadManifest_<platform_code>.txt
```

5. generates a temporary name that it will save that file under

```
downloadManifest_<platform_code>_new.txt
```

6. downloads that file and saves as that name

```
new_manifest_pathname = downloadFile(avatolcv_root, new_manifest_filename,  
manifest_truename)
```

7. if this is not the first installation maneuver, the prior maneuver would have saved the file as...

```
downloadManifest_<platform_code>_old.txt
```

...so we look to see if that file exists

8. If that old file exists we dump it to the console, then dump the new one to the console for sanity comparison by user

9. We determine which bundle names need downloading by comparing old and new.

- If no old file exists, everything in new will be pulled.
- If any bundle name represented in new is not represented in old, it is pulled
- otherwise, we compare the datestamp (i.e. the version) associated with new to see if it is different than old. If so, we download.

Note that we don't check for a more recent version, just difference with prior file. This will simplify regressing back to prior version if need be.

If no bundles are needed, we declare we are up to date and exit, otherwise we continue

10. foreach bundle we need to download, we first uninstall the prior-installed version of that bundle using the

```
allFiles_<bundle_name>.txt
```

file that came with each bundle as a guide for which files to delete.

11. we delete the old manifest file and rename the new one to

```
downloadManifest_<platform_code>_old.txt
```

...so it can be checked at the later download

12. now we install each bundle by looking up the bundle filename from the downloadManifest and pulling from the distro site.

This yields a .tgz bundle that we then unwrap underneath the avatolcv dir



