Skyline External Tools Documentation

External Tools are customizable functions that can be accessed through the Tools drop down menu of the main Skyline window. The external tools framework allows researchers to share programs they have written to process the raw mass spectrometer data from a Skyline document in a user friendly manner. One example of an external tool would be an R script that runs a statistical analysis on Skyline data and plots the results to PDF files. Thus the goal of the external tools framework is to provide a convenient way to connect the data processing features written by researchers around the world with the data found in a Skyline document.

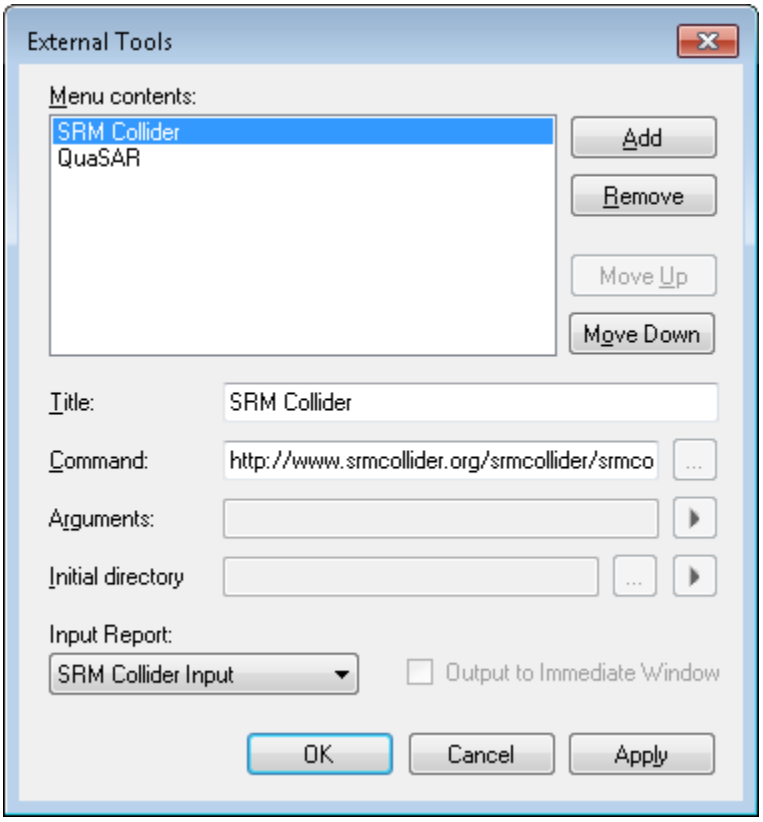

Figure 1 - The Configure Tools Dialog where tools can be edited.
Bring up this dialog by clicking Tools>External Tools...

Currently we envision tools being written in one of the following forms: R, Python or Perl scripts, executable, batch or command files, or a website that accepts Skyline data via http.

Custom Reports

An external tool accesses Skyline data through the use of Skyline custom reports. A custom report is used to describe the expected contents of a file containing Skyline data that will be passed to the tool either as a temporary file saved locally on the file system or piped directly into the tool itself. If you are

not familiar with the custom report framework check out the Skyline Custom Reports and Results Grid documentation on how to create, use, and export Skyline data via custom reports.

When a tool is run, the custom report sources its data from the open Skyline document. That report is then passed to the tool one of three ways:

1. If the tool is a website, a web browser will be launched with the report passed as an HTML encoded string through HTTP POST to the website specified by the tool.
2. If the tool contains a $(InputReportTempPath) macro in its arguments field, the report is saved to the local file system and the path to that file is given to the tool.
3. Otherwise the report is piped directly to the tool via standard input.

The input report for a tool can be accessed in the Input Report drop down menu located in the bottom left corner of the External Tools window.

Macros

Macros are context sensitive values that will be replaced when the tool is run.

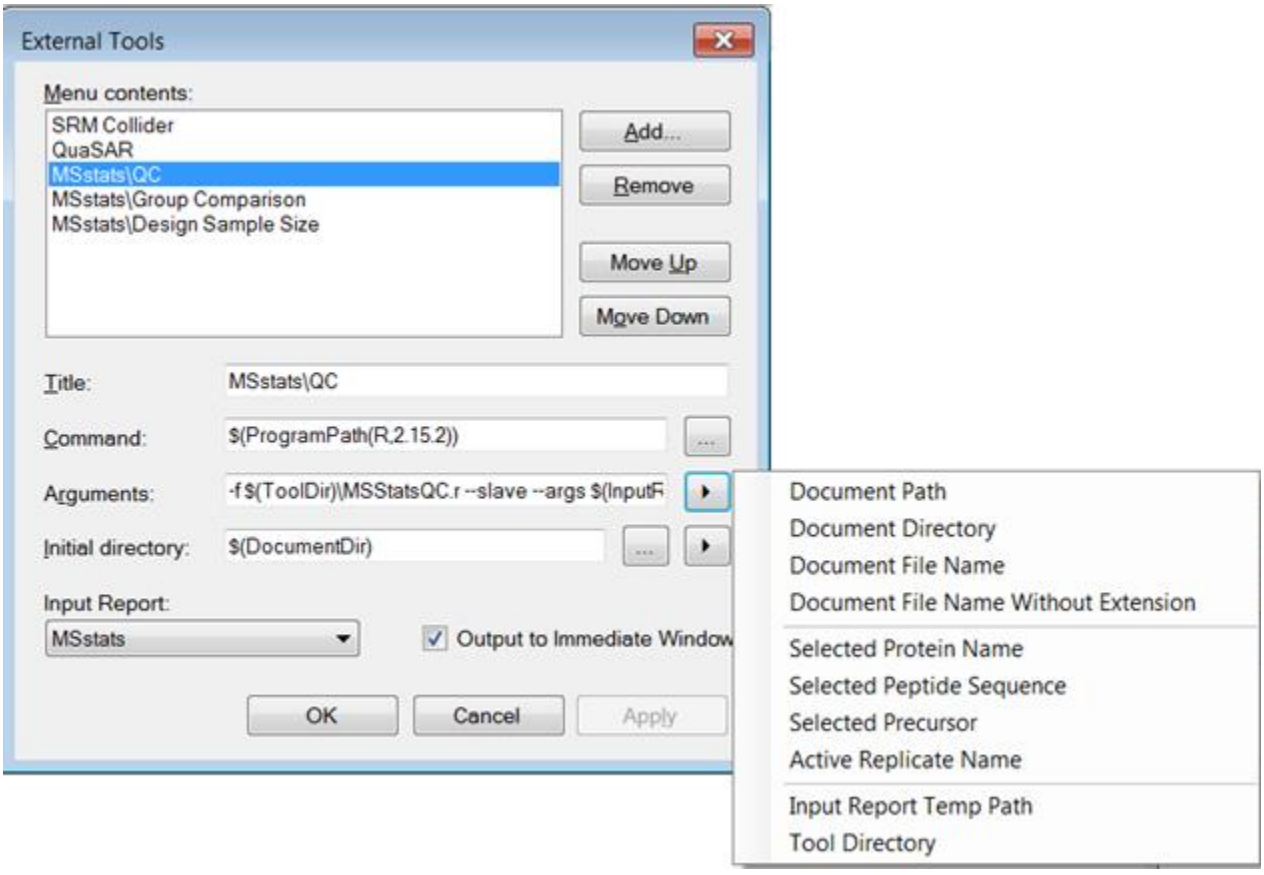

Figure 2 - Clicking the arrow button displays all the macros available. Selecting a macro off of the menu will add the macro text to the end of the relevant string. All macros are of the form $(<something>) to be clear that they are a macro. Hover over the menu items on the right to preview their value in the current context.

1. Document Path - $(DocumentPath)

When a tool is run, this value will be replaced with the path to the open Skyline document on the local file system.

2. Document Directory - $(DocumentDir)

When a tool is run, this value will be replaced with the path to the directory containing the open Skyline document on the local file system.

3. Document File Name - $(DocumentFileName)

When a tool is run, this value will be replaced with the name of the open Skyline document, including the .sky extension.

4. Document File Name without Extension - $(DocumentBaseName)

When a tool is run, this value will be replaced with the name of the open Skyline document, without the .sky extension.

5. Selected Protein Name, Peptide Sequence, Precursor or Active Replicate Name - $(SelProtein), $(SelPeptide), $(SelPrecursor), $(ReplicateName)

When a tool is run, these values will be replaced based on what the user has selected in the sequence tree of the open Skyline document.

6. Input Report Temp Path - $(InputReportTempPath)

When an external tool uses this macro, it indicates that the external tool requires input from a custom report. This macro will be replaced with the path to a temporary file containing the custom report that was generated from the open Skyline document using the specified input report format. If a report is selected and this macro is not used then the report will be piped to the tool directly via standard input.

7. Tool Directory - $(ToolDir)

This macro is only available if the tool was installed from a zip file (see Installable Tools below). When a package of tools is installed a file directory is created to store the relevant files for all the tools in that package. When the tool is run, the $(ToolDir) macro will be replaced with the path to that directory. To view the tool directory path, hover over the Tool Directory menu item until the information text is shown.

8. Collected Arguments - $(CollectedArgs)

This macro is also only available if the tool was installed from a zip file. For more information on argument collectors see the ArgsCollector section of this documentation. An ArgsCollector is a Windows Form graphical user interface packaged with a tool that will prompt the user for input each time the tool is run. The ArgsCollector will produce a string of arguments for the tool that will replace the $(CollectedArgs) macro if it is present, otherwise the collected arguments are placed after the last provided argument in the Arguments field.

9. Program Path - $(ProgramPath(<ProgramTitle>,<ProgramVersion>)) or
$(ProgramPath(<ProgramTitle>)

This macro is used when the tool requires an external application in order to perform analysis on Skyline data. For example, a number of external tools written for Skyline require the use of the R statistical programming language. To have a tool run an R script you need the path to R executable file on the current machine as the command.

Currently, if tool authors specify the <ProgramTitle> as R or Python, when a user installs the tool, Skyline will search for the correct <ProgramVersion> of R or Python on the user's machine. If it is not installed, Skyline will guide the user through the installation of that specified version of R or Python. If this macro is used when specifying an external tool that does not use R or Python, then Skyline will prompt the user to provide the path to the correct program of the correct version.

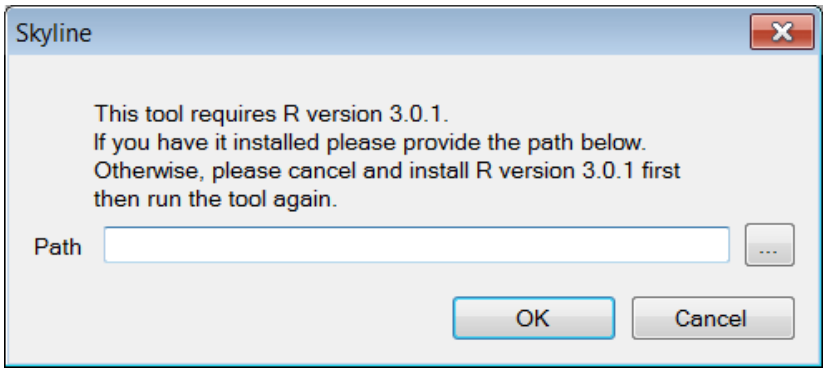

Figure 3 - This is an example of the dialog that will come up if a tool with the command $(ProgramPath(R,3.0.1)) is run and Skyline does not already have a saved value for the path to R version 3.0.1.

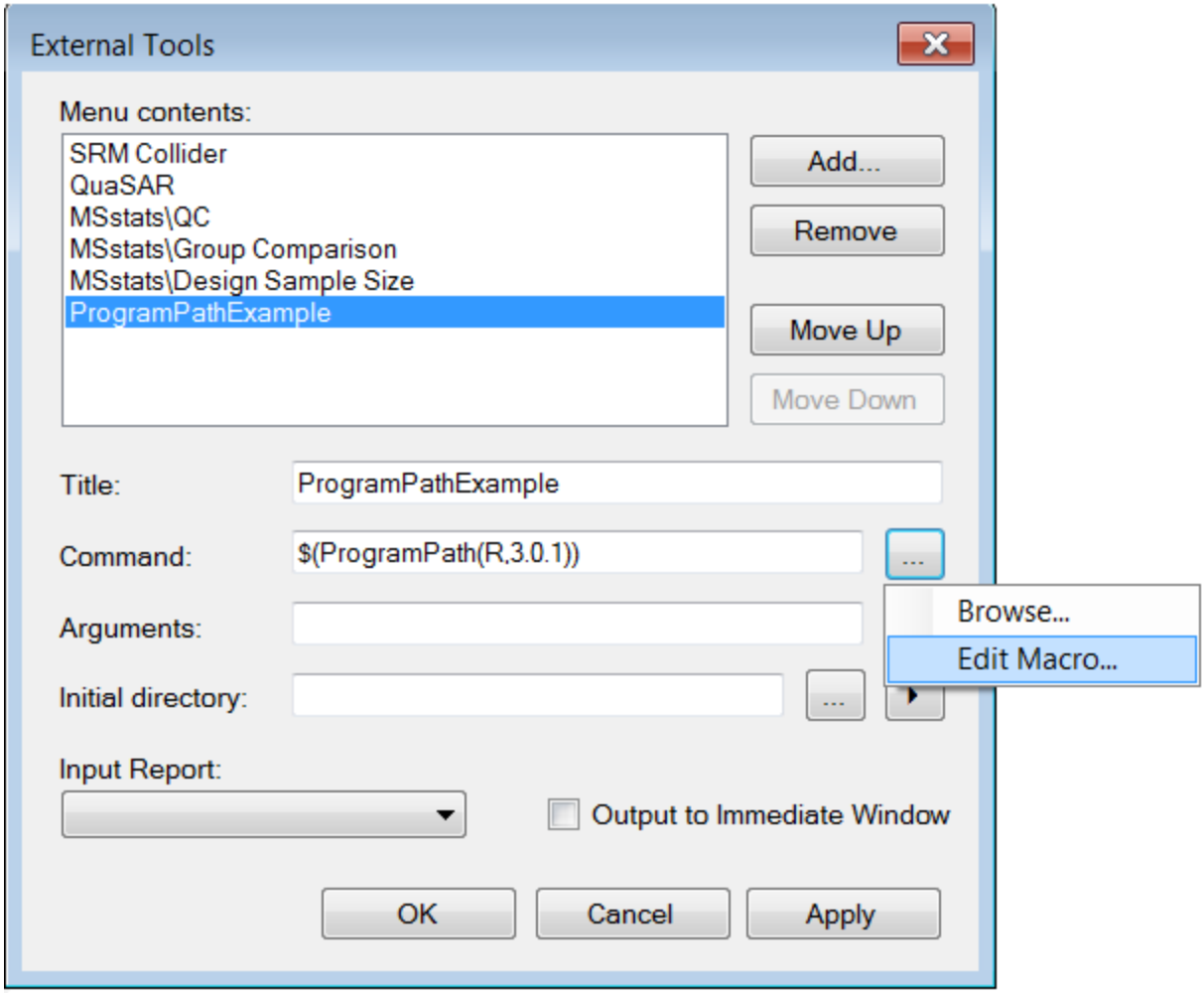

Figure 4 - An alternative way to access to the dialog pictured in Figure 3 is shown above. If Skyline does not have a saved value for the program version the user will be prompted for it when they run the tool. The Edit Macro menu item highlighted above provides a way to edit the saved value for the path later.

Output to Immediate Window

This option should be checked if your tool writes text to standard output or standard error and you would like that output displayed in the immediate window of the Skyline program. Otherwise, a separate window will be displayed with the output of the tool.

Title

Tool titles should be unique. Also tool titles can be used to group multiple tools together in a cascading menu (Figure 5). Cascading tool titles can be specified by including the '\' character.

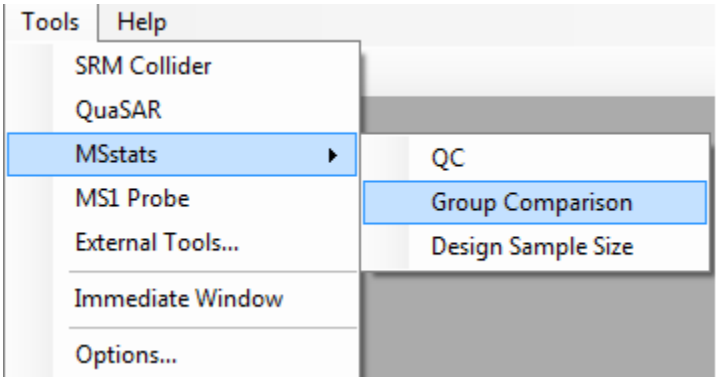

Figure 5 - Menu items can be made to cascade by adding a '\' to the title. The above tool menu has three tools titled MSstats\QC, MSstats\Group Comparison, and MSstats\Design Sample Size.

Command

To have a tool launch a website ensure the URL starts with 'http://' or 'https://'. The Input Report will be posted to the provided URL. Otherwise the command should be the path to an executable file on the local machine or a Program Path macro.

Argument Collectors

Argument collectors can only be used with tools installed from zip files. Argument collectors are designed for when a tool requires the user to specify command line arguments when a tool is run. These are particularly useful for tools such as R or Python scripts that do not have a way of displaying a GUI on their own. Argument collectors must be formatted as implementations of a .NET Windows Form and packaged in dynamically linked library (.dll) files. The dll file must contain a class that implements a method called 'CollectArgs' with the following declaration:

public static string[] CollectArgs(string report, string[] oldArguments)

This method is invoked by Skyline to collect arguments for the tool it is associated with. The parameter 'report' is a string representation of the associate tool's input report. The parameter 'oldArguments' is the array of arguments created the last time the argument collector was run if that value is available. This can be used to populate the form with the values the user specified in the previous run. CollectArgs should return an array of arguments to be passed to the tool. These arguments will be placed where the $(CollectedArgs) macro is in the argument string or will be appended to the end of the arguments string if the macro is not present.

A dll file can contain multiple argument collectors. When you set up a description of a tool to be installed you must specify the full name of the class that implements the CollectArgs method (Namespace.Classname). Different tools can point to different classes that implement different versions of the CollectArgs method in the same dll file. Argument collectors are called every time before the tool begins execution. If the argument collector returns null, Skyline assumes the user canceled or there was some sort of error and that the argument collector has already displayed the relevant error message. When an argument collector returns null, tool execution is terminated silently. If you would like to implement an args collector that optionally generates arguments, have it return an empty string in the case that you want no arguments passed on to the tool.

Installable Tools

It is possible to automate the installation of a tool by creating an appropriately formatted zip file. The zip file describes one or more items that will appear on the tools drop down menu. Each item that will appear on the drop down menu is called a tool component. A zip file describes a tool that can be made up of one or more tool components. The MSstats tool is a concrete example of this. The tool is packaged as a single zip file but there are three tool components QC, Group Comparison, and Design Sample Size.

The zip file must contain a directory with the name tool-inf, which contains:

1. **Required:** One and only one info.properties file (detailed below)
2. **Required:** A .properties file for each tool component to be installed. (detailed below)
3. **Optional:** A Skyline Report (.skyr) file that contains the format for the custom report required by the tool. See the Skyline Custom Reports and Results Grid tutorial for information about creating a .skyr file.
4. **Optional:** A Skyline (.sky) file containing the custom annotations used in the Skyline custom report required by the tool. See Skyline Custom Reports and Results Grid tutorial section on annotations.
5. **Optional:** More files to deal with R or Python Packages detailed in the appendix.

Required info.properties

Each tool requires an info.properties file. This file provides information about the collection of tool components as a whole. This is where the tool Name, Version, Author, Description, Provider and Identifier should be provided. Package version, Identifier, and Name are required attributes; the others are options but should be provided if available. **Note:** the name of the file should be info.properties with a lowercase i and it should be in the tool-inf directory.

```
1  Name = MSstats
2  Version = 0.9
3  Author = Meena Choi, Ching-Yun Chang, Dr. Timothy Clough, Dr. Olga Vitek
4  Description = "MSstats is an R package that provides tools for protein qu
5  Provider = http://www.msstats.org/
6  Identifier = URN:LSID:www.msstats.org
```

Figure 6 – This is an example of a info.properties file. Name, Version, and Identifier are required values.

Attributes of an info.properties file

Name **Example:** Name = MSstats	Name of the Tool (Required) **Note:** If your tool has one tool component it makes sense for the Name in the info.properties and the Title in the tool component .properties to be the same.
Version **Example:** Version = 1.23.2	Tool Version (Required) **Note:** Version can be up to 4 numbers each separated by a dot.
Author **Example:** Author = Trevor Killeen	Tool Author. **Note:** Can be more than one name.
Description	Tool Description **Notes:** This is a short blurb about your tool that will be used in the future for the tool store.
Provider	Tool Provider and Maintainer. **Notes:** This should be a link to a website with more information on the tool (if available)
Identifier	Tool Identifier (Required) **Notes:** Identifier should be a Life Science Identifier (LSID) more information here. The identifier will be how we determine which tool to replace when installing an update of a tool.

Tool Component .properties file

Each tool component has its own .properties file is required to have a Title and Command attribute. Other than those two attributes all others are optional. Below there is a table of all attributes.

```
1  Title = Counter
2  Command = NumberWriter.exe
3  Arguments = 100 100
4  OutputToImmediateWindow = True
```

Figure 7 - This is an example of what a simple .properties file might look like. This can be found in the provided example files in the counter.zip file. The grey area is just line numbering done by Notepad++ and is not part of the text in the .properties file.

Figure 7 shows what an example .properties file looks like. It has one attribute on each line with the attribute name on the left of an = symbol and the value on the right.

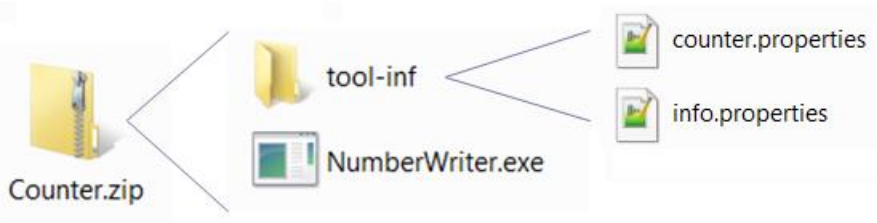

Figure 8 – This is a schema of how a simple zip tool should be laid out. Figure 6 shows the text of the counter.properties.

Note: For the Command attribute specify the path to the file from the top level of the zip file if the command is not a website or a program path macro. This is not the case if you want to have the path to a file as an argument. As we will see in a later example, in the arguments attribute you must use the Tool Directory macro in the form $(ToolDir)\\<FileName> if you have a file that you want to make it into the Tool Directory and have its path as an argument.

To create a .properties file create a simple text file (.txt) than change the extension to .properties later. Notepad++ is a free text editor that can save files in the .properties format.

In the .properties file all instances of a '\' must be replaced with the escape sequence '\\'.

List of all Tool Component Attributes and what they do.

Title **Example:**	Title of the tool. (Required)

Title = MSstats\\Group Comparison	**Note:** The use of the escape sequence '\\'
Command **Examples:** Command = NumberWriter.exe Command = $(ProgramPath(R,2.15.2)) Command = https://www.google.com	Tool Command. (Required) **Note:** you do not need to use the $(ToolDir) macro in this case.
Arguments **Example:** Arguments = -f $(ToolDir)\\MSStatsGC.r	Tool Arguments. **Note:** The use of $(ToolDir) macro to specify file locations
InitialDirectory **Example**: InitialDirectory = $(DocumentDir)	Initial Directory
InputReportName **Example:** InputReportName = MSstats	The name of the Input Report. **Note:** If a tool has its own report type, make sure its format is provided in a .skyr file in the tool-inf directory.
ArgsCollectorDll **Example:** ArgsCollectorDll = MSStatArgsCollector.dll	Path to the .dll file that implements the args collectors. **Note:** Requires that ArgsCollectorType is also specified.
ArgsCollectorType **Example:** ArgsCollectoryType=MSStatArgsCollector.MSstatsGroupComparisonCollector	Full name of the class that implements the CollectArgs method that should be used for this tool. **Note:** Requires that ArgsCollectorDll is also specified.
OutputToImmediateWindow = True	Specifies that the tool should write output to the immediate window. **Note:** If this line is not included, Skyline defaults

	to false.
Package<n> See Figure 9 below.	Currently only looked at if the ProgramPath macro is in use and the Program is R or Python. Packages will be installed in order from <n> = 1 to <n> = N where N is the highest number. Do not skip any numbers. See Appendix for details on specifying Python and R packages.
Package<n>Version See Figure 9 below.	Sets a minimum version for the corresponding Package<n>.
Annotation<n> See Figure 9 below.	Associates the specified custom annotations with the tool. **Note:** This requires a .sky file containing the custom annotation definitions to be included in the tool directory. When a tool is run, Skyline will check to see if the specified custom annotations are enabled. If not, the tool run is canceled, and Skyline will prompt the user to enable the annotations and fill in data for them.

```
Title = MSstats\\Group Comparison
Command =$(ProgramPath(R,3.0.1))
Package1 = gplots
Package2 = lme4
Package3 = ggplot2
Package4 = limma
Package5 = marray
Package6 = MSstats
Package6Version = 0.99.0
Arguments = -f $(ToolDir)\\MSStatsGC.r --slave --args $(InputReportTempPath)
InitialDirectory = $(DocumentDir)
InputReportName = MSstats
ArgsCollectorDll =MSStatArgsCollector.dll
ArgsCollectorType = MSStatArgsCollector.MSstatsGroupComparisonCollector
OutputToImmediateWindow = True
Annotation1 = Condition
Annotation2 = BioReplicate
Annotation3 = Run
```

Figure 9 - This is an example of a more complicated tool that uses all of the attributes available.

Looking Forward

There will undoubtedly be changes and additions to the process described here. In the future we would also like to collect tools and make an External Tools Library on the Skyline website where people can post tools and browse for tools they would find useful. We would also like to add better ways for tools to affect the current Skyline document.

Appendix

Package Installation

Skyline Installable Tools support in the installation of R and Python packages. These packages are listed in the tool component .properties file or in the info.properties file. Package installation only applies to when the command uses the $(ProgramPath()) macro. If packages are listed in the info.properties, you must provide the command that they are associated with.

```
1  Name = MSstats
2  Version = 0.9
3  Author = Meena Choi, Ching-Yun Chang, Dr. Timothy Clough, Dr. Olga Vitek
4  Description = "MSstats is an R package that provides tools for protein quantific
5  Provider = http://www.msstats.org/
6  Identifier = URN:LSID:www.msstats.org
7  # Optional Specify Required Program and Packages in the info.properties file.
8  # useful when three tools have same program and package dependencies.
9  Command =$(ProgramPath(R,3.0.1))
10 Package1 = gplots
11 Package2 = lme4
12 Package3 = ggplot2
13 Package4 = limma
14 Package5 = marray
15 Package6 = MSstats
```

Figure 10 – An example of listing packages in the info.properties file. Note the Command attribute must be specified in the info.properties file if packages are listed.

R packages:

For R packages we require that there is a provided InstallPackages.r script provided in the tool-inf directory of the zipped tool. We encourage following the format of the provided InstallPackages.r which checks if the package is already installed and only installs the package if it is missing. List packages by name and use the Package<n>Version property when a minimum version is required. The example InstallPackages.r will also show how we installed MSstats by downloading the MSstats.tar.gz then installed from that if the package isn't available from a CRAN mirror.

Python Packages:

Skyline also supports the automated installation of Python packages. Python packages are specified as links to executable (.exe), zip or tar.gz files stored on the web. When a tool is installed, Skyline will do the following:

1. If the packages are specified as web links, Skyline will download them.
2. Skyline will attempt to install the executable (.exe) packages first.
3. If .zip or .tar.gz files are also listed, Skyline will look for the Python package manager Pip on the user's machine. If it cannot be found, Skyline will attempt to install it.
4. When Pip is successfully installed on the user's machine, Skyline installs zip or tar.gz packages using pip install <package>.

```
1  Title = MS1 Probe
2  Command = $(ProgramPath(Python,2.7))
3  Arguments = $(ToolDir)\\MS1Probe.py $(InputReportTempPath) $(DocumentBaseName) $(Doc
4  InitialDirectory = $(DocumentDir)
5  InputReportName = MS1Probe Input
6  ArgsCollectorDll = MS1ProbeArgsCollector.dll
7  ArgsCollectorType = MS1ProbeArgsCollector.MS1ProbeArgsCollector
8  OutputToImmediateWindow = True
9  Package1 = https://pypi.python.org/packages/2.7/n/numpy/numpy-1.7.1.win32-py2.7.exe
10 Package2 = http://downloads.sourceforge.net/project/scipy/scipy/0.12.0/scipy-0.12.0-
```

Figure 11 – An example of installing python packages for MS1 Probe. For python packages provide the link to either the .exe, .zip, or .tar.gz online.

Considerations for specifying Python Packages:

Link Permanence

Links to packages may break when the package maintainer updates the package. Please point to package source files that are as permanent as possible.