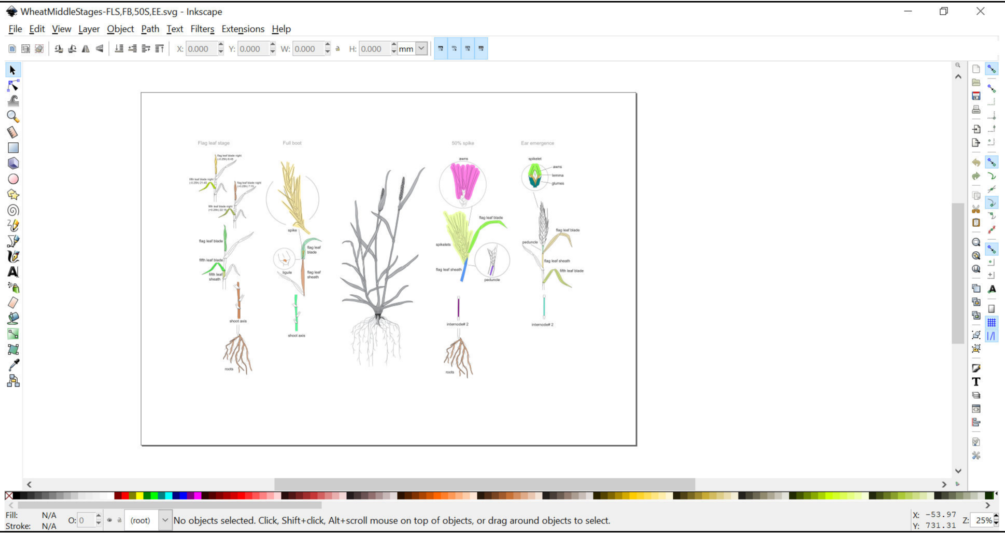
ePlant SVG and Data Guide

Contents

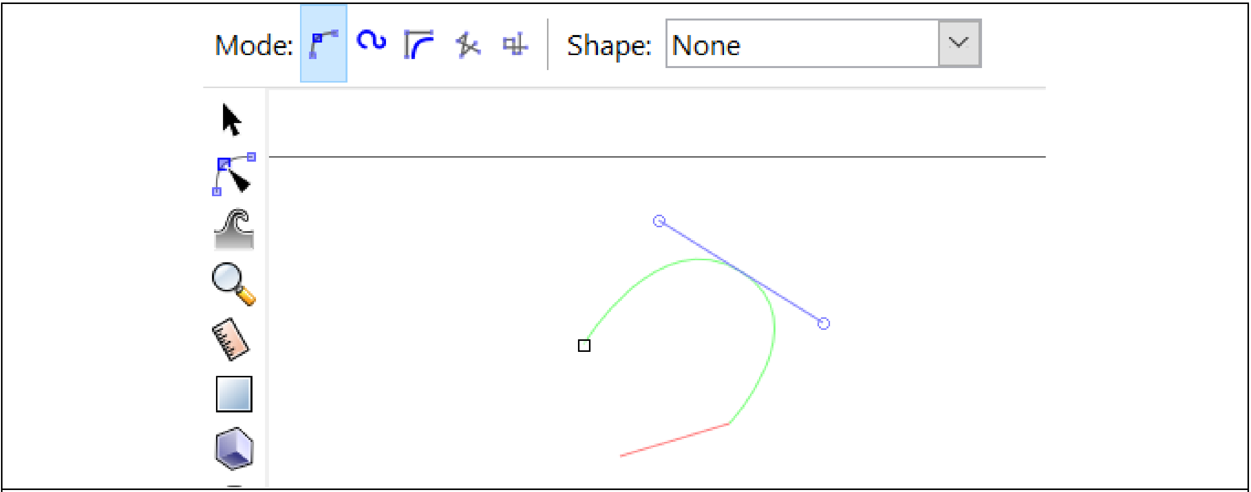
Creating an SVG with Inkscape and creating the corresponding XML

Use an SVG program with a graphical user interface to create an SVG from scratch. We highly recommend Inkscape, and as such the following steps will be described for this program.


WheatMiddleStages-FLS,FB,50S,EE.svg - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
No objects selected. Click, Shift+click, Alt+scroll mouse on top of objects, or drag around objects to select.

SVG of the middle stages of wheat open on Inkscape.

Open Inkscape. If you have a reference image from which to work, open the image or paste it on the canvas. In Inkscape, you draw images with paths connected by nodes. Draw your image with the *Draw Bezier curves and straight lines* tool (shortcut: Shift + F6). Drawing with Bezier curves is the ideal method to maximize on detail while minimizing the number of coordinates needed to represent the path. Begin with your first node by clicking anywhere in the canvas, and continue adding nodes by subsequent clicks elsewhere in the canvas. Finish your path by either closing the path (looping back to your first node) or pressing Esc. If your path is closed, the path fill colour (selected through the bottom colour strip toolbar) will be enclosed within the loop. Otherwise, the fill will be enclosed in the loop that would be created when the first and last nodes are connected by a straight line.

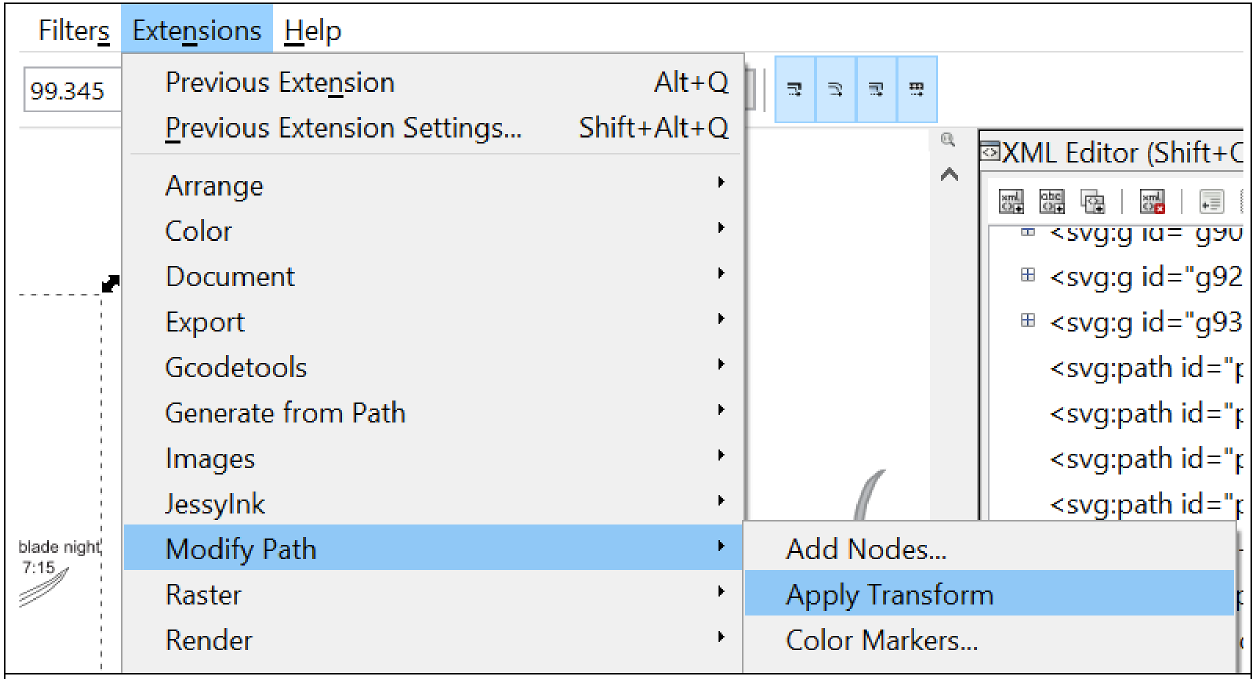


The Bezier tool has many shortcuts. For example, holding while clicking will bend the path parabolically when you drag, but holding Shift while doing this will adjust the axis to which the parabola relatively bends.

You can select and resize your paths or reference image with the *Select and transform object* tool (shortcut: F1), and edit the nodes within a path with the *Edit nodes by path* tool (shortcut: F2). Other tools exist within Inkscape. Familiarize yourself and you'll reap the benefits in improved efficiency and speed.

You will want to create paths in your SVG that will eventually be colour-filled to represent gene expression levels within the tissue that your paths resemble. Select the paths that correspond to one tissue and group them together with Object→Group (shortcut: Ctrl + G). Ensure you have exactly one group for every tissue type your dataset includes, including when the tissue is only represented with a single path (i.e. make sure to group tissues with only one path too). Fill the group with a colour to make sure it fills the tissue representation as expected. Open the XML editor with Edit→XML Editor (shortcut: Shift + Ctrl + X). From here, you can label your group with an id that will then be matched to the contents of your XML file. Ensure your id labels begin with a letter of the alphabet and do not contain any special characters (excepting underscore _ and dash -).

Follow the linked instructions to download Klowner's Inkscape *Apply Transform* tool at Github (you will need to restart Inkscape the first time you download this to use this tool). Deselect any selections you currently have, and apply the tool through Extensions→Modify Path→Apply Transform. This will remove transformations in every group by having their children inherit their transformations. This step is necessary as ePlant can only parse SVG tissue groups without transforms.



"Apply Transform" with no selections will apply transforms directly to the paths of the entire SVG. With selections, it will apply transforms only to the paths within the selection.

Save the file as it is. From here, you will optimize your file in two different ways: first through the "Save as type: Optimized SVG", and second through the script "OptimizeSVG.py" developed by Asher Pasha. As of 2018-04-24, you will need to modify your SVG in order to make it compliant with the OptimizeSVG.py file so it has the following properties:

- The data-fillable groups do not contain any groups i.e. they only contain paths as direct children (if it does contain grouped paths, ungroup those paths)
- Non-filled objects are not grouped with data-fillable groups i.e. data-fillable groups have no parents
- Group all applicable text into a group element with attribute id="label" and group all applicable outlines into a group element with attribute id="outlines"
  - Within the raw script, move these elements so they precede all other object elements to ensure that they do not cover the fillables

You will also need to move all text 20 pixels to the right, as Asher's script (as of 2018-04-24) moves all text 20 pixels to the left. Ensure that you Apply Transform again after this.

Save your file normally under a different name, then save again under another different name as an Optimized SVG (under the "Save as type:" dropdown menu). You will be prompted to select the parameters for your Optimized SVG. Select the following:

Options	✓ Work around render bugs Keep everything else unchecked
SVG Output	**Document options** ✓ Remove metadata ✓ Remove comments ✓ Embed raster images **Pretty printing**

	✓ Format output with line-breaks and indentation ✓ Strip the “xml:space” attribute from the root SVG element Keep everything else unchecked
IDs	✓ Preserve manually created IDs not ending with digits Keep everything else unchecked or unfilled

Now, run your optimized SVG file through the OptimizeSVG.py script. This can be obtained from the BAR’s github repo (download raw or pull to your own local directory). As of 2018-04-19, you will need to change the script directly to take your file: within the main() function (line 121+), change the assignment of old_svg_file to the name of your once-inkscape-optimized SVG, and change the assignment of new_svg_file to your desired output SVG name.

To create an XML metadata file, you can save any text file with a name ending in .xml . We highly recommend you use a source code editor such as Sublime Text, which automatically color-codes features of interest and has several options and functions for ease of use.

Ensure the header is proper. Below is an example of a proper header from the Klepikova Atlas dataset. Adjust it appropriately for your specific dataset.

```
<!-- Header of XML -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE specimen SYSTEM '//bar.utoronto.ca/efp/cgi-bin/data/efp_config.dtd'>
<specimen name="Arabidopsis">
    <info>
        <li>Data from A high resolution map of the Arabidopsis thaliana developmental tr
Klepikova et al., 2016, Plant J. 88:1058-1070.</li>
        <li>Total RNA was extracted with RNeasy Plant Kit and Illumina cDNA libraries we
manufacturer's protocols. cDNA was then sequenced using Illumina HiSeq2000 with a 50bp r
available in NCBI’s Sequence Read Archive under the BioProject ID 314076 (accession: PRJ
        <li>Reads were aligned to the reference TAIR10 genome (Lamesch et al., 2012) usi
        <li>Reads per gene were counted with an in-house Python script using functions f
2015).</li>
        <li>RPKM values were compiled using an in-house R script.</li>
    </info>
    <view class="" db="klepikova" img="Klepikova.svg" name="all">
        <!-- Body of XML -->
    </view>
</specimen>
```

In the <info> element, you will want to put a summary of from where the data originates, how it was collected, and (if applicable) any other data transformations applied. This will be presented in bullet points, with each point declared with the <li> element. Within the <view> start tag, match the appropriate database (db) name found in the BAR.

For the body of the XML, you will organize your declared tissues in groups. These groups will be based on which tissues share the same control. If all the tissues share the same control, only a single group is necessary. The below example is adapted from the XML for Tomato ILs in Leaf.

```
<!-- Body of XML -->
<group name="CTRL_MED">
    <control sample="CTRL_MED"/>
    <tissue colorKey="#ff0000" id="IL1_1" name="IL1.1">
        <link url="http://bar.utoronto.ca"/>
        <area coords="59,92,84,92,84,78,59,78"/>
        <sample name="IL1.1"/>
    </tissue>
    <tissue colorKey="#FF1400" id="IL1_1_2" name="IL1.1.2">
        <link url="http://bar.utoronto.ca"/>
        <area coords="59,108,84,108,84,93,59,93"/>
        <sample name="IL1.1.2"/>
    </tissue>
    <!-- More tissues -->
</group>
<!-- More groups if necessary -->
```

Each <group> element can specify multiple controls with <control> elements. Oftentimes, the control is the median of the entire dataset or within the group. Thereafter, tissues can be specified with a <tissue> element. The start <tissue> tag should contain the id and name attributes. The id attribute matches the XML <tissue> element to the SVG path group ids, while the value specified in the name attribute will be the name displayed when mousing over the SVG path group in the final ePlant viewer. The <sample> element matches the XML <tissue> element to the actual data (labeled the same as the contained name attribute value) within the database specified in the header.

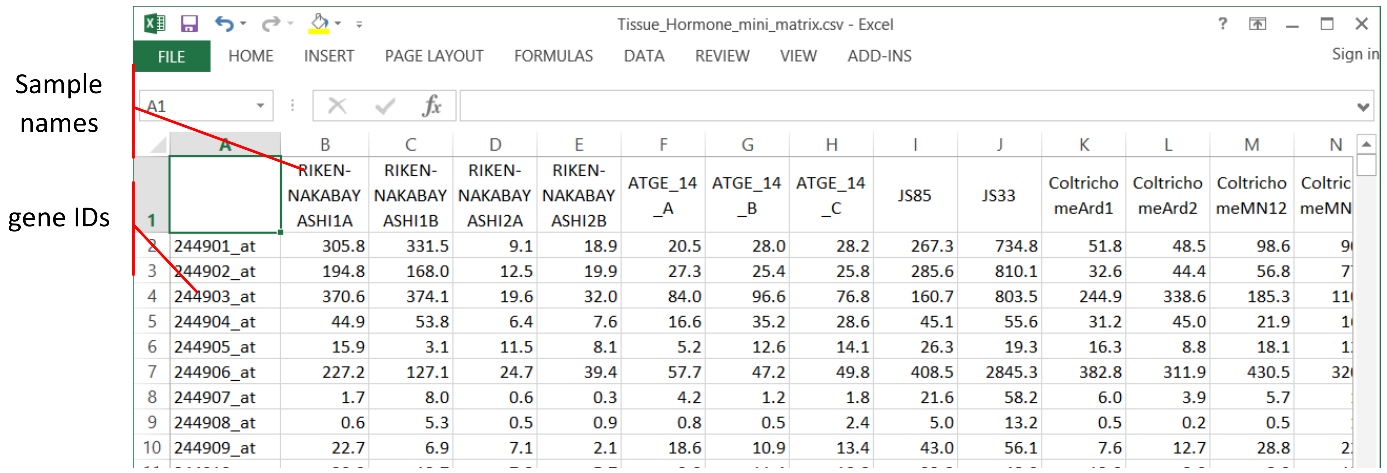
Note that certain information in this above example is not needed for ePlant (they are left over from the original eFP Browser XML from which the ePlant version was adapted). These include the colorKey attribute, <link> element, and <area> element.

Ensure the XML is well-formatted (you can validate this at https://www.xmlvalidation.com/). Any problems with the XML will prevent the SVG from being displayed at ePlant at all. Similarly, ensure your SVG ids match your XML ids and that your SVG groups have no other attributes besides the id.

Data Guide

Please share a data matrix with nicholas.provart@utoronto.ca using Dropbox or a similar file sharing service.

For every part or treatment <group> depicted in your SVG image, you require at least one set of measurements. It is typical, however, to have biological replicates, i.e. duplicated or triplicated measurements. You will be associating these measurements (samples) with the parts or treatments you have depicted in your image with the XML file described above. You can use any normalization/summarization method you wish. For RNA-seq data we recommend FPKM or TPM.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		RIKEN-NAKABAYASHI1A	RIKEN-NAKABAYASHI1B	RIKEN-NAKABAYASHI2A	RIKEN-NAKABAYASHI2B	ATGE_14_A	ATGE_14_B	ATGE_14_C	JS85	JS33	ColtrichomeArd1	ColtrichomeArd2	ColtrichomeMN12	ColtricmeMN
2	244901_at	305.8	331.5	9.1	18.9	20.5	28.0	28.2	267.3	734.8	51.8	48.5	98.6	9
3	244902_at	194.8	168.0	12.5	19.9	27.3	25.4	25.8	285.6	810.1	32.6	44.4	56.8	7
4	244903_at	370.6	374.1	19.6	32.0	84.0	96.6	76.8	160.7	803.5	244.9	338.6	185.3	11
5	244904_at	44.9	53.8	6.4	7.6	16.6	35.2	28.6	45.1	55.6	31.2	45.0	21.9	1
6	244905_at	15.9	3.1	11.5	8.1	5.2	12.6	14.1	26.3	19.3	16.3	8.8	18.1	1
7	244906_at	227.2	127.1	24.7	39.4	57.7	47.2	49.8	408.5	2845.3	382.8	311.9	430.5	32
8	244907_at	1.7	8.0	0.6	0.3	4.2	1.2	1.8	21.6	58.2	6.0	3.9	5.7	
9	244908_at	0.6	5.3	0.5	0.9	0.8	0.5	2.4	5.0	13.2	0.5	0.2	0.5	
10	244909_at	22.7	6.9	7.1	2.1	18.6	10.9	13.4	43.0	56.1	7.6	12.7	28.8	2

Data matrix of expression values with one gene per row, and one sample per column. Cells contain the expression level for a given combination. Here the first 2 data columns are biological replicates for the dry seed sample. Note that sample names must match up with those that will appear in the XML file described above.

Save this file as a CSV (comma separated values) file. The ePlant engine uses a linear colour scale scheme so don't submit log-transformed values.