链滴

一种根据实验室仪器导出的双列 ASCII 数据集进行同时绘图的方法

作者：Roseleaves

原文链接：https://ld246.com/article/1666952327894

来源网站：链滴

```
mypath = "";
filename = {"", "", ""};
reflectances = Drop[#, 90]& @ Import[mypath <> # <> ".asc", "Table"]& /@ filenames;
maxt = Max[ #[[2]]& /@ Select[#, #[[1]] > 600 &] ]& /@ reflectances;
(* maxt[[-1]] = 103; *)

absorbances = Table[ {#[[1]], maxt[[i]] - #[[2]]}& /@ #[[i]], {i, Length[#]}]&[reflectances];
maxa = Max[ #[[2]]& /@ Select[#, 300 < #[[1]] < 600 &] ]& /@ absorbances;
(* maxa[[1]] = 100; *)

unified = Table[ {#[[1]], #[[2]] / maxa[[i]]}& /@ #[[i]], {i, Length[#]}]&[absorbances];

plot1 = ListLinePlot[unified,
  PlotRange -> All,
  PlotLegends -> Placed[
   LineLegend[
      filenames,
      LegendFunction -> "Frame",
      LegendLayout -> "Column"],
   {{0.75, 1}, {0.5, 1}}],
  AxesLabel -> {
   ToExpression["\\lambda (\\rm{nm})", TeXForm, HoldForm],
   "Absorbance"
   }]
Export[mypath <> "diagram_uniform.svg", plot1];
```

主要是在这其中学习 Mathematica 的语法。

Mathematica 的配置：

- 下载 Wolfram Script
- 下载 VSCode，安装其中的 Wolfram Language Notebook 扩展。
- 总之照着它的说明配置就是了。

先说它的运算符吧。在 Mathematica 里面，字符串的连接用的不是 + 而是 <> 。因此 mypath <> "iagram_uniform.svg" 就是把我主动填写的目录字符串（注意这里代码显示的是空字符串，但具体执的时候要补全！）和具体的文件名连在一起。

非常推荐把作出来的图 plot1 用 SVG 或 EPS 格式导出为矢量图。默认的位图 dpi 太低，导出的图特糊。

然后是箭头 -> 。它在 Mathematica 官方付费的记事本里面会被自动转换为 → ，我是相当的不喜欢个设计，它会让初见者为如何输入它感到费解。但是 VSCode 里面就会忠诚地显示为 -> 啦，令人安。它的意思是规定那些【可选】参数的性质。比如说 PlotLegends -> filenames 就是用 filenames 列的每一个字符串逐一给 ListLinePlot 的主参数 unified 所对应的每条曲线标注在图例上。理所当然这整一句都可以去掉，不影响作图，只是图例没了。

然后是括号 (), [], {}, [[]] 。圆括号 () 只用来支撑运算顺序，单方括号 [] 构筑成了几乎所有的函数调用花括号 {} 用来构筑序列，包括矩阵，总之就是搭建有序元组，而双方括号 [[]] 则用来从有序元组中取元素，是从1开始计数的。

然后是 @, /@ 。函数调用不是必须用 []的，它还有非括号的表达方式用的是 @比如输入 Sin @ Pi 得到 0。而相应地，函数调用就可以在序列上进行，相应的用的是 /@。比如输入 Sin @ {Pi/6, Pi/4,

i/3, Pi/2, Pi} 会得到 \{\frac12, \frac{1}{\sqrt{2}}, \frac{\sqrt{3}}{2}, 1, 0\}。

```
In[1]:= Sin @ (Pi/6)

        1
Out[1]= -
        2
In[2]:= Sin @ {Pi/6, Pi/5, Pi/4, Pi/3, Pi/2, Pi}

        1      5   Sqrt[5]      1     Sqrt[3]
Out[2]= {-, Sqrt[- - -------], -------, -------, 1, 0}
        2      8      8       Sqrt[2]     2
```

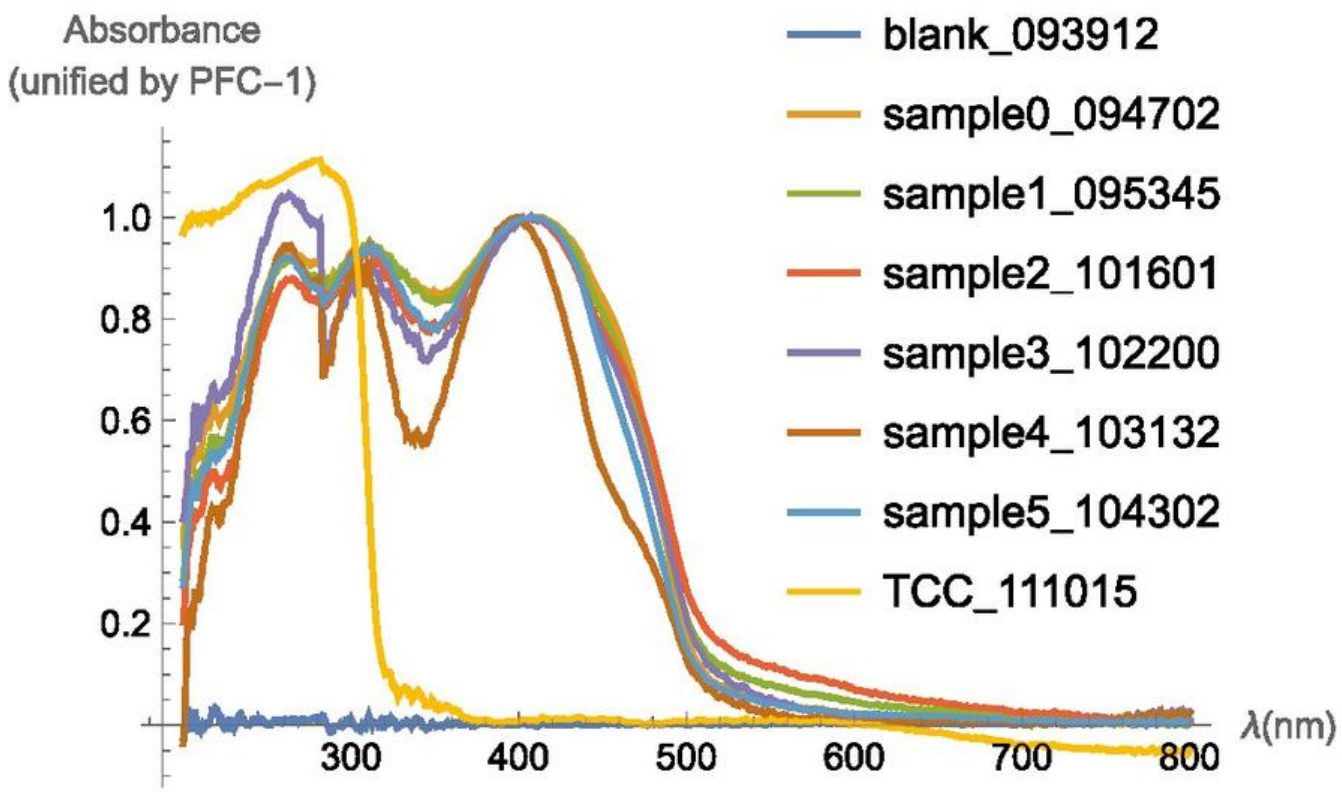
值得注意的是，这个 /@ 算符（或者说 f /@ expr ）的波兰式（把算符写在参与运算各项的前面）表就是 Map[f, expr]，它把函数 f作用到序列 expr第一层的每一个元素上，得到一个元素数目和 expr同的序列。也就是说，**它就是map**。在我的数据处理中，map实在是发挥了重大作用。

然后是 #, & 。在我之前的帖子氢原子的 d 轨道概率密度函数的三维密度图 中就显式地定义过含参函，诸如 r[x_, y_, z_] := Sqrt[x^2 + y^2 + z^2]; ，这里的 x_ 以下划线结尾表示对傀儡变元的声明，之后应用该变元时则使用它去掉末尾下划线的形式 x。而这里我们想把事情做得更加简单，匿名函数是以 & 结尾的一串表达式（不过如果它包含加减乘除等运算，那么还是需要用圆括号框住运算结果这和函数的非括号调用遥相呼应，比如 (#+1)& @(5/2)），如果它参数只有一个，那么用 #表示，否依序用 #1, #2, ...表示。比如匿名函数 #[[2]]&，它提取出来唯一参数的第2个元素。看上去搁这儿搁，但是想把一列数对 pts的第2个元素一起提出来时，对它的使用 #[[2]]& /@ pts就可以简捷达成目。除此之外，我程序中 Drop[#, 90]& @ Import[mypath <> # <> ".asc", "Table"]&的使用就是把数名 Drop和第二参数 90放在一起，而不是被超长的第一参数隔开，由此达到方便阅读的功效。不要，Mathematica 解释器会知道两个 #和两个 & 该如何配对，大不了用圆括号。

除此之外，由于使用了相当长的变量名，所以诸如 unified = Table[{#[[1]], #[[2]] / maxa[[i]]}& /@ [[i]], {i, Length[#]}]&[absorbances]; 的处理，能有效地减小代码长度，方便更改变量名时少改几处以及把作用的变量提到行尾，更加明显。

好了，我觉得我能讲的就这么多。其他诸如 Select[] 函数的使用啊，PlotLegends -> Placed[LineLe end[]] 的具体细节啊，我都是偷的，是□□kr
，读者自己去看MAA官方文档就行了。

附赠一个效果图。



原文链接：一种根据实验室仪器导出的双列 ASCII 数据集进行同时绘图的方法