Kinome-render and AnnotationFromBindingData Read-Me

By Thierry Chénard[1], Matthieu Chartier[1], Jonathan Barker[2], Rafael Najmanovich[1]
[1] Département de Biochimie, Faculté de médecine et des sciences de la santé, Université de Sherbrooke, Québec, Canada
[2] European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK

Reference:
Chartier, M., Chénard, T., Barker, J., & Najmanovich, R. (2013). Kinome Render: a stand-alone and web-accessible tool to annotate the human protein kinome tree. PeerJ, 1, e126. doi:10.7717/peerj.126

Kinome-Render

Kinome Render is a perl script that creates a postscript output based on a template and an input file containing the different kinases to print.

Figures used or published using Kinome Render must be accompanied by the acknowledgement, "Illustration reproduced courtesy of Cell Signaling Technology, Inc. (www.cellsignal.com)".

How To Run Kinome Render Locally

1. Open a terminal, console or command prompt window.
2. Go to the directory containing the kinome-render script using the "cd my-folder" command
3. Run the kinome-render program with the following arguments

-i <input_file>
-o <output_file>
-t2 (optional)

$ kinome-render -i input.txt -o output.ps -t2

This line runs Kinome Render using template 2 and the information in input.txt to create the output.ps file.

The input file contains the information on the annotation you want to add to the template. See the Kinome Render paper for more information on the syntax.

The output_file is the path and name of the postscript file you want to create.

The default template is template.ps. It contains the basic kinase tree without the atypical kinases. If you have created annotations on atypical kinases in the input file, use the argument -t2 to use template2.ps. This template contains the kinase tree and a sub-tree

with the atypical kinase families. The postscript templates are in the postscript directory and they should not be modified nor moved.

############################
Additionnal Information
############################

The names of the kinase in an "at" line must correspond exactly (it is case sensitive) to the names in the column "Name" of Table S1.

"boxed" and "underlined" must be used after the text is defined and before the next "at" line.

The scale and color must be set before the text or shape are defined and will remain the same until a new scale or color is defined.

The number of sides in a polygon must be an integer greater than 2.

To print special characters, you must put the code of the symbol name of the letter (in brackets []). For example:

scale 30
color 0 0 1
text CK1[gamma]1

will print CK1[gamma]1 in blue size 30 (with the actual greek letter instead of [gamma]) at the position of CK1[gamma]1 on the tree.

Various other special characters can be included in the annotation using []. To see a list, refer to the symbols.pdf file.

################################
AnnotationFromBindingData
################################

AnnotationFromBindingData.pl is a perl script that automatically creates kinome-render input files and runs kinome-render with those files based on a tab separated table containing binding affinities. Please refer to the online Guide for more info (bcb.med.usherbrooke.ca/kinomerender).

##
How to Use AnnotationFromBindingData Locally
##

1. Open a terminal, console or command prompt window.
2. Go to the directory containing the AnnotationFromBindingData.pl and kinome-render scripts using the "cd my-folder" command
3. Run the AnnotationFromBindingData.pl program with the following arguments

-i <input_file>
-o <outputprefix> (optional)
-t <threshold> (optional)
-y <IdType> [Name, Uniprot, IPI] (optional, default=Name)
-n <inhibitors per image> (optional, default=6)
-nm (optional)
-ic (optional)
-ip (optional)
-p (optional)
-s <scale> (optional)

-i: The input file must be a tab delimited plain text file where each line represents a particular kinase and each column an inhibitor. Accordingly, the first row should contain inhibitor names and the first column either kinase names (column "Name" in Table S1), UniprotIDs or IPIs. Any particular cell should contain the Kd, IC50 or a %inhibition. All values in the table need to have the same unit of measurement (micromolar or nanomolar, not written with the numbers). A dash “-” should be used for tested kinase/ inhibitor combinations where no effect was observed. If a kinase/inhibitor combination was not tested, "NA" should be written at the corresponding position in the input file.

-o: the outputprefix is the prefix of the names of the files (including the path) that will be produced by the script.

-t: the threshold is the maximum value in the chosen unit which should be shown in the final figures. An integer threshold can be given and a kinase/inhibitor combination with a value above this threshold will be considered to have no effect.

-y: the IdType has three possible values: Name, Uniprot, IPI. This represent the type of identifier used in the input file. In the input file, Names must be those in column Name of Table S1 (case sensitive), Uniprot IDs and IPIs must correspond to the UniprotID and IPIs in Table S1.

-n: inhibitors per image is a number between 1 and 6 inclusively representing the number of inhibitors to be shown in each representation of the kinome tree. Kinases to be put in the same tree will be taken successively in the input file.

-nm: using the -nm argument means the values in the matrix are expressed in nanomolar otherwise, they are considered as being micromolar. This is needed to display the legend properly.

-ic: use if the values in the matrix are IC50. Otherwise, they are considered to be Kd. Again, this is needed strictly to draw the legend properly.

-ip: using the -ip argument means that the values in the matrix are inhibition percentage. This argument should not be used with the -ic, -p or the -nm arguments.

-p: can only be used if the -n argument is set to 1. If this argument is used, all tested kinases with values above the threshold (defined with ‘-t’) or with a "-" (in the input file) will be printed in grey circles of size 20.

-s: using the -s argument multiplies the scale values of each shape by the arguments value. ex: -s 0.5 for half size.