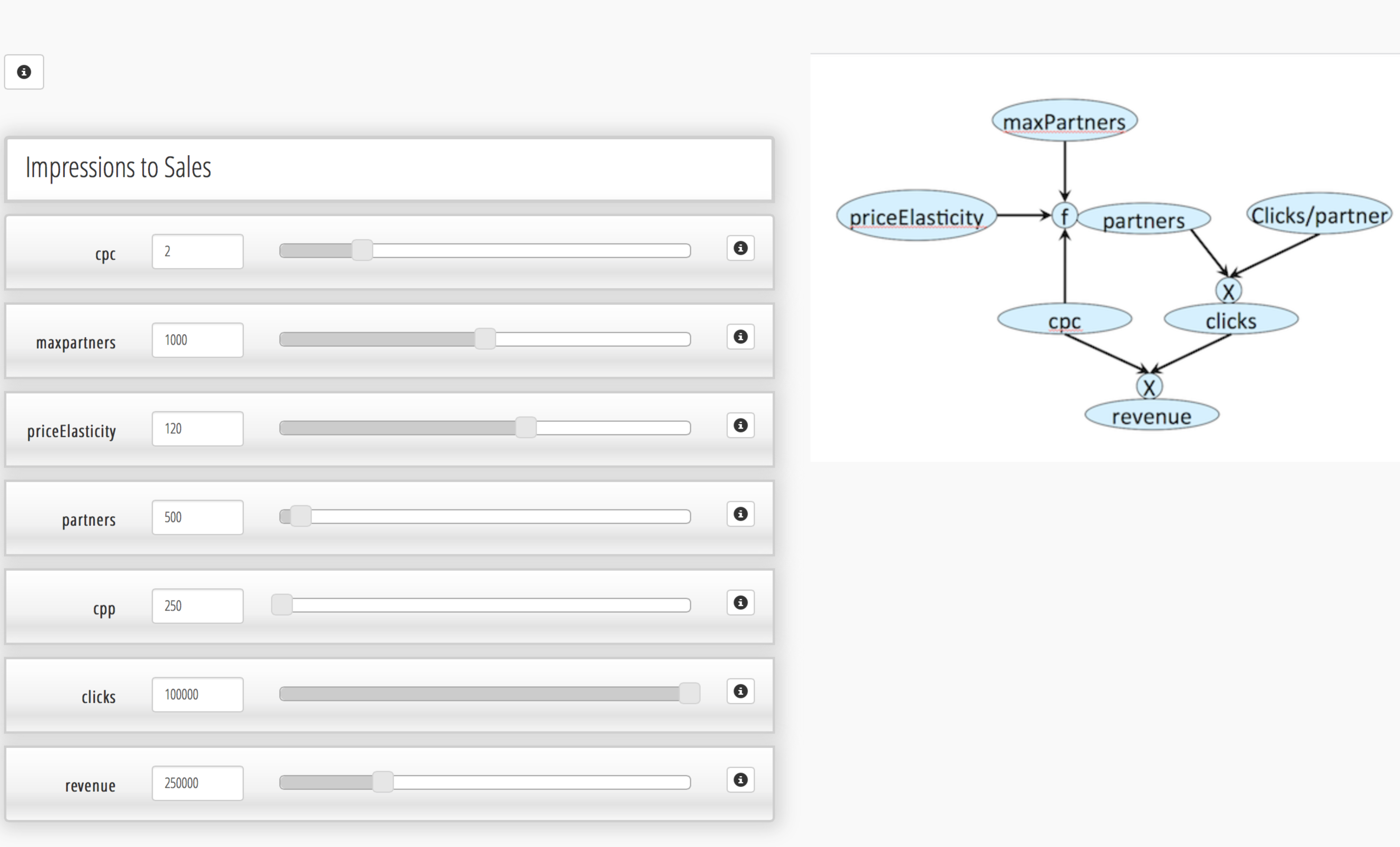
VIN – User manual



Here’s a version of an implementation of a variable interaction network (a.k.a. VIN) in angularJS.

I feel I should write some more “how can this kind of thing be useful” documentation to explain/defend the concept, but no time for this now.

Never-the-less, for those who believe, here is some javascript code you can play with and build on. There’s lots that could be better (and I’ll send you some further better versions, for some time, if you want), but it’s a working work in progress.

**Quick what-is and how-to**

To view, unzip and open index.html in a browser. Note: Sometimes it doesn’t work in Chrome by double-clicking. In this case use another browser or copy the path to the index.html and paste it to in the navigation bar. That works for me.

The specific network is a simple impressions to clicks funnel, where impressions become clicks via a ctr (click thru rate) and clicks become sales through a cvr (conversion rate).

You can change values of each of the parameters and observe the effects on others. The effects are only ordered according to the order of the variables are displayed on the page. You can change this order by dragging variable containers to different positions.

In a nutshell, higher variables are used as causal or explanatory variables of lower ones. (Technical detail: The reason for needing some kind of ordering is that of multiple choices of tree traversal).

For example, say you want to know how many clicks you’d need to get to be able to make a given number of sales. In the original order, if you edit sales, this will have no effect on clicks. This is because clicks is above sales. If you drag clicks to be below sales, changing sales will now change clicks. The number of clicks you see is the number of clicks you’d need for the sales you specified, at the given cvr (conversion rate). Try increasing cvr now, and see that (with the same number of sales) you need less clicks.

But now the impressions and ctr are not aligned with these clicks. This is because clicks is below imps and ctr. If you wanted to see what ctr you’d need to reach, for those clicks, with the impressions you have, you’d need to drag ctr below clicks and do what you just did.

Not perfect, lots of improvements possible, but it’s a beginning.

**Building your own network**

If you want to try to write your own network, look into app.js. At the top of the page you can name your network, write some info on it, and especially, define it.

To define it you need to edit the json-like network\_vars variable.

This variable is a JSON of elements such as:

clicks: {

links: [{

inputs: ['imps', 'ctr'],

relation: 'multiply'

}, {

inputs: ['sales', 'cvr'],

relation: 'divide'

}],

value: 50.0,

viewname: 'clicks',

minvalue: 0.01,

maxvalue: 1000,

step: 0.001

},

Edit clicks (in key and in viewname value) to change the name.

Edit value to set a new default value.

minvalue, maxvalue, and step are for the slider.

But now the important part.

You need to specify how this variable (node) is connected to others through the link field.

The link field contains a list of {inputs, relations} which specify a “relations” function that should take said inputs (variables) to compute clicks. The reason there is several relations, is that clicks can be computed as imps \* ctr or sales / cvr.

—> You need to specify ALL relations or the network won’t work correctly.

—> The relation that is used to compute is the first one that will be found with all inputs ABOVE the variable that needs to be computed. So, when clicks is recomputed, the program looks to see if imps and ctr are above (on the screen), if there not, the program looks to see if sales and cvr are above clicks, and if they are, these are used to recompute clicks.

If you want to change the image, just change the img/bg1.png image.