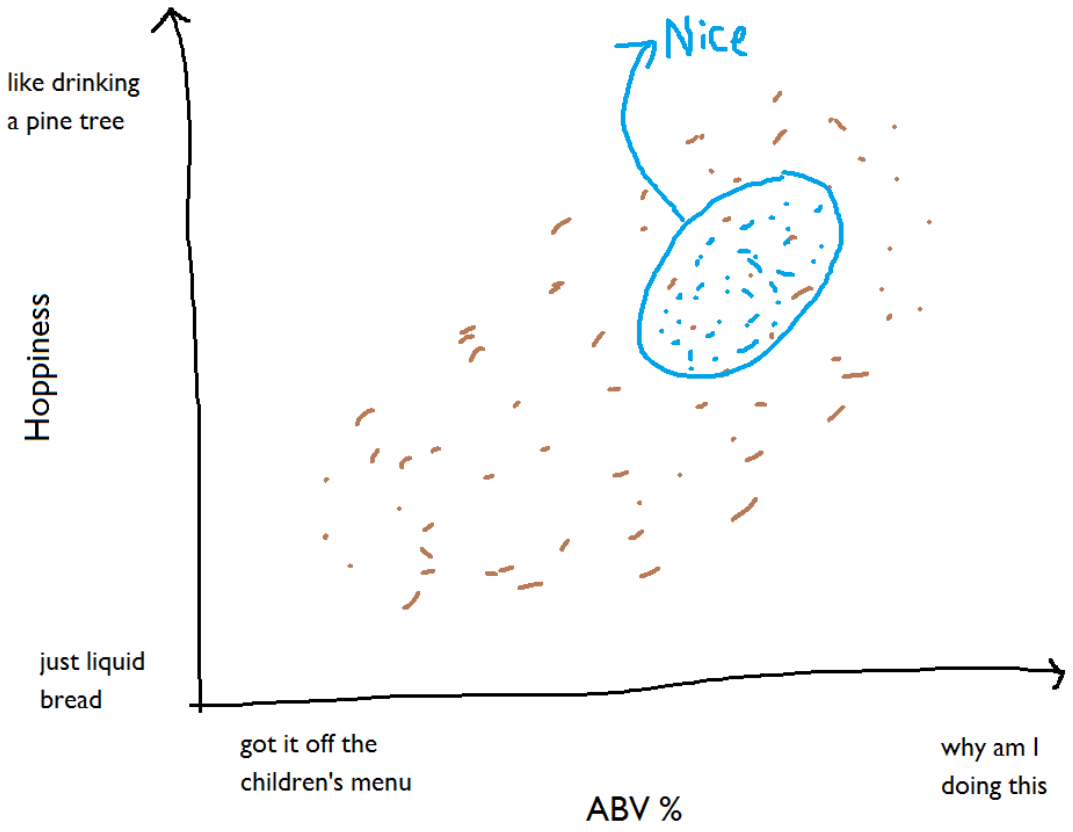
The Information La

<< Back



26 May, 2017 | Gwilym Lockwood

Everything you ever wanted to know about the Mahalanobis Distance (and how to calculate it in Alteryx)

(developed and written by Gwilym and Bethany)

posted the link? is the title interesting? does this sound relevant to your own work? does it have a nice picture? – weighed them up in your mind, and thought "okay yeah, I'll have a cheeky read of that". Other people might have seen another factor, like the length of this blog, or the authors of this blog, and they'll have been reminded of other blogs that they read before with similar factors which were a waste of their time. They'll have passed over it.

This kind of decision making process is something we do all the time in order to help us predict an outcome – is it worth reading this blog or not? There are loads of different predictive methods out there, but in this blog, we'll focus on one that hasn't had too much attention in the dataviz community: the Mahalanobis Distance calculation. And we're going to explain this with beer.

(for the conceptual explanation, keep reading! But if you just want to skip straight to the Alteryx walkthrough, click here and/or download the example workflow from The Information Lab's gallery here)

work consuming as little of your precious time as you can. But (un)fortunately, the modern beer scene is exploding; it's now impossible to try every single new beer out there, so you need some statistical help to make sure you spend more time drinking beers you love and less time drinking rubbish.

Luckily, you've got a massive list of the thousands of different beers from different breweries you've tried, and values for all kinds of different properties. You've got a record of things like; how strong is it? How bitter is it? What sort of hops does it use, how many of them, and how long were they in the boil for? What kind of yeast has been used?

If you know the values of these factors for a new beer that you've never tried before, you can compare it to your big list of beers and look for the beers that are most similar. This is the K Nearest Neighbours approach. For a given item (e.g. a new bottle of beer), you can find its three, four, ten, however many nearest neighbours based on particular characteristics. So, if the new beer is a 6% IPA from the American North West which wasn't too bitter, its

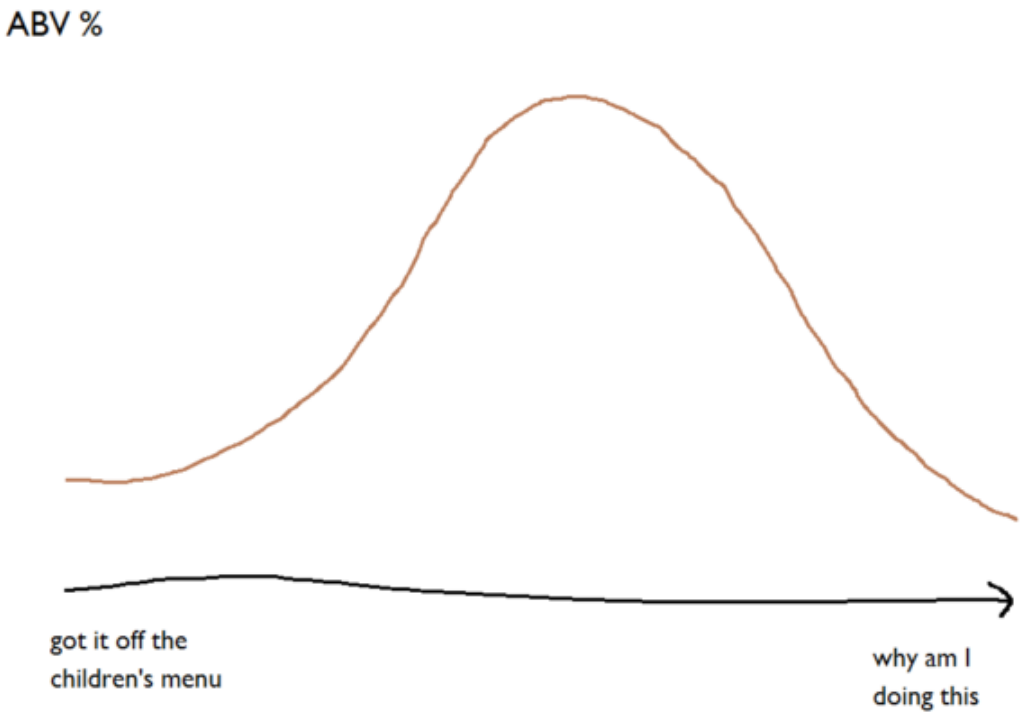
nearest neighbours before, and you liked them, then great! This new beer is probably going to be a bit like that. But if you thought some of the nearest neighbours were a bit disappointing, then this new beer probably isn't for you. The distance between the new beer and the nearest neighbour is the Euclidian Distance.

The Mahalanobis Distance is a bit different. Look at your massive list of thousands of beers again. You've probably got a subset of those, maybe fifty or so, that you absolutely love. They're your benchmark beers, and ideally, every beer you ever drink will be as good as these. The Mahalanobis Distance is a measure of how far away a new beer is away from the benchmark group of great beers.

To show how it works, we'll just look at two factors for now. Let's say your taste in beer depends on the hoppiness and the alcoholic strength of the beer. You like it quite strong and quite hoppy, but not too much; you've tried a few 11% West Coast IPAs that look like orange juice, and they're not for you.

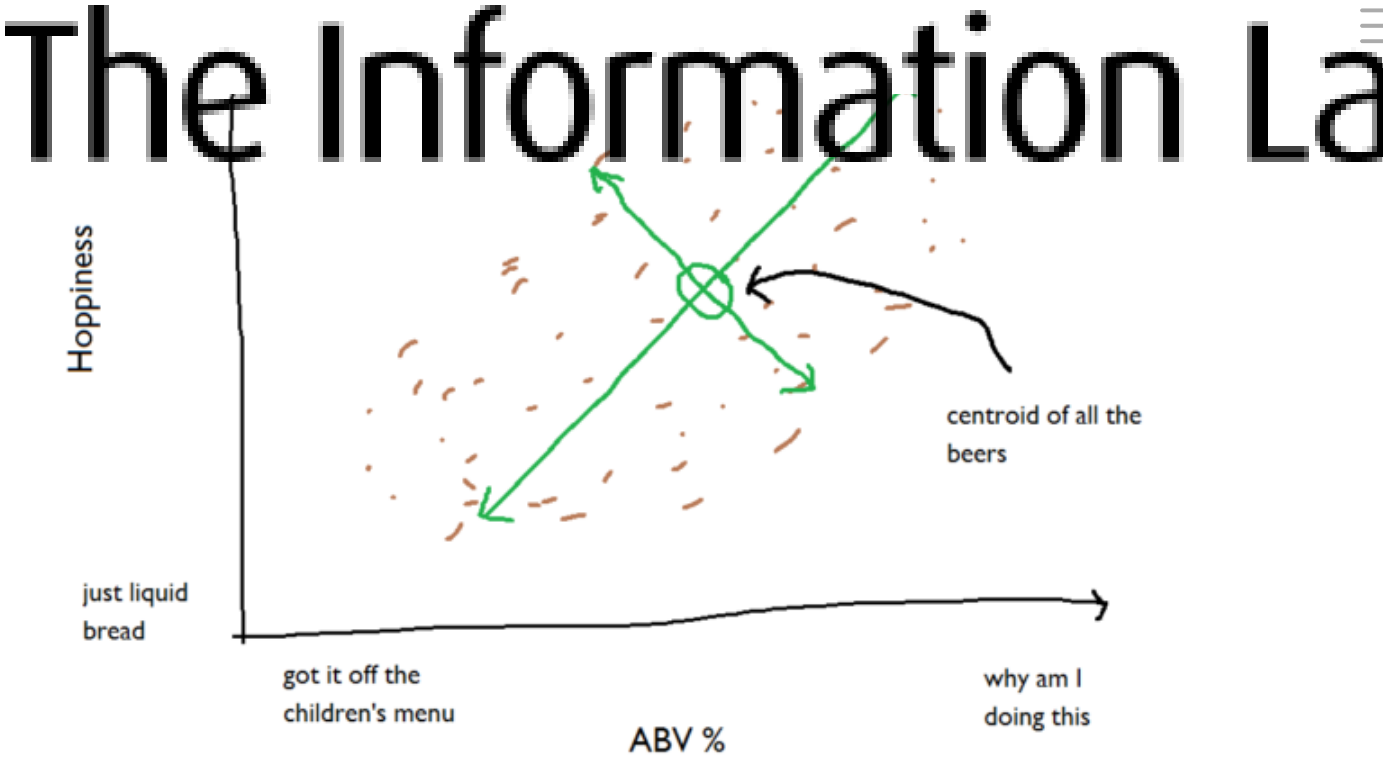
Because there's so much data, you can see that the two factors are normally distributed:

The Information La

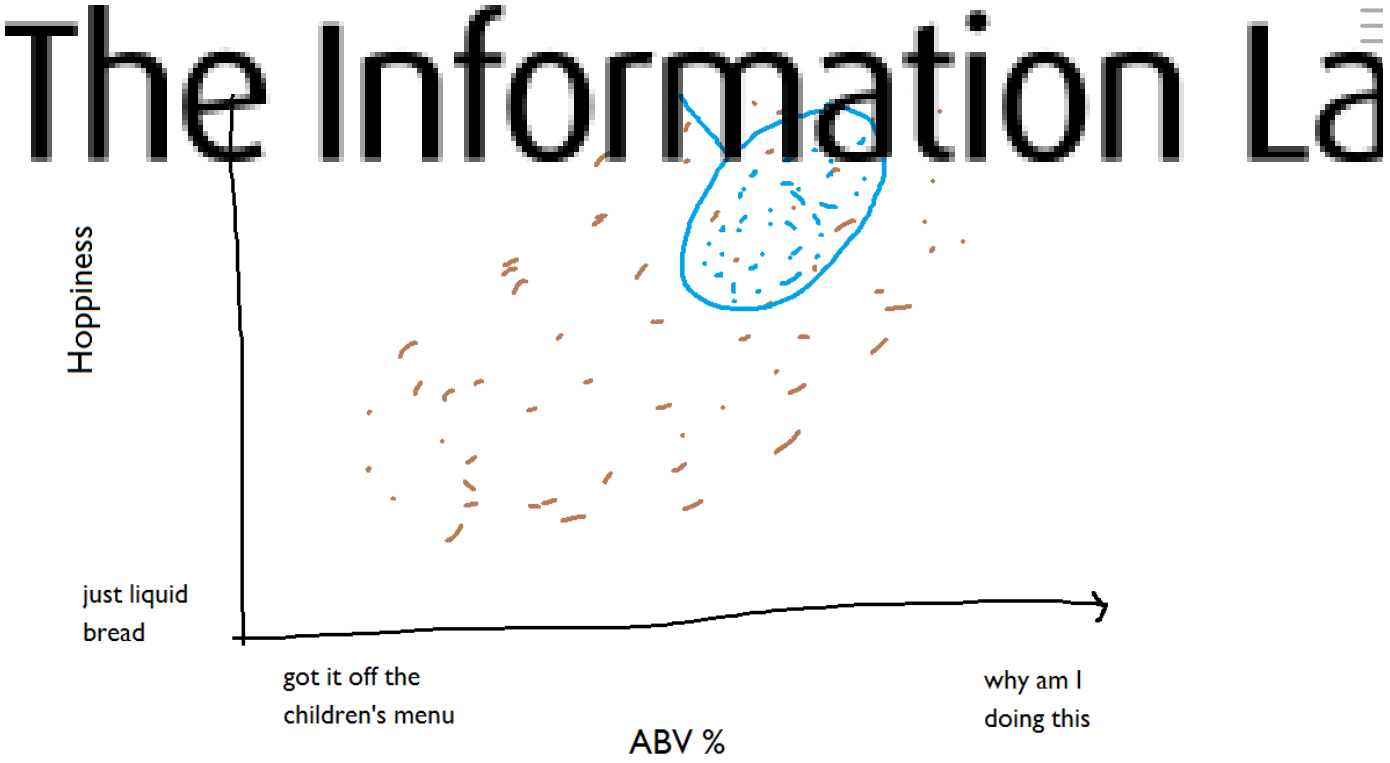


The Information La

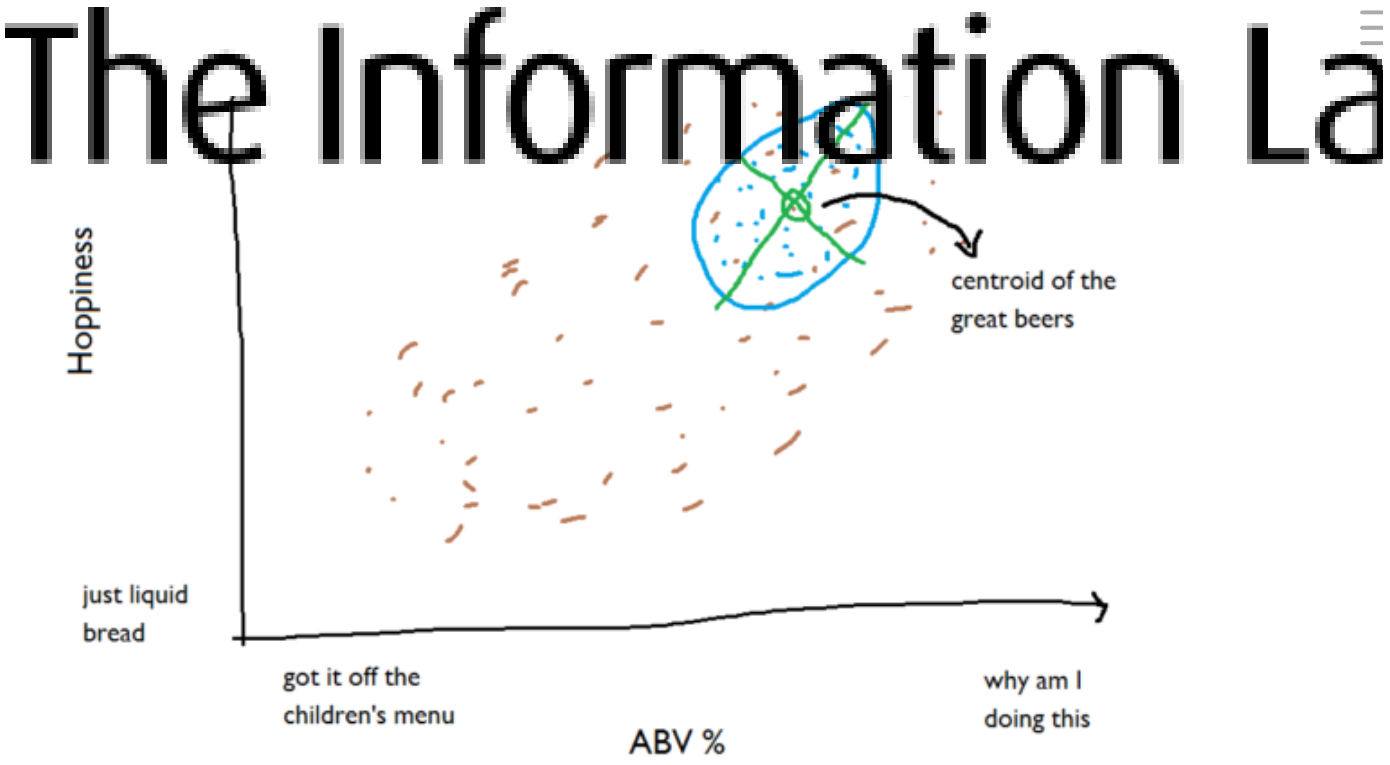
The distribution of the cloud of points means we can fit two new axes to it; one along the longest stretch of the cloud, and one perpendicular to that one, with both axes passing through the centroid (i.e. the mean ABV% and the mean hoppiness value):



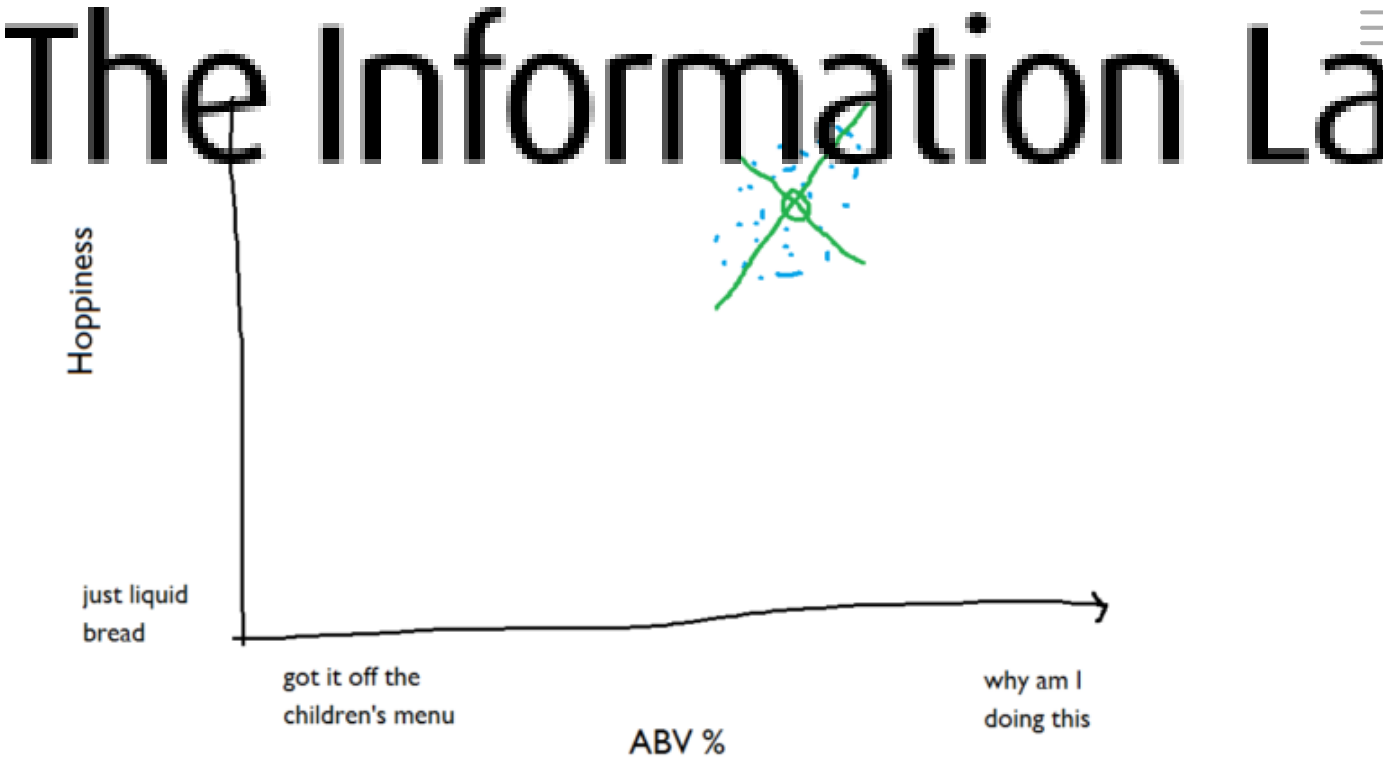
This is all well and good, but it's for all the beers in your list. Let's focus just on the really great beers:



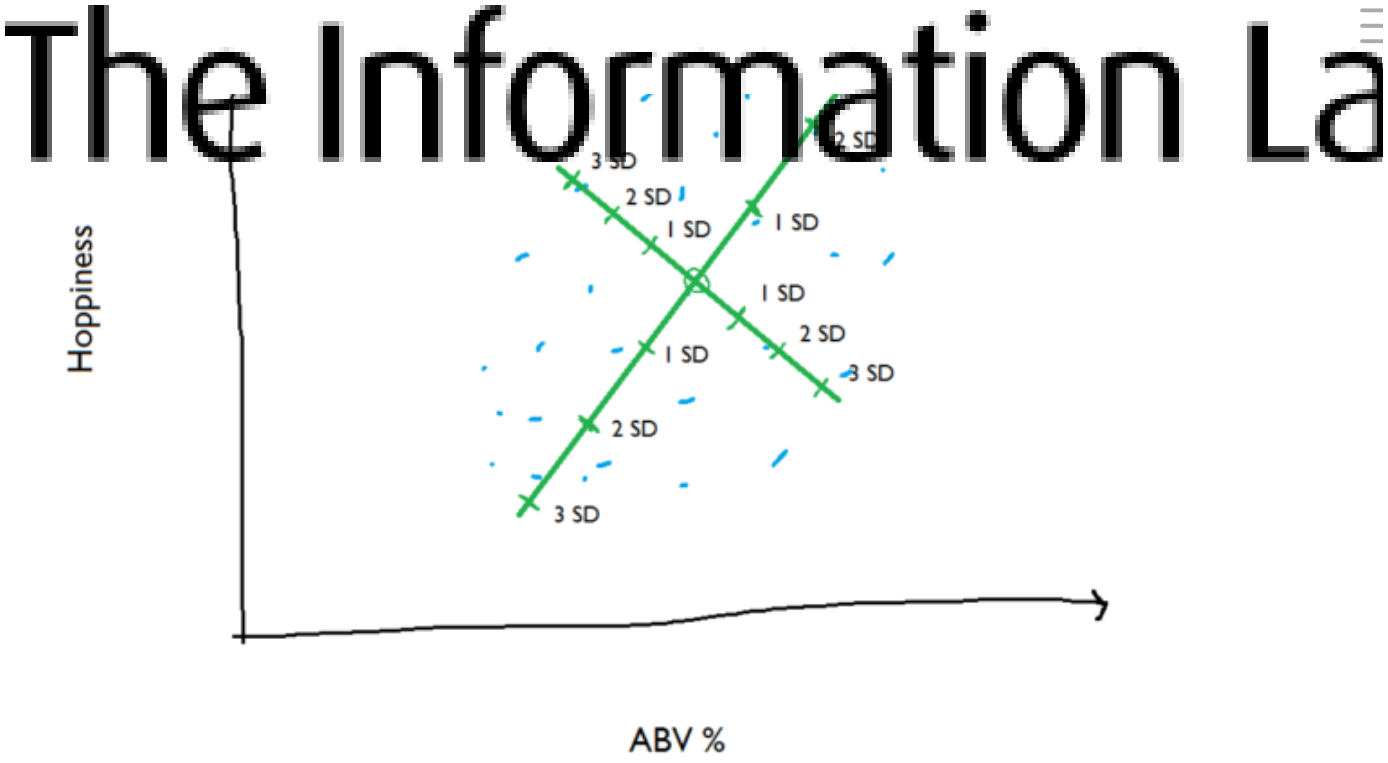
We can fit the same new axes to that cloud of points too:



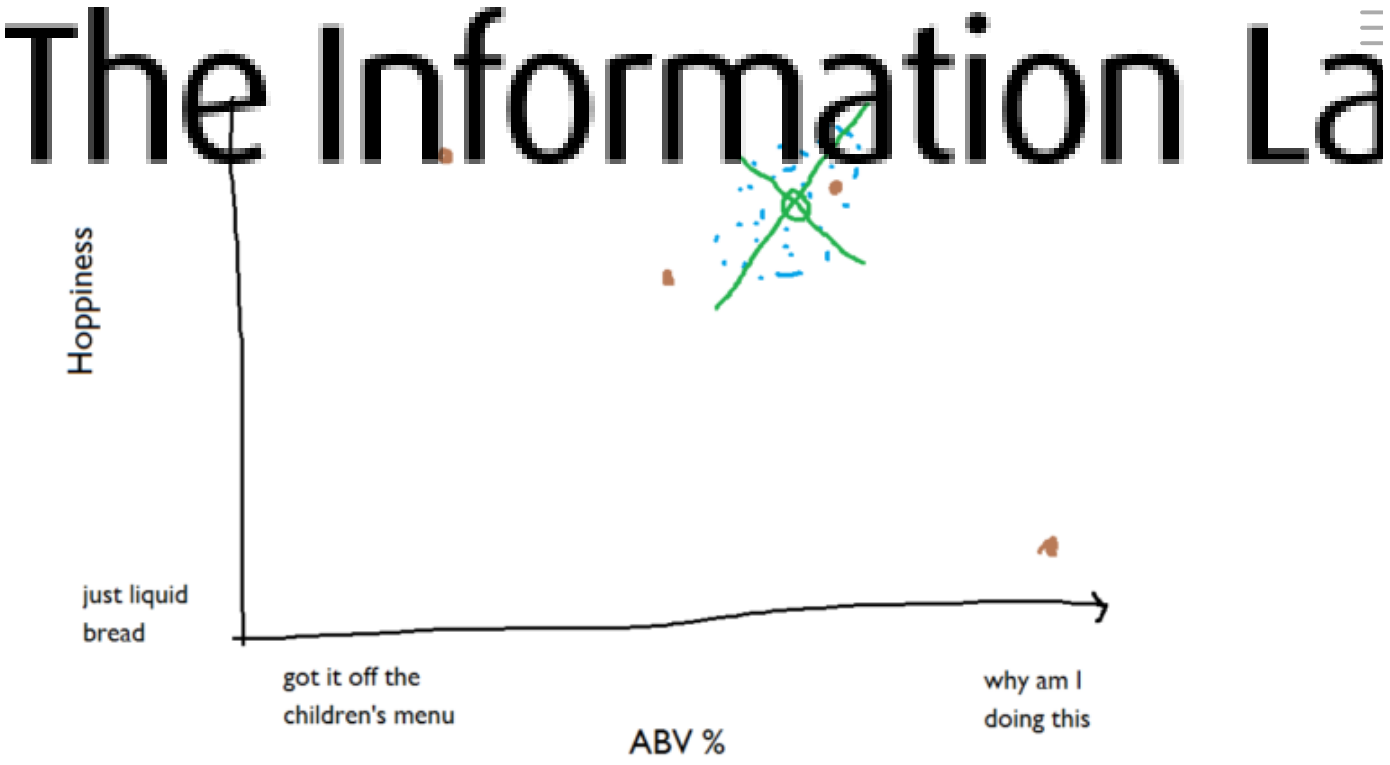
We're going to be working with these new axes, so let's disregard all the other beers for now:



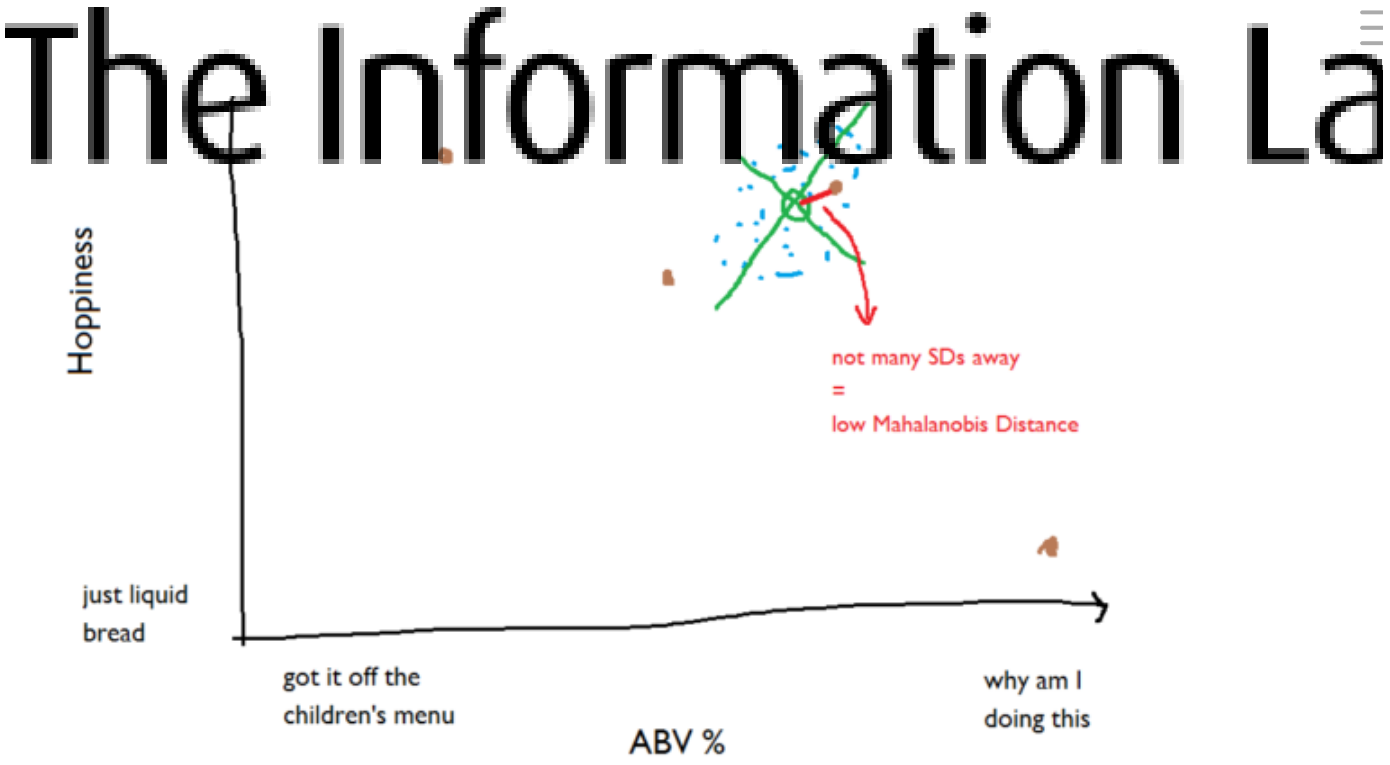
...and zoom in on this benchmark group of beers. We can put units of standard deviation along the new axes, and because 99.7% of normally distributed factors will fall within 3 standard deviations, that should cover pretty much the whole of the elliptical cloud of benchmark beers:



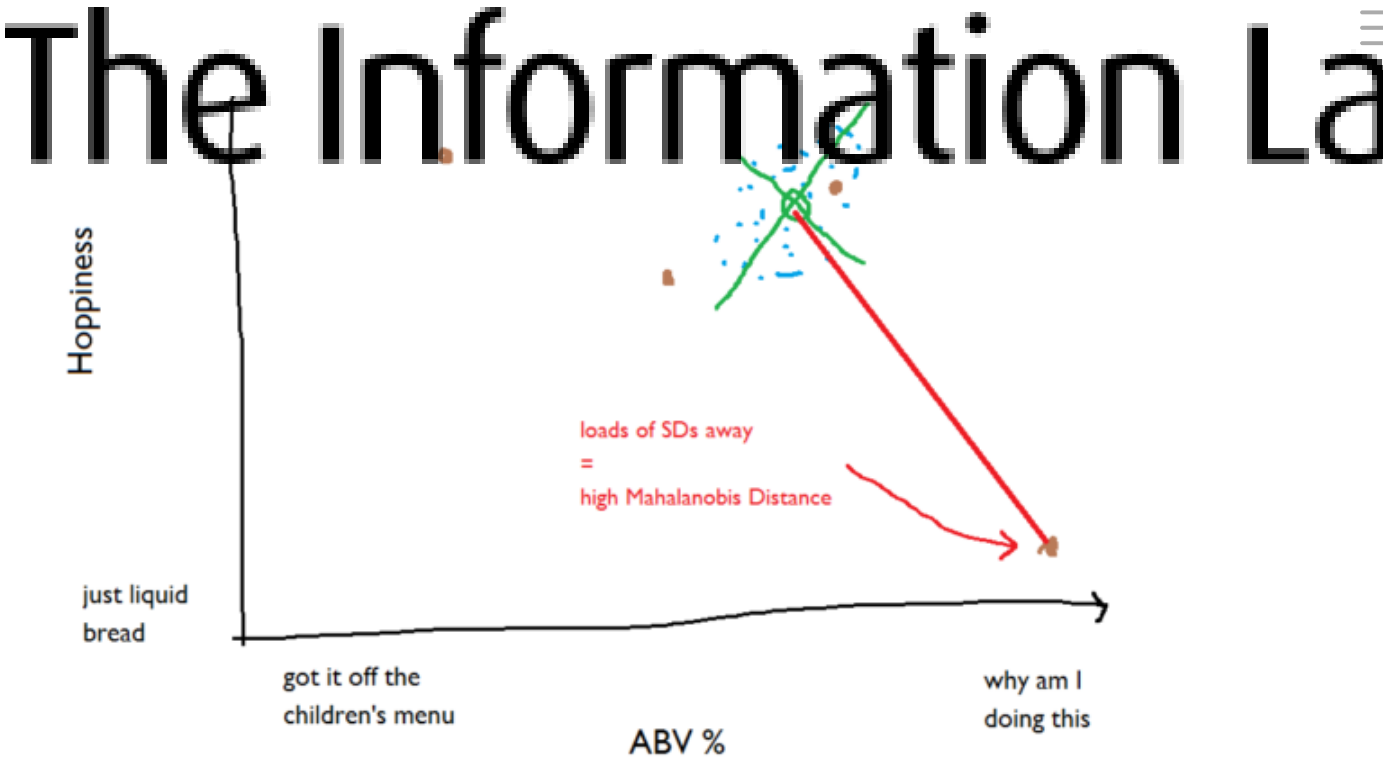
So, we've got the benchmark beers, we've found the centroid of them, and we can describe where the points sit in terms of standard deviations away from the centroid. Now, let's bring a few new beers in. You haven't tried these before, but you do know how hoppy and how strong they are:



The new beer inside the cloud of benchmark beers is pretty much in the middle of the cloud; it's only one standard deviation or so away from the centroid, so it has a low Mahalanobis Distance value:



The new beer that's really strong but not at all hoppy is a long way from the cloud of benchmark beers; it's several standard deviations away, so it has a high Mahalanobis Distance value:



This is just using two factors, strength and hoppiness; it can also be calculated with more than two factors, but that's a lot harder to illustrate in MS Paint.

The exact calculation of the Mahalanobis Distance involves matrix calculations and is a little complex to explain (see here for more mathematical details), but the general point is this:

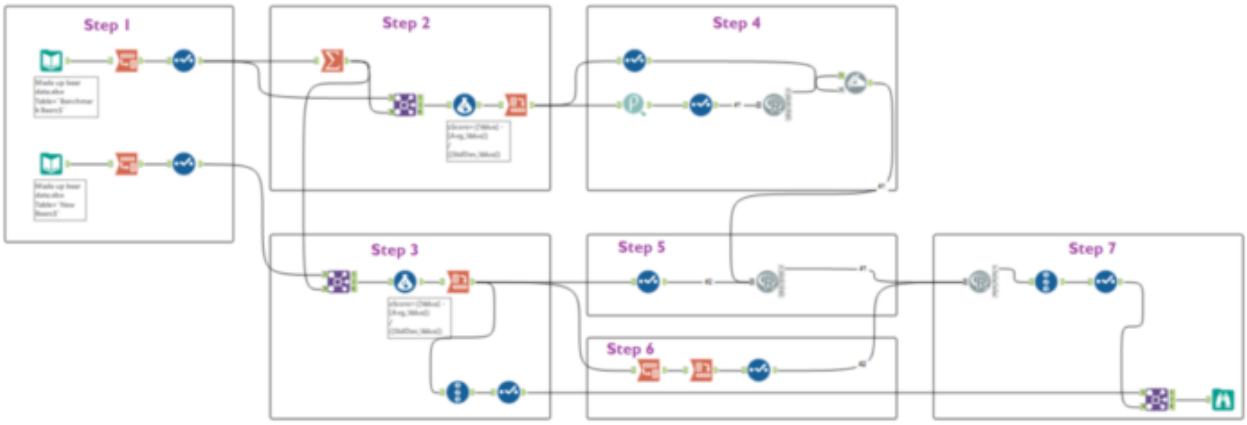
The lower the Mahalanobis Distance, the closer a point is to the set of benchmark points.

A Mahalanobis Distance of 1 or lower shows that the point is right among the benchmark points. This is going to be a good one. The higher it gets from there,

The Information La

Distance is and how to interpret it; the next stage is how to calculate it in Alteryx. This will involve the R tool and matrix calculations quite a lot; have a read up on the R tool and matrix calculations if these are new to you.

The overall workflow looks like this, and you can download it for yourself here (it was made with Alteryx 10.6):



...but that's pretty big, so let's break it down.

Step 1

Start with your beer dataset. Create one dataset of the benchmark beers that you know and love, with one row per beer and one column per factor (I've just

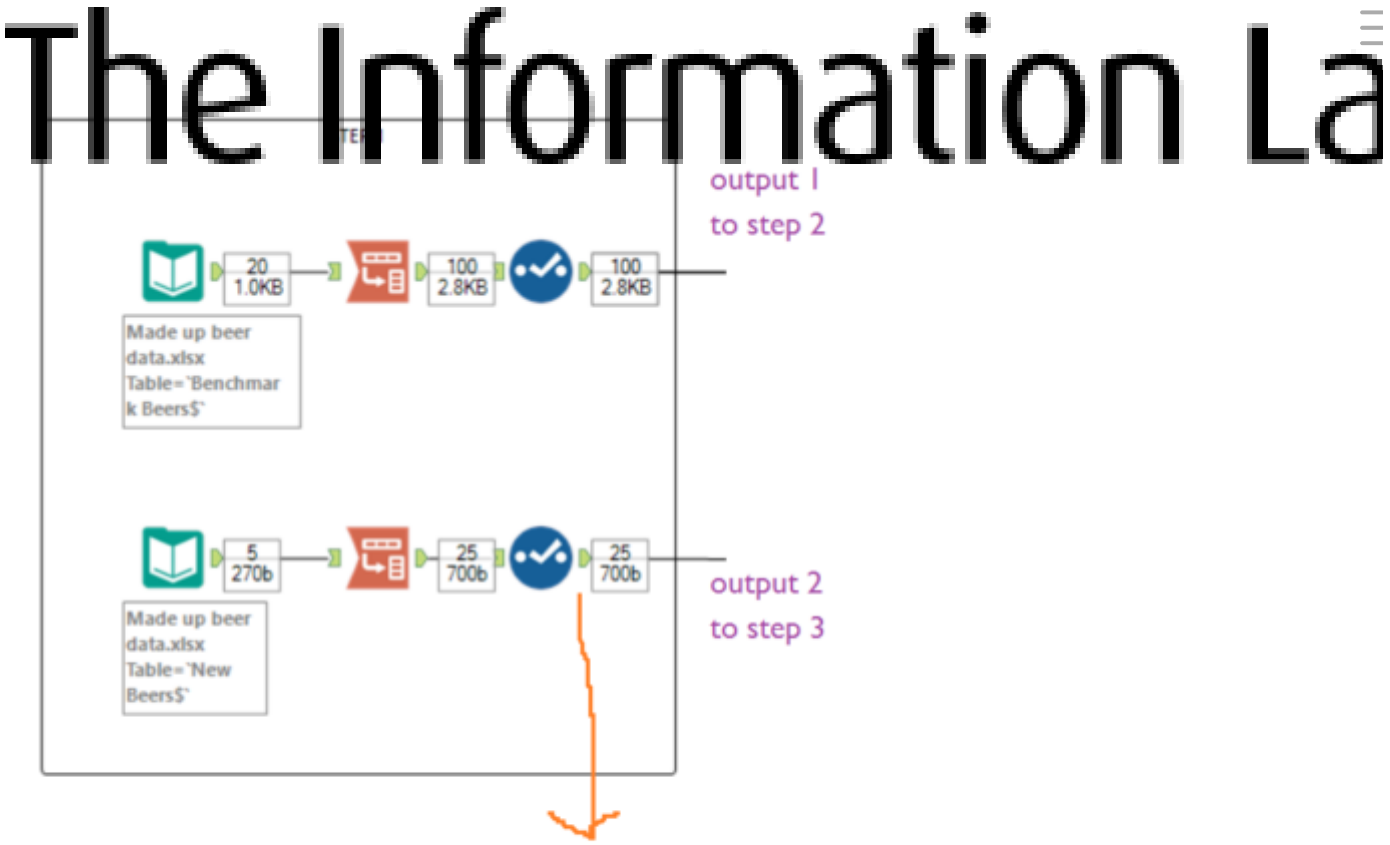
The Information La

	A	B	C	D	E	F
1	Beer	ABV	SRM	Hoppiness	IBU	Sourness
2	1	5.6	11	4	40	1
3	2	6.8	7	3	34	1
4	3	6.5	17	6	51	1
5	4	4.8	14	6	55	2
6	5	6.5	5	6	36	1
7	6	5.4	11	7	53	2

Note: you can't calculate the Mahalanobis Distance if there are more factors than records. Here, I've got 20 beers in my benchmark beer set, so I could look at up to 19 different factors together (but even then, that still won't work well). It's best to only use a lot of factors if you've got a lot of records.

Another note: you can only calculate the Mahalanobis Distance with continuous variables as your factors of interest, and it's best if these factors are normally distributed. So, beer strength will work, but beer country of origin won't (even if it's a good predictor that you know you like Belgian beers).

Now create an identically structured dataset of new beers that you haven't tried yet, and read both of those into Alteryx separately.



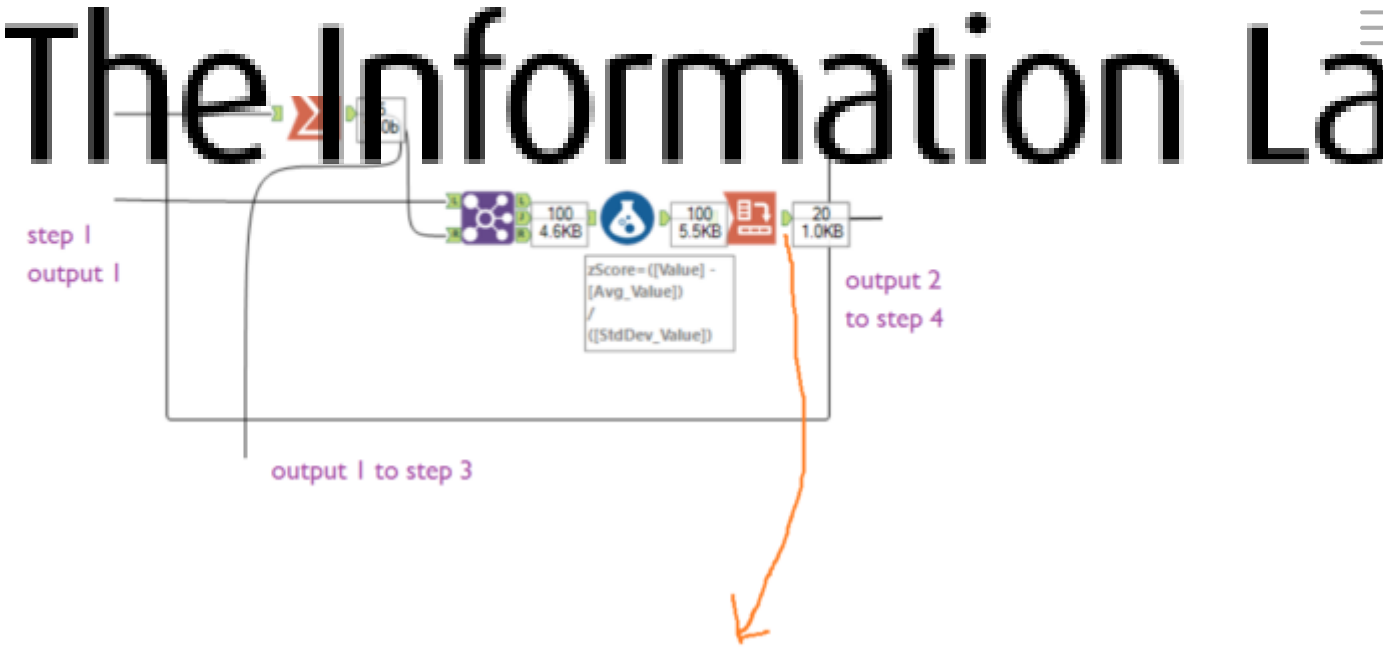
Results - Select (10) - Output

3 of 3 Fields | Cell Viewer | 25 records displayed

Record #	Beer	Factor	Value
1	21	ABV	2.3
2	21	SRM	3
3	21	Hoppiness	2
4	21	IBU	15
5	21	Sourness	10
6	22	ABV	4
7	22	SRM	7
8	22	Hoppiness	5
9	22	IBU	34
10	22	Sourness	1
11	23	ABV	8.5
12	23	SRM	19
13	23	Hoppiness	9

Step 2

Factor validity, the mean and standard deviation of the values, and join the output together with the benchmark beer data by joining on Factor. Now calculate the z scores for each beer and factor compared to the group summary statistics, and crosstab the output so that each beer has one row and each factor has a column. You should get a table of beers and z scores per factor:



Results - Cross Tab (16) - Output

6 of 6 Fields | Cell Viewer | 20 records displayed

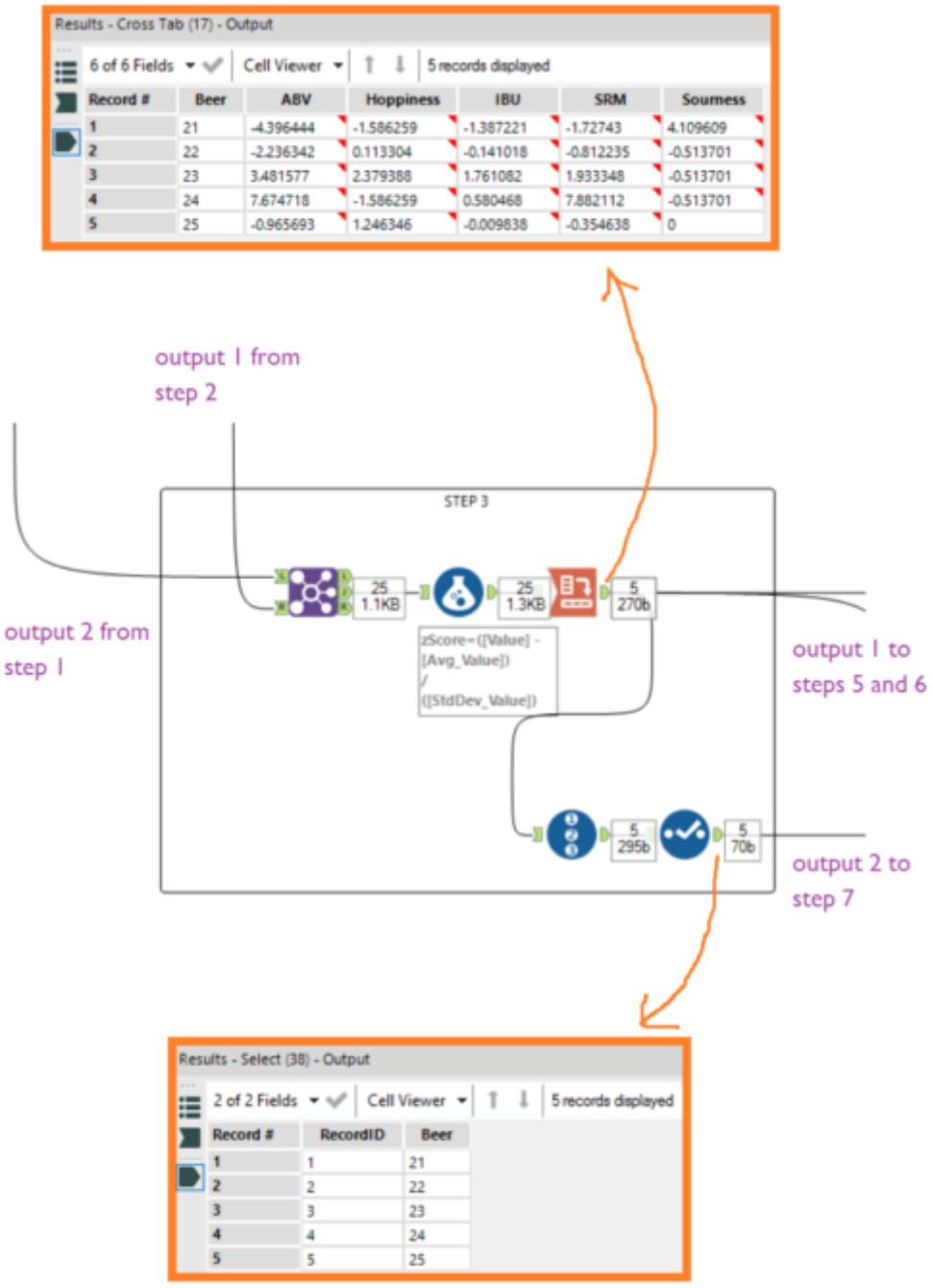
Record #	Beer	ABV	Hoppiness	IBU	SRM	Sourness
1	1	-0.203304	-0.453217	0.25252	0.102959	-0.513701
2	2	1.321475	-1.019738	-0.141018	-0.812235	-0.513701
3	3	0.94028	0.679825	0.974006	1.475751	-0.513701
4	4	-1.219823	0.679825	1.236365	0.789355	0
5	5	0.94028	0.679825	-0.009838	-1.269832	-0.513701
6	6	-0.457434	1.246346	1.105185	0.102959	0
7	7	-0.711563	1.812867	-1.715169	-0.354638	3.082207
8	8	1.448539	-1.019738	-1.190452	1.475751	-0.513701
9	9	-1.092758	0.113304	-0.796914	1.246952	0
10	10	-0.838628	1.246346	-1.5184	-0.812235	2.568506
11	11	0.432021	-1.019738	0.383699	-1.269832	-0.513701
12	12	0.559085	-1.019738	0.383699	1.246952	-0.513701
13	13	-1.092758	-0.453217	-1.124862	0.560557	-0.513701

Step 3

Now take your new beers, and join in the summary stats from the benchmark group. This time, we're calculating the z scores of the new beers, but in relation to the mean and standard deviation of the benchmark beer group, not the new beer group. Bring

The Information La

so that we can join on this later.

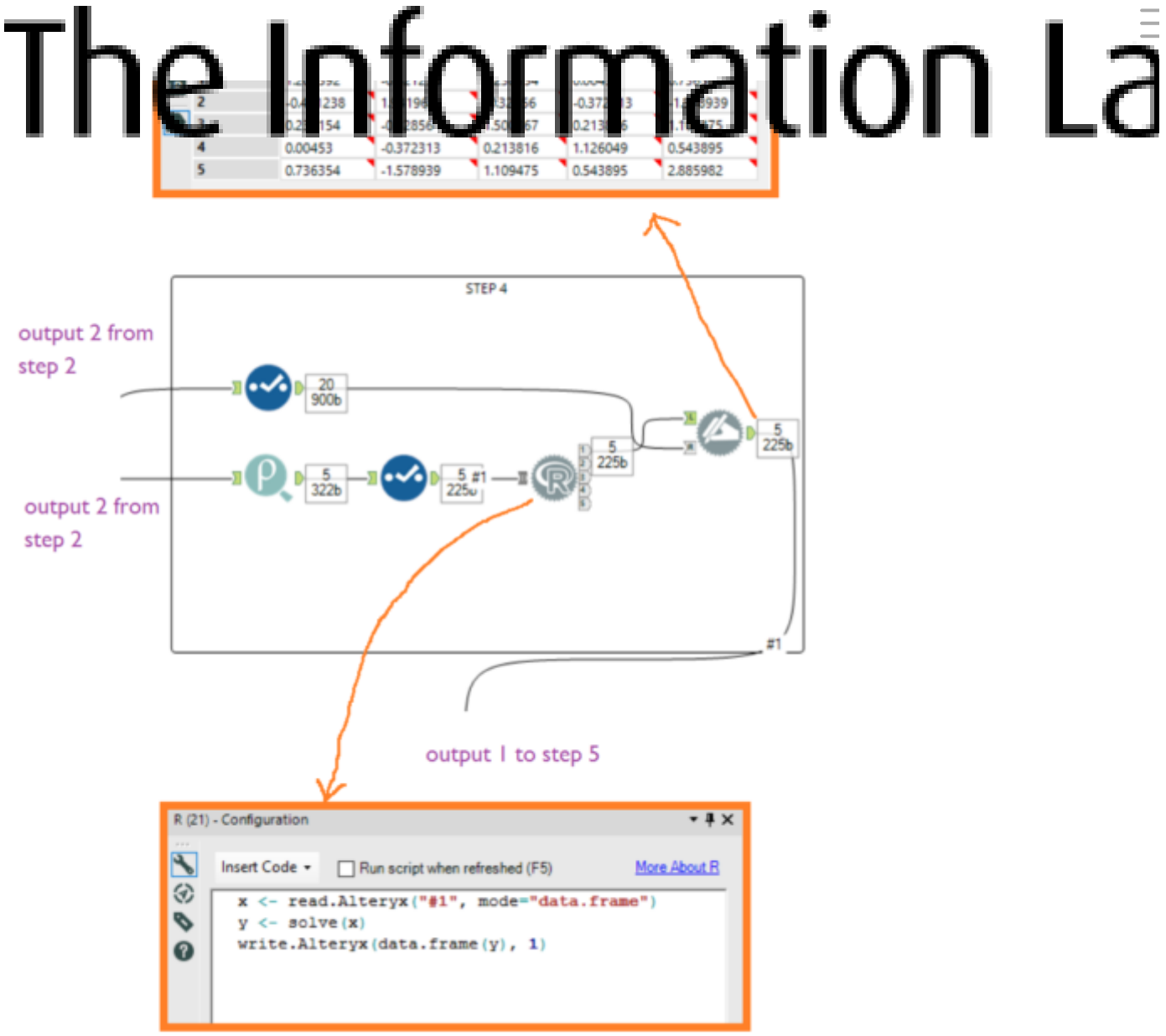


Step 4

correlations to obtain a table of correlations between the different factors. This will result in a table of correlations, and you need to remove Factor field so it can function as a matrix of values. Now read it into the R tool as in the code below:

```
x <- read.Alteryx("#1", mode="data.frame")
y <- solve(x)
write.Alteryx(data.frame(y), 1)
```

The solve function will convert the dataframe to a matrix, find the inverse of that matrix, and read results back out as a dataframe. This will remove the Factor headers, so you'll need to rename the fields by using a Dynamic Rename tool connected to the data from the earlier crosstab:



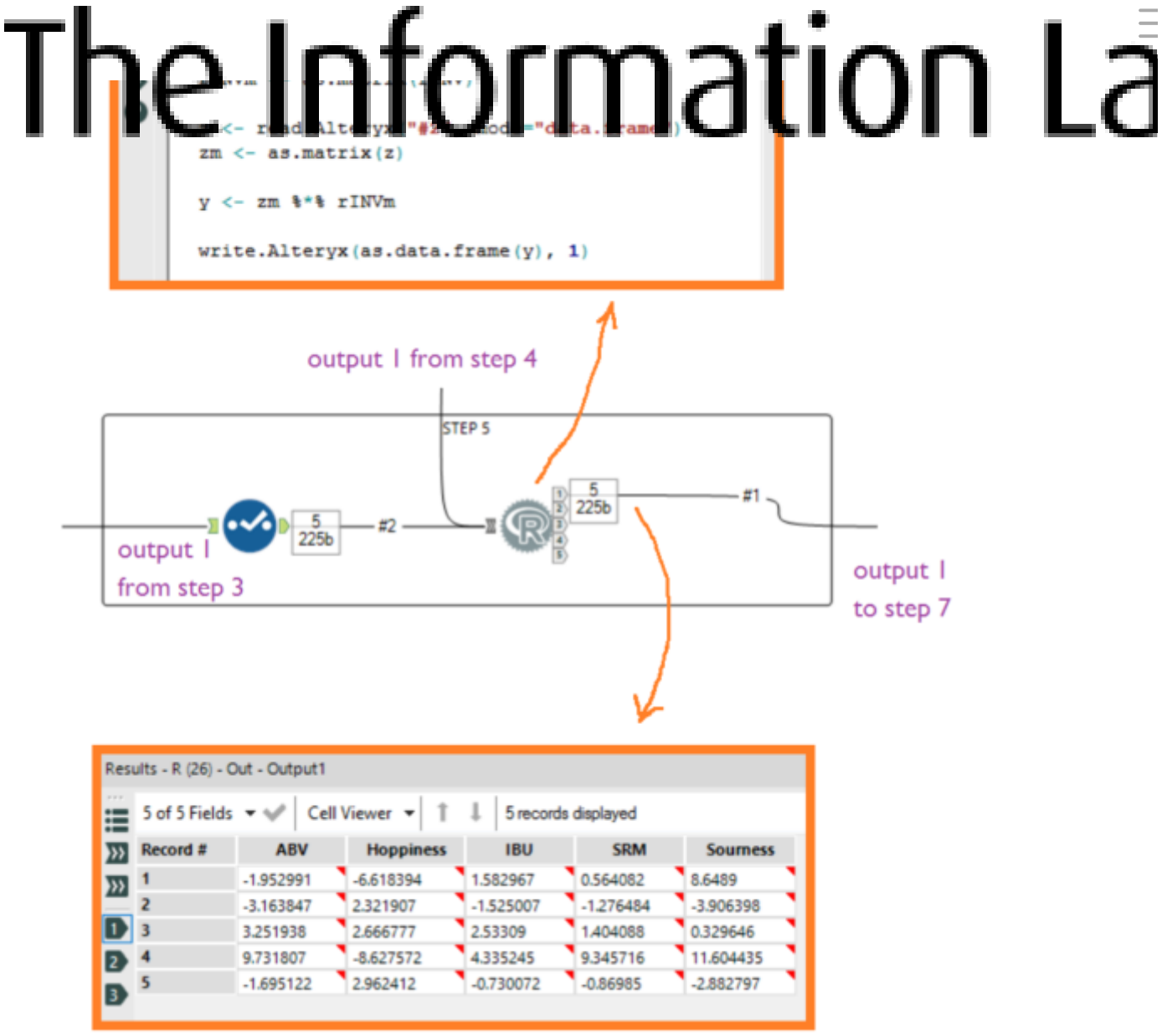
Step 5

If you liked the first matrix calculation, you'll love this one. Take the correlation matrix of factors for the benchmark beers (i.e. the output of step 4) and the z scores per factor for the new beer (i.e. output 1 of

The Information La

```
rINV <- read.Alteryx("#1",
mode="data.frame")
rINVm <- as.matrix(rINV)

z <- read.Alteryx("#2", mode="data.frame")
zm <- as.matrix(z)

y <- zm %*% rINVm

write.Alteryx(as.data.frame(y), 1)
```

This will convert the two inputs to matrices and multiply them together. Because this is matrix multiplication, it has to be specified in the correct order; it's the [z scores for new beers] x [correlation matrix], not the other way around. This will return a matrix of numbers where each row is a new beer and each column is a factor:



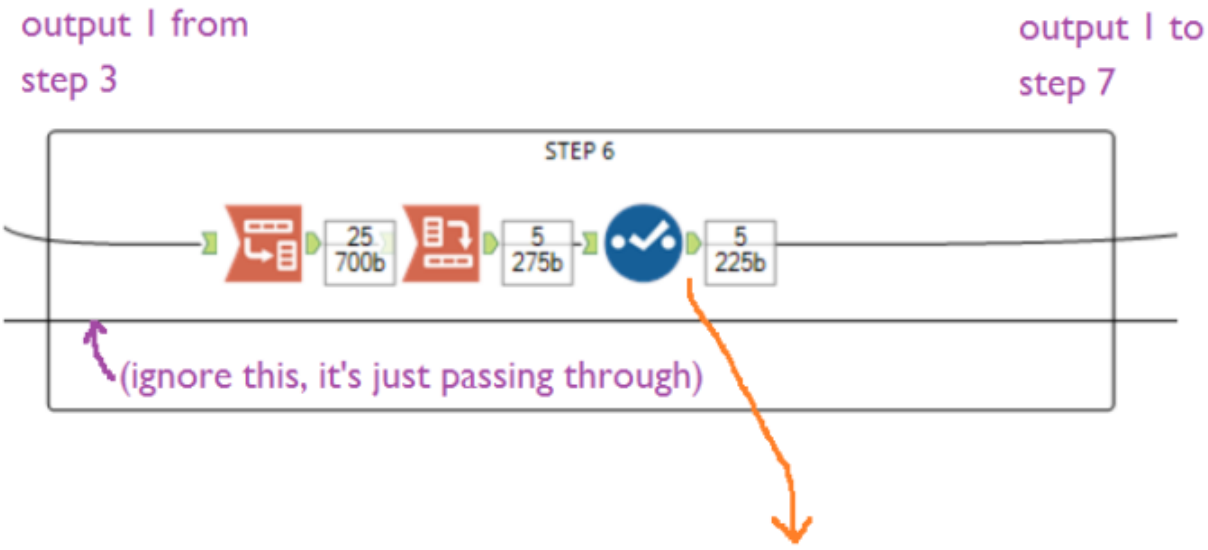
Record #	ABV	Hoppiness	IBU	SRM	Sourness
1	-1.952991	-6.618394	1.582967	0.564082	8.6489
2	-3.163847	2.321907	-1.525007	-1.276484	-3.906398
3	3.251938	2.666777	2.53309	1.404088	0.329646
4	9.731807	-8.627572	4.335245	9.345716	11.604435
5	-1.695122	2.962412	-0.730072	-0.86985	-2.882797

Step 6

Now take the z scores for the new beers again (i.e. output 1 from step 3). We need it to be in a matrix format where each column is each new beer, and each row is the z score for each factor. First transpose it with Beer as a key field, then crosstab it with name

The Information La

column with the factor names in it:



Results - Select (30) - Output

5 of 5 Fields | Cell Viewer | 5 records displayed

Record #	21	22	23	24	25
1	-4.396444	-2.236342	3.481577	7.674718	-0.965693
2	-1.586259	0.113304	2.379388	-1.586259	1.246346
3	-1.387221	-0.141018	1.761082	0.580468	-0.009838
4	-1.72743	-0.812235	1.933348	7.882112	-0.354638
5	4.109609	-0.513701	-0.513701	-0.513701	0

Step 7

...finally! We can calculate the Mahalanobis Distance. And if you thought matrix multiplication was fun, just wait til you see matrix multiplication in a for-loop.

beer (i.e. output 1 from step 6) as the second input.

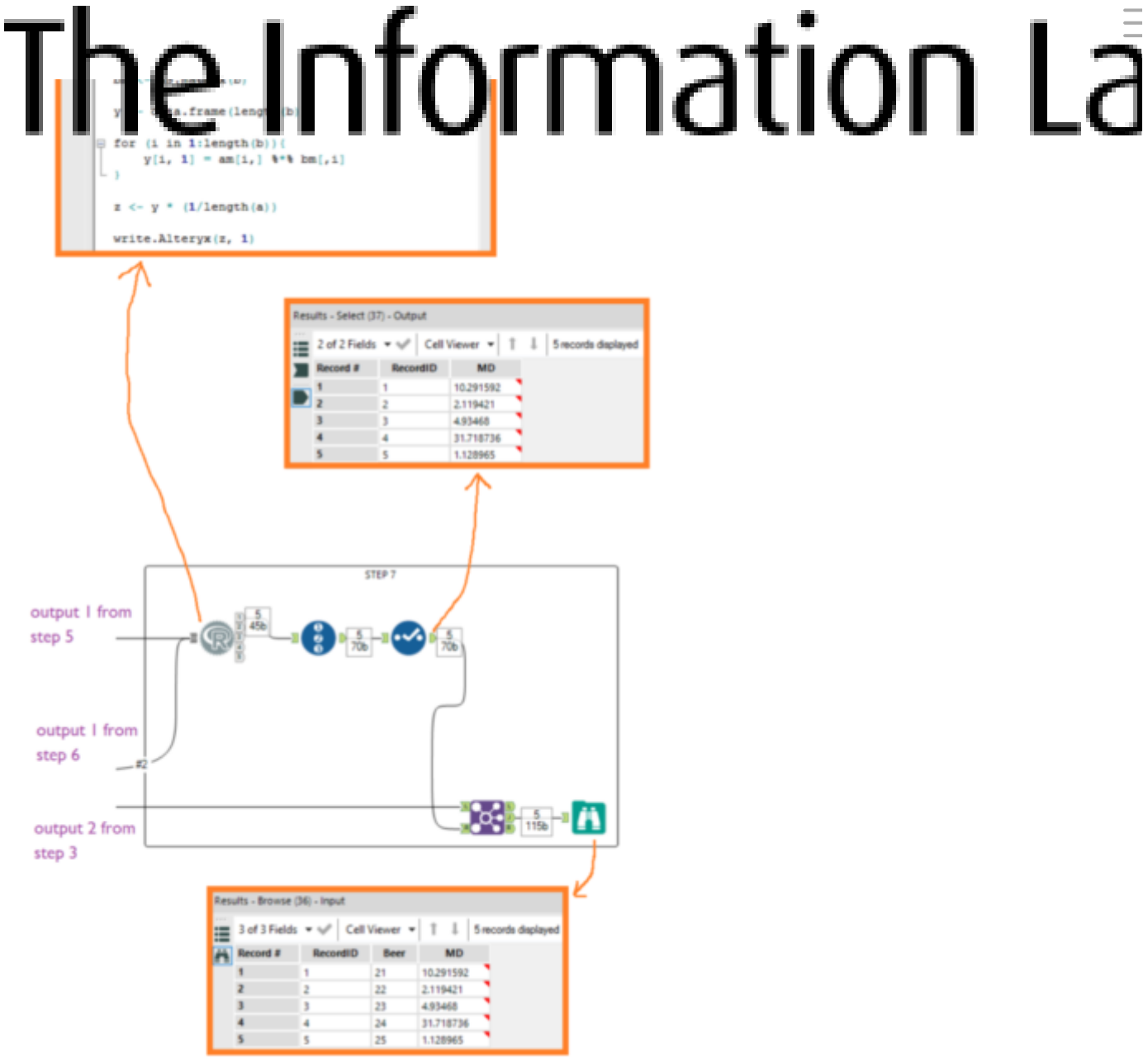
Now put in this code:

```
a <- read.Alteryx("#1", mode="data.frame")
am <- as.matrix(a)

b <- read.Alteryx("#2", mode="data.frame")
bm <- as.matrix(b)

y <- data.frame(length(b))

for (i in 1:length(b)){
y[i, 1] = am[i,] %*% bm[,i]
}

z <- y / length(a)

write.Alteryx(z, 1)
```

Each row in the first input (i.e. “a” in this code) is for the new beer, and each column in the second input (i.e. “b” in this code”) is for the new beer. What we need to do is to take the Nth row of the first input and multiply it by the corresponding Nth column of the second input. This means multiplying particular

[illegible] number of factors we're investigating. We could simply specify five here, but to make it more dynamic, you can use length(), which returns the number of columns in the first input.

This returns a simple dataframe where the column is the Mahalanobis Distance and each row is the new beer. But because we've lost the beer names, we need to join those back in from earlier. Remember how output 2 of step 3 has a Record ID tool? Well, put another Record ID tool on this simple Mahalanobis Distance dataframe, and join the two together based on Record ID. Alteryx will have ordered the new beers in the same way each time, so the positions will match across dataframes.



And there you have it! The Mahalanobis Distance for five new beers that you haven't tried yet, based on five factors from a set of twenty benchmark beers that you love.

quite many, you'd all like beer 22. The next lowest is 2.12 for beer 22, which is probably worth a try.

The highest Mahalanobis Distance is 31.72 for beer 24. If time is an issue, or if you have better beers to try, maybe forget about this one. The Mahalanobis Distance calculation has just saved you from beer you'll probably hate.

...but then again, beer is beer, and predictive models aren't infallible. Even with a high Mahalanobis Distance, you might as well drink it anyway. Cheers!

(the authors)

The Information La

Be a community star, share away!

5 thoughts on “Everything you ever wanted to know about the Mahalanobis Distance (and how to calculate it in Alteryx)”

Mike Seskin

27 May, 2017 at 4:04 pm

Gwilym and Beth are currently on their P1 placement with me at Solar Turbines, where they’re helping us link data to product quality improvements. One of the many ingredients in cooking up a solution to make this connection is the Mahalanobis distance, currently encoded in an Excel macro. I reluctantly asked them about the possibility of re-coding this in an Alteryx workflow, while thinking to myself, “I really shouldn’t be asking them to do this — it’s too difficult”. Clearly I was wrong, and also blown away by this outcome!! I definitely owe them a beer at Ballast Point Brewery, with a

There are plenty of multivariate distance metrics so why use this one? Why not for instance use a Cartesian distance? Because if we draw a circle around the “benchmark” beers it fails the capture the correlation between ABV% and Hoppiness. We would end up ordering a beer off the children’s menu and discover it tastes like a pine tree. An unfortunate but recoverable event. Much more consequential if the benchmark is based on for instance intensive care factors and we incorrectly classify a patient’s condition as normal because they’re in the circle but not in the ellipse.

REPLY

jonathan vendituoli

30 May, 2017 at 9:11 pm

Thank you for the creative statistics lesson. Learned something new about beer and Mahalanobis distance.

REPLY

The Information La

alphabetically but inconsistently – Cloud Data Architect

Laura

25 November, 2017 at 6:40 pm

This is the best explanation ever.

REPLY

John

21 April, 2018 at 10:54 pm

As someone who loves statistics, predictive analysis….and beer…..CHEERS! to this wonderful piece of work! Great write up!

REPLY

Leave a Reply

Your email address will not be published.Required fields are marked *

The Information La

Name

Email Address

POST COMMENT

14 March, 2019

Project Griffon: Admin Views

13 March, 2019

How not to Ask Data (aka, how

11 March, 2019

Importing and Exporting Non-

7 March, 2019

[redacted] 2019

How to – Tableau Javascript API for

27 February, 2019

How to show dynamic date

21 February, 2019

How to label the longest of a Dual

21 February, 2019

Consultant Diary: Nested LoD for Ranking

18 February, 2019

We Need to Talk About Density

15 February, 2019

Dynamic Date

1st Floor
25 Watling Street
London
EC4M 9BR

T: 08453 888 289
E: info@theinformationlab.co.uk