

Item Explorer: How to interactively explore combinatorial questions

Dr. Mihael Ankerst
Munich, Feb 2nd 2017

- **motivation**
- demo
- design choices
- how can you use the tool?

What kind of products do customers typically buy together in a grocery store?



# customers	fruit	beer	candy	magazines	...
6.388.860	1	0	0	0	
898.973	1	0	1	0	
4.231.452	0	1	0	0	
5.123.433	0	1	1	1	
...	

Sorting by frequency ...



# customers	fruit	beer	candy	magazines	...
6.567.680	1	1	0	0	
6.549.840	1	1	1	0	
6.488.320	1	0	1	0	
6.388.860	1	0	0	0	
...	

... or creating a pivot table



Zeilenbeschriftungen		Summe von _customers
=0		20089368
=0		9965592
+0		4823352
=1		5142240
=0		1267040
=0		632864
=0		315776
=0		157232
=0		77960
=0		77960
=0		46840
=0		23020
=0		23020
=0		23020
=0		8482
=1		8482
=0		8482
=0		1213
=0		1213
=0		256
	0	256
	=1	957
	0	957
=1		7269
=0		7269
=0		3252
	0	3252
	=1	4017
	0	4017
=1		14538
=1		14538
=0		14538
=0		4209
=0		4209
=0		1722

... or mining association rules
doesn't give you the full picture!



```

100 (f.meat) => (f.fat) 0.2389923 0.3021804 0.7868422
107 (b.wat) => (f.fat) 0.2767582 0.5461100 1.0002072
108 (f.fat) => (b.wat) 0.2767582 0.5114472 1.0002072
109 (b.wat) => (f.veg) 0.3124586 0.6165395 0.9872194
110 (f.veg) => (b.wat) 0.3124586 0.5805842 0.9872194
111 (b.wat) => (f.fru) 0.3128880 0.6152919 1.0005244
112 (f.fru) => (b.wat) 0.3128889 0.5102953 1.0009344
113 (b.jui) => (f.meat) 0.2635273 0.5106947 0.9897580
114 (f.meat) => (b.jui) 0.2635273 0.5107323 0.9897580
115 (b.jui) => (f.fat) 0.2811916 0.5440267 1.0070205
116 (f.fat) => (b.jui) 0.2811916 0.5106400 1.0070205
117 (b.jui) => (f.veg) 0.3157756 0.6119476 0.9798669
118 (f.veg) => (b.jui) 0.3157756 0.5856283 0.9798669
119 (b.jui) => (f.fru) 0.3165283 0.6133988 1.0029593
120 (f.fru) => (b.jui) 0.3165283 0.5175444 1.0029593
121 (f.meat) => (f.fat) 0.2888593 0.5558271 1.0345563
122 (f.fat) => (f.meat) 0.2888593 0.5338097 1.0345563
123 (f.meat) => (f.veg) 0.3048155 0.5907513 0.9459268
124 (f.veg) => (f.meat) 0.3048155 0.4880787 0.9459268
125 (f.meat) => (f.fru) 0.3241218 0.6281681 1.0271218
126 (f.fru) => (f.meat) 0.3241218 0.5299737 1.0271218
127 (f.fat) => (f.veg) 0.3005035 0.5604161 0.9809085
128 (f.veg) => (f.fat) 0.3005035 0.4907815 0.9809085
129 (f.fat) => (f.fru) 0.3417171 0.6314986 1.0325545
130 (f.fru) => (f.fat) 0.3417171 0.5507439 1.0325545
131 (f.veg) => (f.fru) 0.3299639 0.5283470 0.8639036
132 (f.fru) => (f.veg) 0.3299639 0.5395261 0.8639036
133 (f.can.f.can) => (f.veg) 0.1578878 0.6250926 1.0022920
134 (f.can.f.veg) => (f.can) 0.1578878 0.5025113 0.9993173
135 (f.can.f.veg) => (f.can) 0.1578878 0.5824255 0.9994376
136 (f.can.f.can) => (f.fru) 0.1546607 0.6131823 1.0020184
137 (f.can.f.fru) => (f.can) 0.1546607 0.5826169 0.9995273
138 (f.can.f.fru) => (f.can) 0.1546607 0.5020038 0.9995914
139 (f.egg.f.can) => (f.veg) 0.1579408 0.6258022 1.0020511
140 (f.can.f.veg) => (f.egg) 0.1579408 0.5826881 0.9998752
141 (f.egg.f.veg) => (f.can) 0.1579408 0.5024257 0.9994360
142 (f.egg.f.can) => (f.fru) 0.1547359 0.6131035 1.0024896
143 (f.can.f.fru) => (f.egg) 0.1547359 0.5828420 0.9993970
144 (f.egg.f.fru) => (f.can) 0.1547359 0.5025027 0.9995891
145 (b.mil.f.can) => (f.veg) 0.1574137 0.6236467 0.9985998
146 (f.can.f.veg) => (b.mil) 0.1574137 0.5810825 0.9952151
147 (b.mil.f.veg) => (f.can) 0.1574137 0.5039046 0.9887774
148 (b.mil.f.can) => (f.fru) 0.1551516 0.6146844 1.0058746
149 (f.can.f.fru) => (b.mil) 0.1551516 0.5819227 1.0015523
150 (b.mil.f.fru) => (f.can) 0.1551516 0.5024959 0.9952757
151 (b.win.f.can) => (f.veg) 0.1580469 0.6255017 1.0015701
152 (f.can.f.veg) => (b.win) 0.1580469 0.5810177 0.9989310
153 (b.win.f.veg) => (f.can) 0.1580469 0.5024241 0.9994327
154 (b.win.f.can) => (f.fru) 0.1548745 0.6120462 1.0022324
155 (f.can.f.fru) => (b.win) 0.1548745 0.5812923 0.9991369
156 (b.win.f.fru) => (f.can) 0.1548745 0.5025884 0.9995846
157 (b.bee.f.can) => (f.veg) 0.1578489 0.6224635 0.9967051
158 (f.can.f.veg) => (b.bee) 0.1578489 0.5823630 0.9943523
159 (b.bee.f.veg) => (f.can) 0.1578489 0.5020889 0.9987669
160 (b.bee.f.can) => (f.fru) 0.1557057 0.6140432 1.0040262
161 (f.can.f.fru) => (b.bee) 0.1557057 0.5819916 1.0000370
162 (b.bee.f.fru) => (f.can) 0.1557057 0.5024870 0.9995580
163 (b.wat.f.can) => (f.veg) 0.1569873 0.6137939 0.9888087
164 (f.can.f.veg) => (b.wat) 0.1569873 0.4990452 0.9859191
165 (b.wat.f.veg) => (f.can) 0.1569873 0.5024388 0.9994520

```

Output of arules package in R Studio

The idea of item explorer was born

- D3.js



- Use bar charts to represent item frequencies!

- motivation
- **demo**
- design choices
- how can you use the tool?

Demo: item explorer

Info

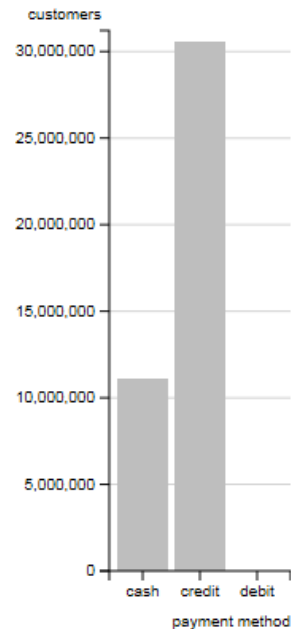
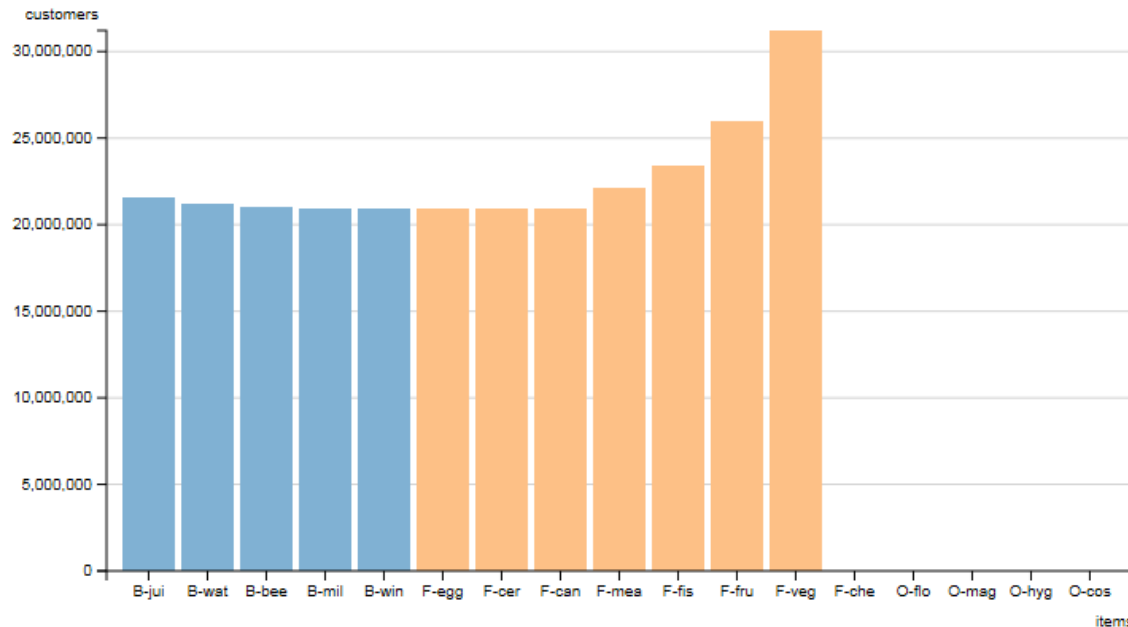
current filter:

frequency: 41,636,931 percent: 100%

Exploration

item 1	item 2	frequency	percent
F-veg	credit	24,307,588	58.4%
F-fru	credit	19,251,390	46.2%
F-fru	F-veg	18,205,890	43.7%

☐ sort by frequency ☐ update axis [help](#)



- motivation
- demo
- **design choices**
- how can you use the tool?

Design choices: #1 – virtual area for tooltip

Info

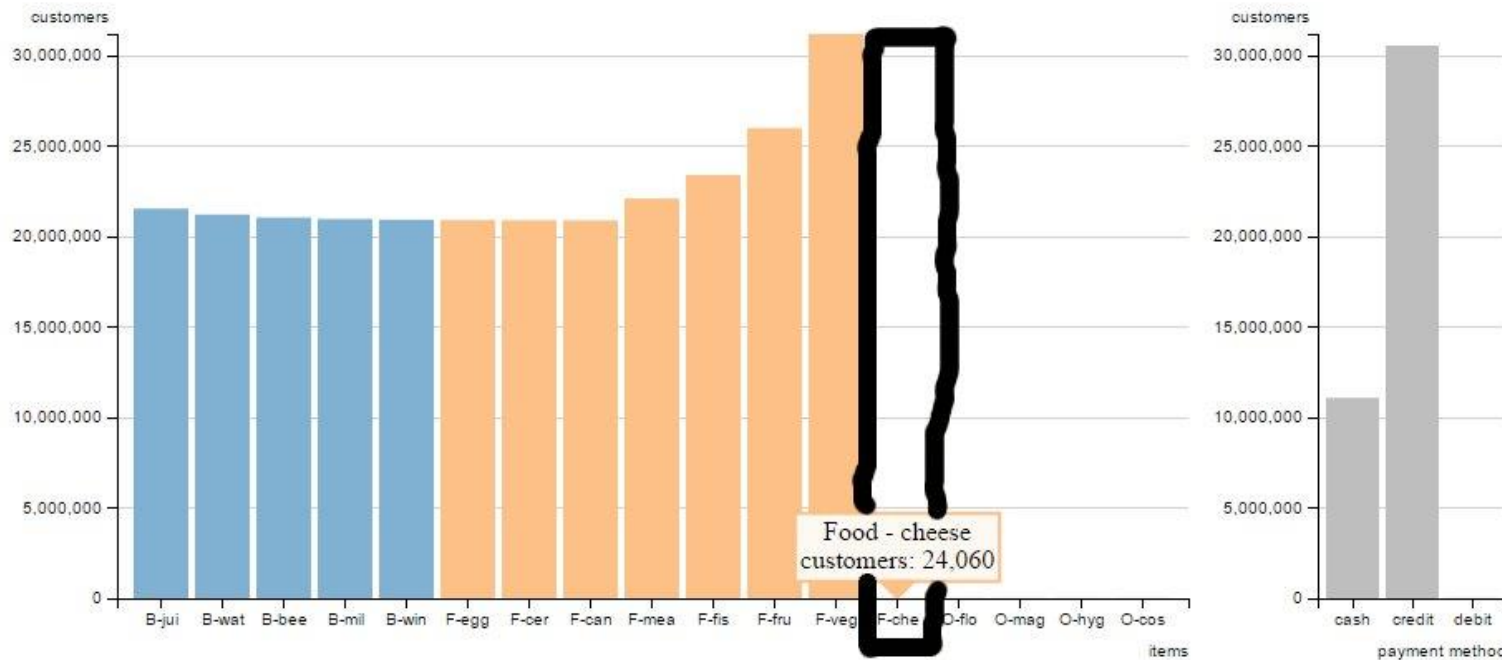
current filter:

frequency: 41,641,078 percent: 100%

Exploration

item 1	item 2	frequency	percent
F-veg	credit	24,307,588	58.4%
F-fru	credit	19,251,390	46.2%
F-fru	F-veg	18,205,890	43.7%

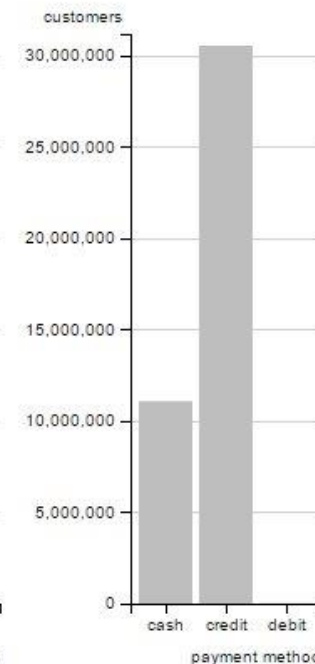
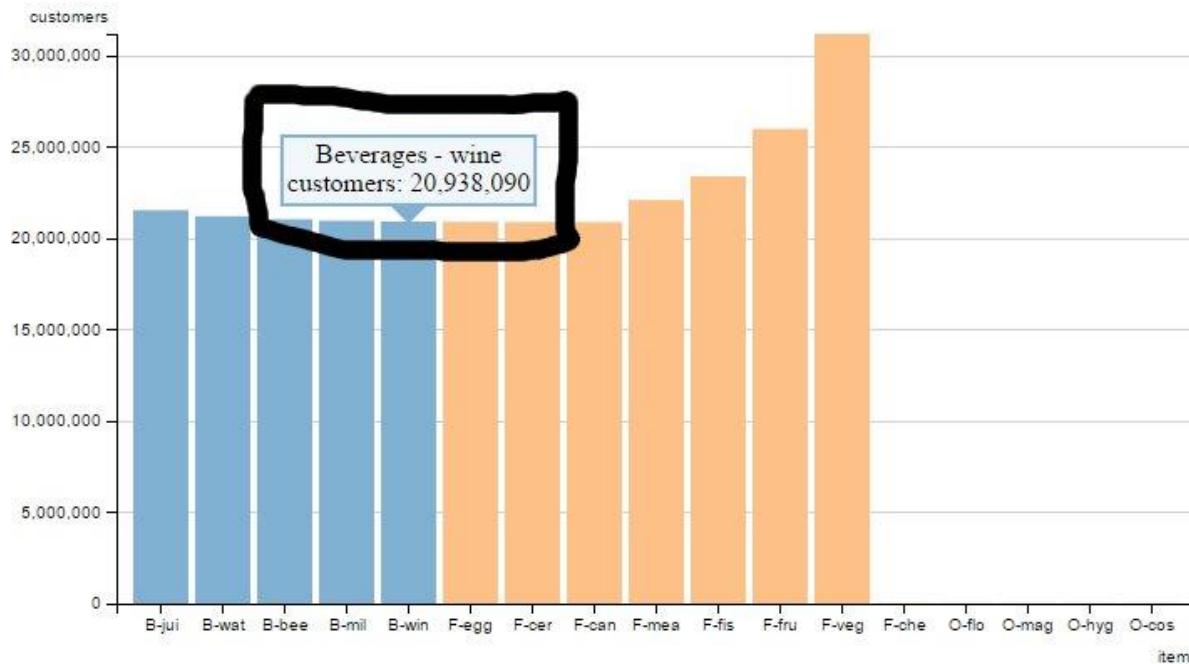
☐ sort by frequency ☐ update axis [help](#)



Design choices:

#2 – descriptive name of item in the tooltip

Info		Exploration			
current filter:		item 1	item 2	frequency	percent
		F-veg	credit	24,307,588	58.4%
		F-fru	credit	19,251,390	46.2%
		F-fru	F-veg	18,205,890	43.7%
frequency: 41,641,078		percent: 100%		<input type="checkbox"/> sort by frequency <input type="checkbox"/> update axis <button>help</button>	



Design choices: #3 – separate chart area from axis labels

Info

current filter:

frequency: 41,641,078

percent: 100%

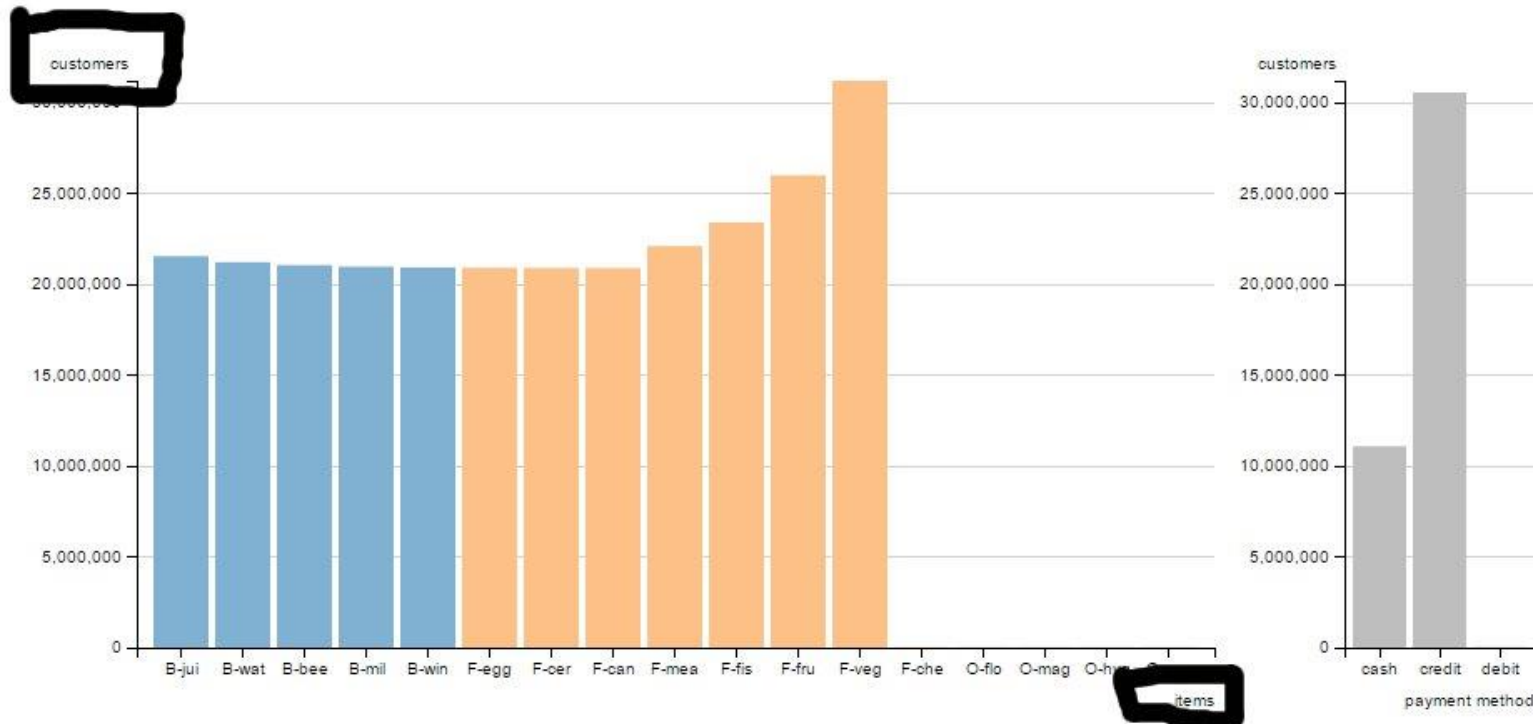
Exploration

item 1	item 2	frequency	percent
F-veg	credit	24,307,588	58.4%
F-fru	credit	19,251,390	46.2%
F-fru	F-veg	18,205,890	43.7%

☐ sort by frequency

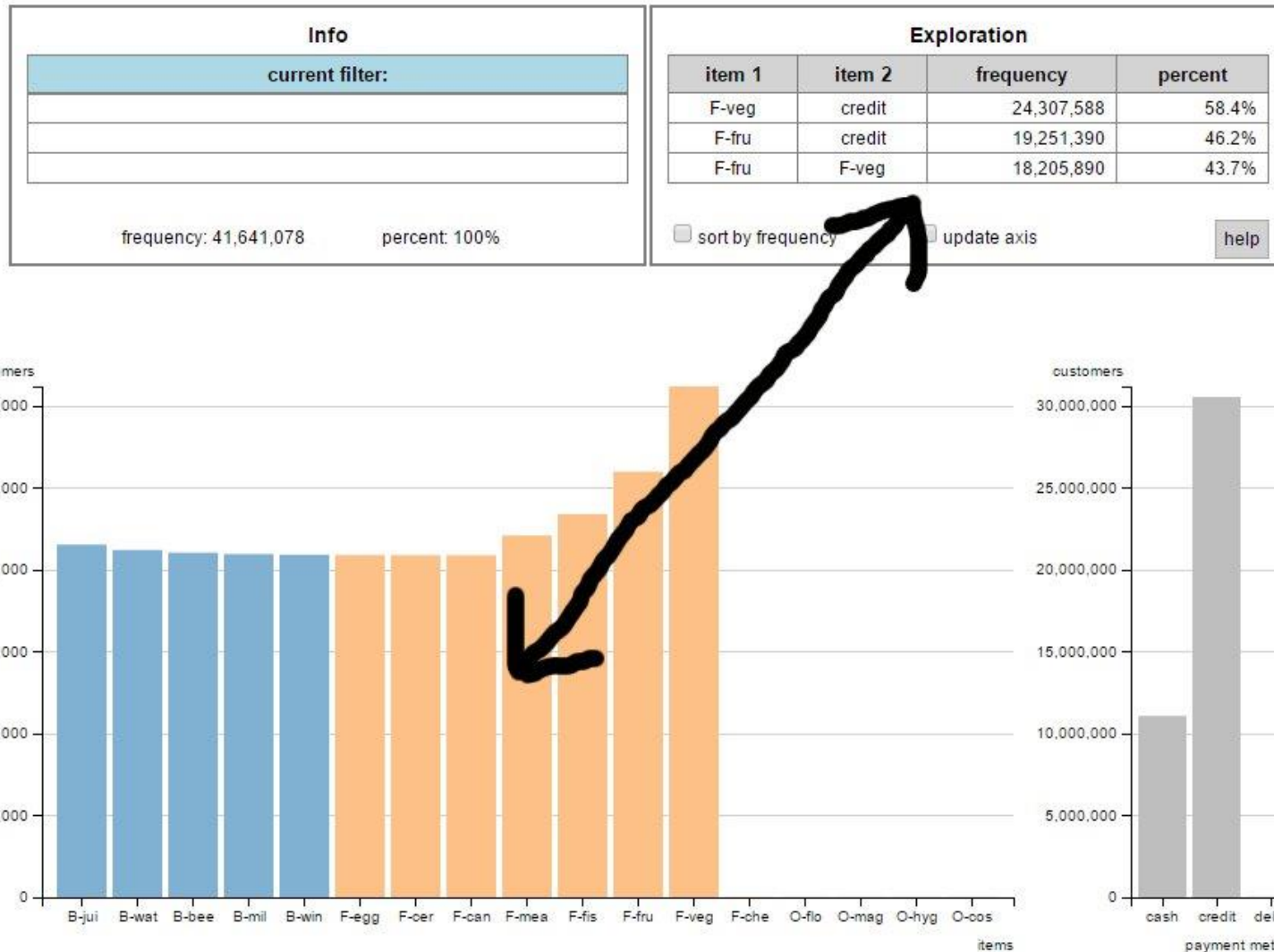
☐ update axis

help



Design choices:

#4 – combine interactive exploration with algorithmic guidance



Design choices: #5 – resorting with staggered delay

Info

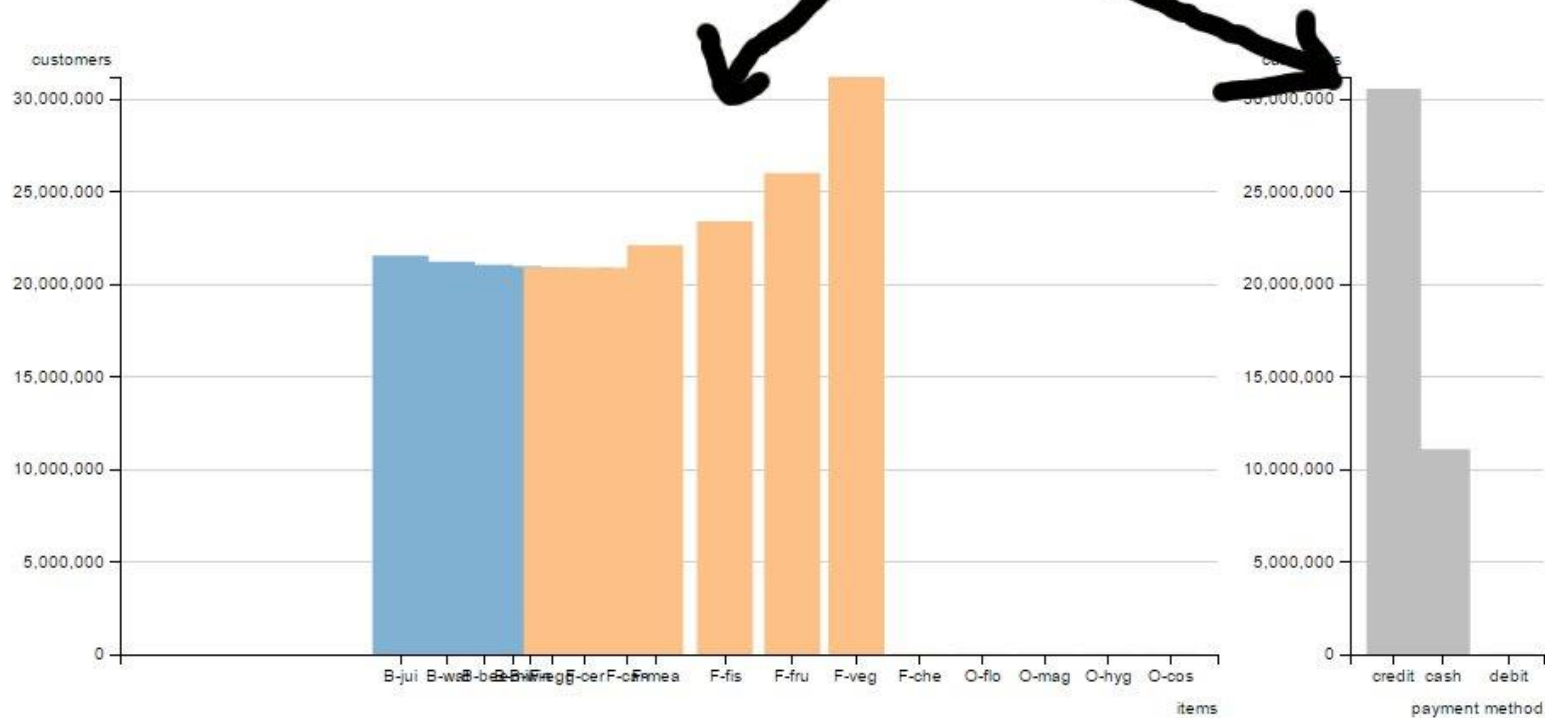
current filter:

frequency: 41,641,078 percent: 100%

Exploration

item 1	item 2	frequency	percent
F-veg	credit	24,307,588	58.4%
F-fru	credit	19,251,390	46.2%
F-fru	F-veg	18,205,890	43.7%

☒ sort by frequency ☐ update axis [help](#)



Design choices: #6 – displaying OR selections

Info

current filter:

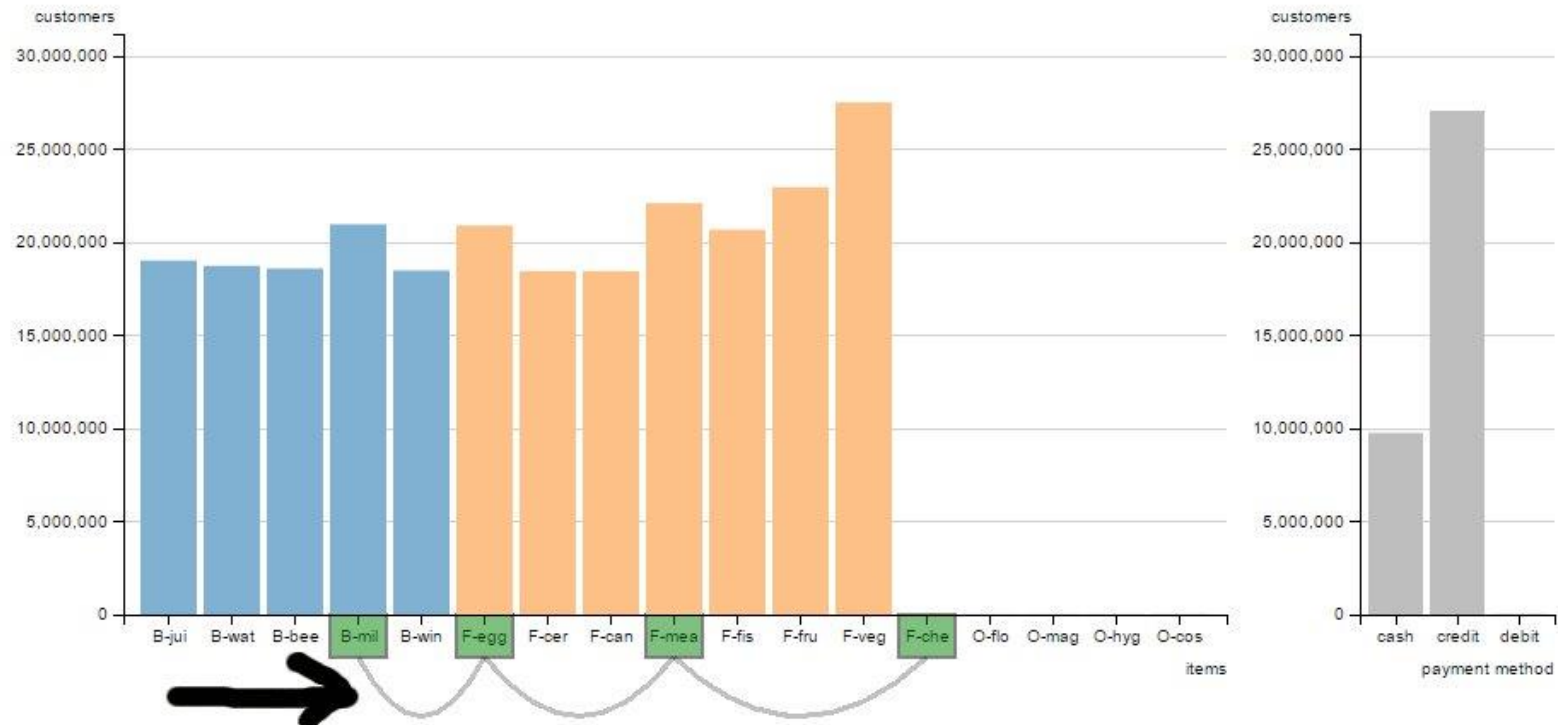
(+Beverages - milk || +Food - eggs || +Food - meat || +Food - cheese)

frequency: 36,837,974 percent: 88.5%

Exploration

item 1	item 2	frequency	percent
F-veg	credit	21,495,868	51.6%
F-fru	credit	16,993,400	40.8%
F-fru	F-veg	16,042,240	38.5%

☐ sort by frequency
 ☐ update axis
 [help](#)



Design choices: #7 – displaying additional structure

Info

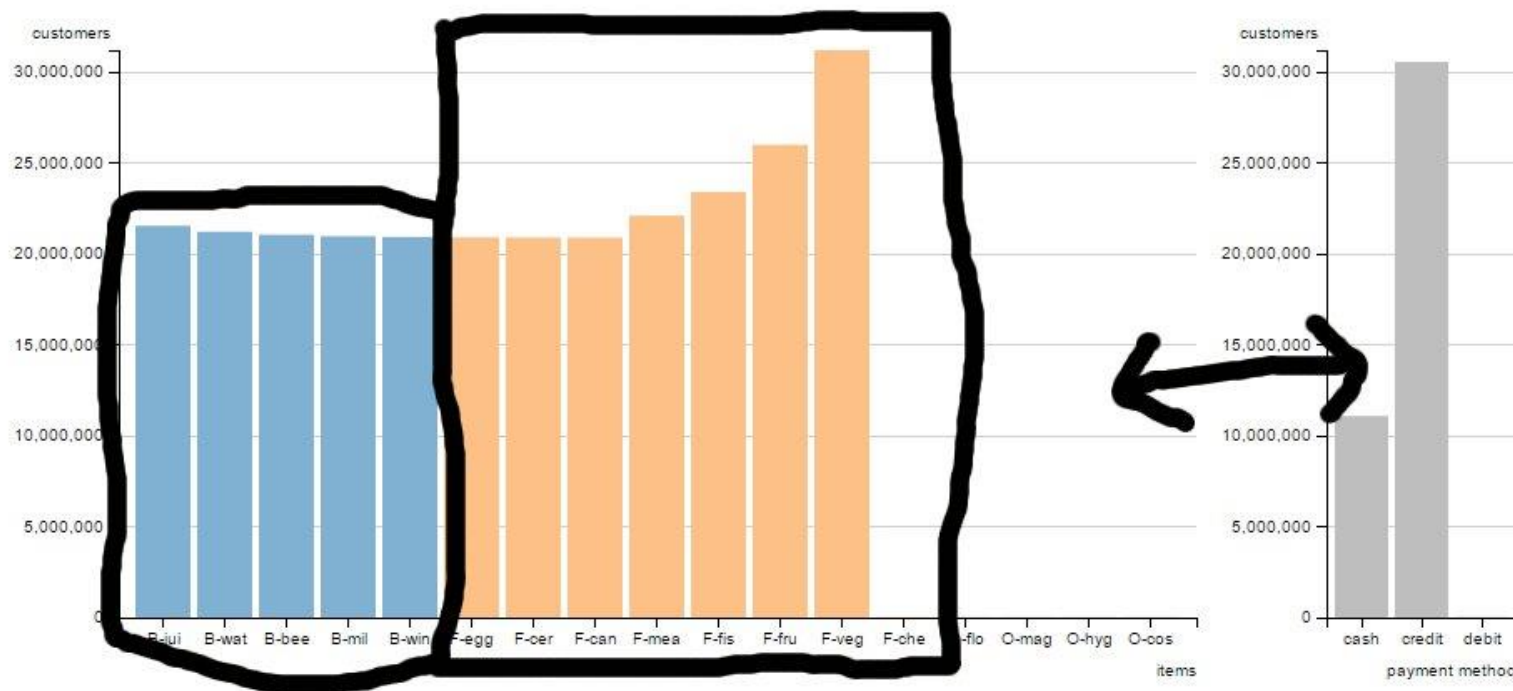
current filter:

frequency: 41,641,078 percent: 100%

Exploration

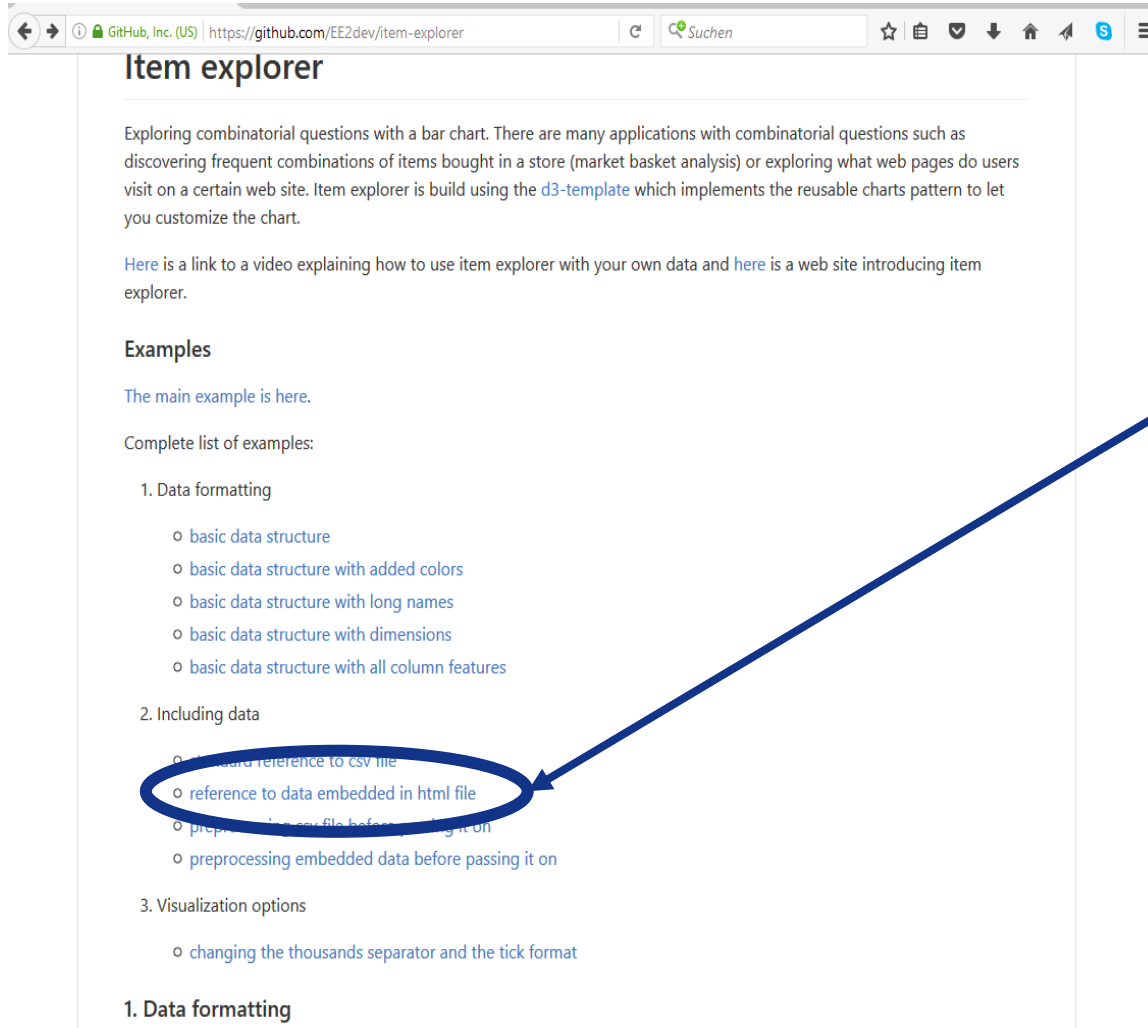
item 1	item 2	frequency	percent
F-veg	credit	24,307,588	58.4%
F-fru	credit	19,251,390	46.2%
F-fru	F-veg	18,205,890	43.7%

☐ sort by frequency
 ☐ update axis
 [help](#)



- motivation
- demo
- design choices
- **how can you use the tool?**

How can I use it?



The screenshot shows a web browser window with the URL `https://github.com/EE2dev/item-explorer`. The page title is "Item explorer". The main text describes the tool as a bar chart for exploring combinatorial questions. It includes links to a video and a website. Under the "Examples" section, there is a list of examples. A blue arrow points from the text "Click on the link: „reference to data embedded in html file“" to the link "reference to data embedded in html file" in the list.

Item explorer

Exploring combinatorial questions with a bar chart. There are many applications with combinatorial questions such as discovering frequent combinations of items bought in a store (market basket analysis) or exploring what web pages do users visit on a certain web site. Item explorer is build using the [d3-template](#) which implements the reusable charts pattern to let you customize the chart.

[Here](#) is a link to a video explaining how to use item explorer with your own data and [here](#) is a web site introducing item explorer.

Examples

The main example is [here](#).

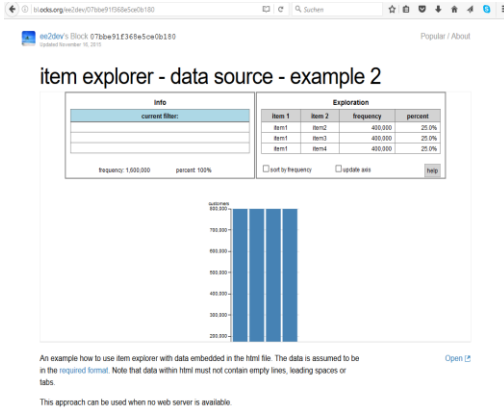
Complete list of examples:

1. Data formatting
 - [basic data structure](#)
 - [basic data structure with added colors](#)
 - [basic data structure with long names](#)
 - [basic data structure with dimensions](#)
 - [basic data structure with all column features](#)
2. Including data
 - [reference to csv file](#)
 - [reference to data embedded in html file](#)
 - [preprocessing csv file before passing it on](#)
 - [preprocessing embedded data before passing it on](#)
3. Visualization options
 - [changing the thousands separator and the tick format](#)

1. Data formatting

- 1) Go to the github page
- 2) Click on the link:
„reference to data embedded
in html file“

How can I use it?



-
-
-

3) Scroll down to index.html

4) Copy the code and paste it into an editor. Save file as „index.html“

5) Change data in `<pre></pre>` tag to your data

6) Save file as „index.html“

7) Open file in browser

For data formatting or advanced visualization options, see docu on github site.

[illegible]

<http://www.ankerst.de/Mihael/proj/mbc/>

and

<https://github.com/EE2dev/item-explorer>