Data Processing Dashboard

Mats Willemsen

December 2015

1 Introductie

Het doel van dit dashboard is de totale handel tussen landen te kunnen visualiseren tussen de landen van de wereld. Hiervoor is een dataset gebruikt van de afdeling van *Political Sciences* van de *University of California*. In deze dataset worden een groot aantal variabelen weergegeven. In de visualisaties worden alleen de import/export-cijfers gebruikt, de jaartal, en de land-code.

De visualisatie gebruikt vervolgens een tweetal technieken om de data visueel te maken. Allereerst wordt er op basis van exportcijfers een groepering gemaakt om de kaart te kleuren. Des te donkerder de kleur, des te groter de export van het land. Er is bewust gekozen om hier geen absolute waardes weer te geven, omdat twee andere grafieken dit doel hebben. In het begin geven de twee grafieken de totale handel weer over de tijd, verdeeld in twee lijnen voor import/export.

Als er vervolgens op een land wordt geklikt, wordt de data gefilterd op dat land, en worden de grafieken dynamisch aangepast aan de nieuwe data. Uiteindelijk worden hier wel de absolute waardes getoond, over de tijd heen van de dataset.

2 Technische details

Het was technisch gezien heel lastig om meer dan 500,000 rows snel te analyseren, te filteren, en te verwerken. Om dit mogelijk te maken wordt er gebruik gemaakt van een library genaamd *crossfilter*. Crossfilter maakt het mogelijk om MapReduce-achtige berekeningen uit te voeren op je data, je data te groeperen, en deze uiteindelijk ook te kunnen filteren, met hele hoge performance. De hoge complexiteit die dit met zich mee bracht, bood wel veel mogelijkheden om de data uiteindelijk makkelijk te verwerken. Om deze data uiteindelijk om te zetten naar grafieken is gebruik gemaakt van de library *dc.js*. Deze koppelt crossfilter aan D3-grafieken, en zorgt ervoor dat de performance-winst die wordt behaald door het gebruik maken van crossfilter kan worden doorgezet in het intieel tekenen van grafieken, en het latere hertekenen (na filteren van data) van deze grafieken.

Om de kaart te laten zien is gebruik gemaakt van de library *datamaps*. Er is een afweging gemaakt om hier reguliere GeoJSON/dc.js-kaarten van te maken, maar het toevoegen van de interactiviteit (het tonen van de top import/export van een land bijvoorbeeld) is hierin vrijwel onmogelijk, zonder handmatige berekeningen te doen over de huidige projectie / locatie van een punt op de kaart.

3 Interne en externe databronnen

In het project wordt gebruik gemaakt van een aantal externe databronnen. Allereerst wordt de externe databron genaamd *restcountries* ingeladen. Deze wordt gebruikt om middelpunten van landen te kunnen bepalen, om de locatie van de arcs in de datamap te kunnen berekenen. Ook wordt er een CSV-bestand ingeladen die oude landcodes (die voornamelijk gebruikt werden voor de tweede wereldoorlog) om kan zetten naar de nieuwe ISO-3 landcodes. Dit omdat de handels-dataset gebruik maakt van die verouderde codes (omdat deze ook historische data in zich heeft).

Uitendelijk wordt de expdata-dataset gebruikt om de handelscijfers te kunnen berekenen. Hiervoor moeten er kleine modificaties aan de data worden uitgevoerd (omdat een DSV-bestand by default geen type-data meegeeft). Uiteindelijk wordt al deze data gebundeld via de crossfilter-groeperingen om de data te kunnen tonen.

4 Designkeuzes

Voor de kleurcodering van de landen is gebruik gemaakt van de welbekende YlGnBu-schaal van ColorBrewer. Dit is een van de weinige schalen die esthetisch verantwoord is, maar deze maakt ook het ordenen van verschillende groepen makkelijk. Tevens zijn verschillende groepen (ook in relatief kleine intervallen) nog goed van elkaar te onderscheiden. Voor dit soort *range*-waardes zijn er weinig alternatieven te vinden. Een alternatief hiervoor zou de *viridis*-color map kunnen zijn, als het nodig is dat de kleuren ook goed zichtbaar moeten zijn voor iedereen, ook diegenen met een vorm van kleurenblindheid. Een goede implementatie hiervan was op het moment van schrijven alleen niet te vinden via de chroma.js-library.

Tevens is er veel gebruik gemaakt van animaties tussen de verschillende elementen van grafieken. De Y-assen worden automatisch geschaald als er een filter is gemaakt, of een land is uitgekozen. De X-schaal (van de tijd) wordt ook automatisch herschaald als er voor een andere tijdsspan wordt gekozen. Dit maakt het mogelijk dat de ontvanger van de visualisatie het idee krijgt dat er echt wat verandert aan de data, en op wat voor manier deze data verandert.

Een ander element waar aan is gedacht is het verminderen van *clutter* op het scherm. Alleen die elementen die nodig zijn om het verhaal te vertellen dat gewenst is. Dit maakt het mogelijk om het overzicht te behouden, zelfs als er

30 lijnen (de top 15 import en export-lijnen) worden getoond.