9 Visualização de dados para o usuário final

Este capítulo cobre

- ÿ Considerando opções de visualização de dados para seus usuários finais
- ÿ Configurando um Crossfilter MapReduce básico aplicativo
- ÿ Criando um painel com dc.js
- ÿ Trabalhando com ferramentas de desenvolvimento de painéis

CAPÍTULO FOCADO NA APLICAÇÃO Você notará rapidamente que este capítulo é certamente diferente dos capítulos 3 a 8, pois o foco aqui está na etapa 6 do o processo de ciência de dados. Mais especificamente, o que queremos fazer aqui é criar uma pequena aplicação de ciência de dados. Portanto, não seguiremos os dados etapas do processo científico aqui. Os dados utilizados no estudo de caso são apenas parcialmente real, mas funciona como dados fluindo da preparação de dados ou etapa de modelagem. Aproveite o passeio.

Freqüentemente, os cientistas de dados devem entregar seus novos insights ao usuário final. Os resultados pode ser comunicado de diversas maneiras:

- ÿ *Uma apresentação única* – As questões de pesquisa são negócios únicos porque a decisão de negócios derivada delas vinculará a organização a um determinado

claro, por muitos anos. Tomemos, por exemplo, as decisões de investimento das empresas: *distribuímos os nossos produtos a partir de dois centros de distribuição ou apenas de um? Onde eles precisa ser localizado para obter eficiência ideal?* Quando a decisão é tomada, o exercício não pode ser repetido até que você se aposente. Neste caso, os resultados são entregues como um relatório com uma apresentação como a cereja do bolo.

- *Uma nova janela de visualização dos seus dados – o* exemplo mais óbvio aqui é a segmentação de clientes. Claro, os próprios segmentos serão comunicados por meio de relatórios e apresentações, mas em essência constituem ferramentas e não o resultado final em si. Quando uma segmentação de clientes clara e relevante é descoberta, ela pode ser alimentada de volta ao banco de dados como uma nova dimensão nos dados dos quais foi derivado.

  A partir daí, as pessoas podem fazer seus próprios relatórios, como quantos produtos foram vendidos para cada segmento de clientes.

- *Um painel em tempo real — às vezes,* sua tarefa como cientista de dados não termina quando você descobriu as novas informações que estava procurando. Você pode enviar seu informações de volta ao banco de dados e pronto. Mas quando outras pessoas começar a fazer relatórios sobre esta pepita de ouro recém-descoberta, eles podem interpretar incorretamente e fazer relatórios que não fazem sentido. Como o cientista de dados quem descobriu esta nova informação, você deve dar o exemplo: fazer o primeiro relatório atualizável para que outros, principalmente repórteres e TI, possam entendê-lo e seguir seus passos. Fazer o primeiro dashboard também é uma forma de encurtar o tempo de entrega de seus insights ao usuário final que deseja usá-los diariamente base. Dessa forma, pelo menos eles já têm algo com que trabalhar até o O departamento de relatórios encontra tempo para criar um relatório permanente sobre o software de relatórios da empresa.

Você deve ter notado que alguns fatores importantes estão em jogo:

- Que *tipo de decisão* você está apoiando? É estratégico ou operacional? As decisões estratégicas muitas vezes exigem apenas que você analise e relate uma vez, enquanto as decisões operacionais exigem que o relatório seja atualizado regularmente.
- *Qual é o tamanho da sua organização?* Nos menores você será responsável por todo o ciclo: da coleta de dados à elaboração de relatórios. Nos maiores, uma equipe de repórteres pode estar disponível para criar os painéis para você. Mas mesmo nesta última situação, entregar um protótipo de dashboard pode ser benéfico porque apresenta uma exemplo e muitas vezes encurta o tempo de entrega.

Embora todo o livro seja dedicado à geração de insights, neste último capítulo iremos concentre-se em fornecer um painel operacional. Criando uma apresentação para promover suas descobertas ou a apresentação de insights estratégicos estão fora do escopo deste livro.

9.1 Opções de visualização de dados

Você tem várias opções para entregar um painel aos usuários finais. Aqui vamos concentre-se em uma única opção e, ao final deste capítulo, você mesmo será capaz de criar um painel.

O caso deste capítulo é o de uma farmácia hospitalar com um stock de alguns milhares
medicação. O governo lançou uma nova norma para todas as farmácias: todos os medicamentos deveriam ser
verificados quanto à sua sensibilidade à luz e armazenados em recipientes novos e especiais. Uma coisa que o
governo não forneceu às farmácias foi uma lista real
de medicamentos sensíveis à luz. Isso não é problema para você, como cientista de dados, porque
todo medicamento possui um folheto informativo ao paciente que contém essas informações. Você
destilar as informações com o uso inteligente de mineração de texto e atribuir um “sensível à luz”
ou etiqueta “não sensível à luz” para cada medicamento. Essas informações são então carregadas no
banco de dados central. Além disso, a farmácia precisa saber quantos recipientes
seria necessário. Para isso dão-lhe acesso aos dados do stock da farmácia. Quando
você desenha uma amostra apenas com as variáveis necessárias, o conjunto de dados se parece com a figura 9.1
quando aberto no Excel.

	A	B	C	D	E	F
1	MedName	LightSen	Date	StockOut	StockIn	Stock
2	Acupan 30 mg	No	1/01/2015	-8	150	142
3	Acupan 30 mg	No	2/01/2015	-6	5	141
4	Acupan 30 mg	No	3/01/2015	-2	0	139
5	Acupan 30 mg	No	4/01/2015	0	5	144
6	Acupan 30 mg	No	5/01/2015	-8	0	136
7	Acupan 30 mg	No	6/01/2015	-1	0	135
8	Acupan 30 mg	No	7/01/2015	-1	15	149
9	Acupan 30 mg	No	8/01/2015	-10	10	149
10	Acupan 30 mg	No	9/01/2015	-8	15	156

Figura 9.1 Conjunto de dados de medicamentos farmacêuticos aberto no Excel: as primeiras 10 linhas de dados de estoque são aprimoradas com uma variável de sensibilidade à luz

Como você pode ver, as informações são dados de séries temporais para um ano inteiro de movimentação de
estoque, portanto cada medicamento tem 365 entradas no conjunto de dados. Embora o estudo de caso
é existente e os medicamentos no conjunto de dados são reais, os valores dos outros
as variáveis aqui apresentadas foram geradas aleatoriamente, conforme os dados originais são classificados.
Além disso, o conjunto de dados está limitado a 29 medicamentos, pouco mais de 10.000 linhas de dados.
Mesmo que as pessoas criem relatórios usando crossfilter.js (um Javascript MapReduce
biblioteca) e dc.js (uma biblioteca de painéis Javascript) com mais de um milhão de linhas
de dados, por exemplo, você usará uma fração dessa quantidade. Além disso, não é recomendado carregar todo
o seu banco de dados no navegador do usuário; o navegador irá
congelar durante o carregamento e, se houver muitos dados, o navegador irá até travar. Normalmente
os dados são pré-calculados no servidor e partes deles são solicitadas usando, por exemplo, um
Serviço REST .

Para transformar esses dados em um painel real você tem muitas opções e pode encontrar
uma breve visão geral das ferramentas posteriormente neste capítulo.

Entre todas as opções, para este livro decidimos usar dc.js, que é um cruzamento entre a biblioteca
JavaScript MapReduce Crossfilter e a visualização de dados
biblioteca d3.js. O Crossfilter foi desenvolvido pela Square Register, empresa que lida com
transações de pagamento; é comparável ao PayPal, mas seu foco está no celular. Quadrado

desenvolveram o Crossfilter para permitir que seus clientes cortem e cortem dados extremamente rapidamente seu histórico de pagamentos. Crossfilter não é a única biblioteca JavaScript capaz de processar Map-Reduce, mas certamente faz o trabalho, é de código aberto, de uso gratuito, e é mantido por uma empresa estabelecida (Square). Exemplos de alternativas para filtro cruzado são Map.js, Meguro e Underscore.js. JavaScript pode não ser conhecido como dados linguagem complicada, mas essas bibliotecas dão aos navegadores da web um pouco mais de força caso os dados precisem ser manipulados no navegador. Não entraremos em detalhes sobre como o Java-Script pode ser usado para cálculos massivos em estruturas distribuídas colaborativas, mas um exército de anões pode derrubar um gigante. Se este tópico lhe interessa, você pode leia mais sobre isso em https://www.igvita.com/2009/03/03/collaborative-map-reduce-in-the-browser/ e em http://dyn.com/blog/browsers-vs-servers -usando-javascript-para-teorias-de-processamento-de-números/.

d3.js pode ser chamada com segurança de a biblioteca de visualização de dados JavaScript mais versátil disponível no momento da redação; foi desenvolvido por Mike Bostock como sucessor de seu Biblioteca Protovis. Muitas bibliotecas JavaScript são construídas sobre d3.js.

NVD3, C3.js, xCharts e Dimple oferecem aproximadamente a mesma coisa: uma abstração camada sobre d3.js, o que facilita o desenho de gráficos simples. Eles diferem principalmente no tipo de gráficos que eles suportam e em seu design padrão. Sinta-se à vontade para visitar seus sites e descobrir por si mesmo:

- ÿ NVD3—http://nvd3.org/
- ÿ C3.js—http://c3js.org/
- ÿ xCharts—http://tenxer.github.io/xcharts/
- ÿ Dimple—http://dimplejs.org/

Existem muitas opções. Então, por que dc.js?

O principal motivo: comparado ao que ele oferece, um painel interativo onde clicar em um gráfico criará visualizações filtradas em gráficos relacionados, dc.js é surpreendentemente fácil de configurar acima. É tão fácil que você terá um exemplo prático no final deste capítulo. Como um dado cientista, você já dedicou tempo suficiente à sua análise real; fácil de implementar painéis são um presente de boas-vindas.

Para ter uma ideia do que você está prestes a criar, você pode acessar o seguinte site, http://dc-js.github.io/dc.js/ e role para baixo até o exemplo NASDAQ , mostrado na figura 9.2.

Clique no painel e veja os gráficos reagindo e interagindo quando você seleciona e desmarque os pontos de dados. Não gaste muito tempo; é hora de criar isso você mesmo.

Conforme afirmado anteriormente, dc.js tem dois grandes pré-requisitos: d3.js e crossfilter.js. d3.js tem um curva de aprendizado íngreme e há vários livros sobre o assunto que vale a pena ler se você estiver interessado na personalização completa de suas visualizações. Mas para trabalhar com dc.js não é necessário nenhum conhecimento sobre ele, então não entraremos nisso neste livro. Crossfilter.js é outro assunto; você precisará ter um pouco de conhecimento desta biblioteca MapReduce para instalar o dc.js e executando em seus dados. Mas como o conceito de MapReduce em si não é novo, isso irá Vá tranquilamente.

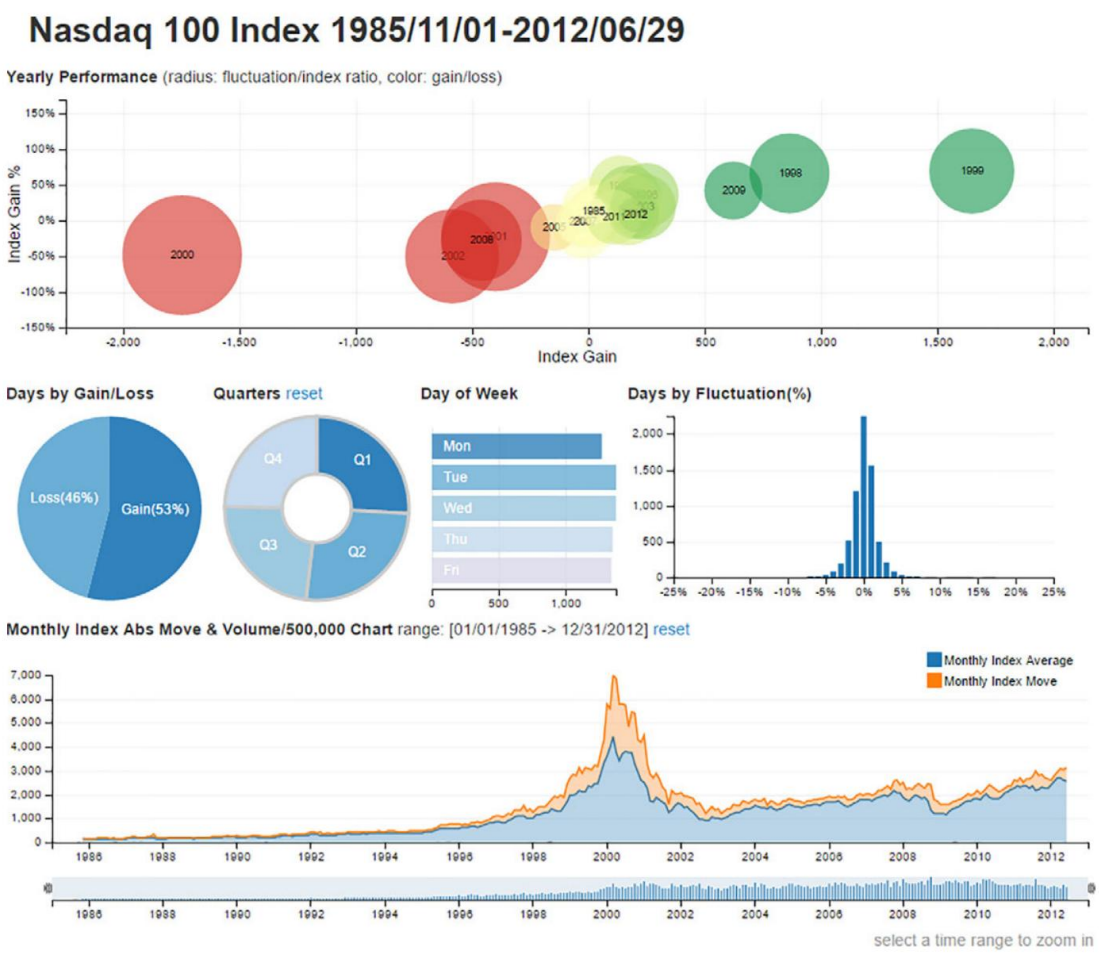


Figura 9.2 Um exemplo interativo do dc.js em seu site oficial

9.2 Crossfilter, a biblioteca JavaScript MapReduce

JavaScript não é a melhor linguagem para processamento de dados. Mas isso não impediu que pessoas, como o pessoal da Square, desenvolvessem bibliotecas MapReduce para isso. Se você estiver lidando com dados, todo ganho de velocidade ajuda. Você não deseja enviar enormes cargas de dados pela Internet ou mesmo pela sua rede interna, pelos seguintes motivos:

- O envio de uma grande quantidade de dados sobrecarregará a *rede* a ponto de incomodar outros usuários.
- O *navegador* está recebendo e, ao carregar os dados, ele irá *congelar temporariamente.* Para pequenas quantidades de dados, isso é imperceptível, mas quando você começa a observar 100.000 linhas, pode se tornar um atraso visível. Quando você passar

1.000.000 de linhas, dependendo da largura dos seus dados, seu navegador pode fornecer
sobre você.

Conclusão: é um exercício de equilíbrio. Para os dados que você envia, existe um Crossfilter para
cuide disso para você assim que chegar ao navegador. No nosso estudo de caso, o farmacêutico
solicitou ao servidor central dados de estoque de 2015 para 29 medicamentos nos quais ela estava particularmente
interessada. Já demos uma olhada nos dados, então vamos mergulhar no aplicativo em si.

9.2.1 Configurando tudo

É hora de construir o aplicativo real, e os ingredientes do nosso pequeno aplicativo dc.js são os seguintes:

- ÿ *JQuery—Para* lidar com a interatividade
- ÿ *Crossfilter.js — uma* biblioteca MapReduce e pré-requisito para dc.js
- ÿ *d3.js — Uma* biblioteca popular de visualização de dados e pré-requisito para dc.js
- ÿ *dc.js—A* biblioteca de visualização que você usará para criar seu painel interativo
- ÿ *Bootstrap — Uma* biblioteca de layout amplamente utilizada que você usará para melhorar a aparência de tudo

Você escreverá apenas três arquivos:

- ÿ *index.html—A* página HTML que contém seu aplicativo
- ÿ *application.js — Para* armazenar todo o código JavaScript que você escreverá
- ÿ *application.css—Para* seu próprio CSS

Além disso, você precisará executar nosso código em um servidor HTTP . Você poderia passar pelo
esforço de configurar um LAMP (Linux, Apache, MySQL, PHP), WAMP (Windows, Apache,
MySQL, PHP) ou servidor XAMPP (Cross Environment, Apache, MySQL, PHP, Perl). Mas
por uma questão de simplicidade, não configuraremos nenhum desses servidores aqui. Em vez disso você pode fazer
com um único comando Python. Use sua ferramenta de linha de comando (shell Linux ou CMD do Windows) e vá
para a pasta que contém seu index.html (quando estiver lá). Você
Você deve ter o Python instalado para outros capítulos deste livro, portanto o comando a seguir deve iniciar um
servidor HTTP Python em seu host local.

```
python -m SimpleHTTPServer
```

Para Python 3.4

```
python -m http.servidor 8000
```

Como você pode ver na figura 9.3, um servidor HTTP é iniciado na porta 8000 do host local.
navegador isso se traduz em “localhost:8000”; colocar “0.0.0.0:8000” não funcionará.

Figura 9.3 Iniciando um servidor HTTP Python simples

Certifique-se de ter todos os arquivos necessários disponíveis na mesma pasta do index.html. Você pode baixá-los no site da Manning ou nos sites de seus criadores.

- ÿ *dc.css e* dc.min.js — https://dc-js.github.io/dc.js/
- ÿ d3.v3.min.js—http://d3js.org/
- ÿ crossfilter.min.js—http://square.github.io/crossfilter/

Agora sabemos como executar o código que estamos prestes a criar, então vamos dar uma olhada no página index.html, mostrada na listagem a seguir.

Listagem 9.1 Uma versão inicial de index.html

```
<html>
<cabeça>
    <title>Capítulo 10. Aplicação de ciência de dados</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap /3.3.0/css/bootstrap.min.css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
3.3.0/css/bootstrap-theme.min.css">

    <link rel="stylesheet" href="dc.css">
    <link rel="stylesheet" href="application.css">
</head>
<corpo>

    <classe principal='contêiner'>
        <h1>Capítulo 10: Aplicação de ciência de dados</h1> <div class="row">

            <div class='col-lg-12'>
                <div id="inputtable" class="bem, bem-sm"></div>
            </div>
        </div>
        <div class="linha">
            <div class='col-lg-12'>
                <div id="filteredtable" class="bem, bem-sm"></div> </div>

        </div>
    </principal>
```

Todo CSS é carregado aqui.

Certifique-se de baixar dc.css da página de download de Manning ou do site da DC: https://dc-js.github.io/dc.js/. Deve estar presente no mesma pasta do arquivo index.html.

Principal recipiente incorpora tudo Visível para do utilizador.

```
        <script src="https://code.jquery.com/jquery-1.9.1.min.js"></script> <script src="https://
        maxcdn.bootstrapcdn.com/bootstrap/3.3.0 /js
/bootstrap.min.js"></script>

        <script src="crossfilter.min.js"></script> <script
        src="d3.v3.min.js"></script>
        <script src="dc.min.js"></script>
        <script src="application.js"></script>
</body>
</html>
```

Certifique-se de baixar crossfilter.min.js, d3.v3.min.js e dc.min.js de seus sites ou do site da Manning. Filtro cruzado: http://square.github.io/ crossfilter/, d3.js: http://d3js.org/, dc.min.js: https://dc-js.github.io/dc.js/.

Todo o Javascript é carregado aqui.

Não há surpresas aqui. O cabeçalho contém todas as bibliotecas CSS que você usará, então carregaremos nosso JavaScript no final do corpo HTML . Usando um manipulador de carga JQuery, seu o aplicativo será carregado quando o restante da página estiver pronto. Você começa com dois marcadores de tabela: um para mostrar a aparência dos seus dados de entrada, , e o outro será usado com Crossfilter para mostrar uma tabela filtrada, . Várias classes Bootstrap CSS foram usadas, como *"poço", "contêiner",* o sistema de grade Bootstrap com *"linha"* e *"col-xx-xx"* e assim por diante. Eles fazem com que tudo pareça mais bonito, mas eles não são obrigatórios. Mais informações sobre o As classes Bootstrap CSS podem ser encontradas em seu site em http://getbootstrap.com/css/.

Agora que você configurou seu HTML , é hora de mostrar seus dados na tela. Para Para fazer isso, volte sua atenção para o arquivo application.js que você criou. Primeiro, envolvemos todo o codifique "to be" em um manipulador onload JQuery.

```
$(função(){
        //Todo o código futuro terminará neste wrapper
})
```

Agora temos certeza de que nosso aplicativo será carregado somente quando todo o resto estiver pronto. Isso é importante porque usaremos seletores JQuery para manipular o HTML. É hora de carregar dados.

```
d3.csv('medicamentos.csv',function(dados) {
        dados principais)
});
```

Você não tem um serviço REST pronto e esperando por você, então, para o exemplo, você desenhe os dados de um arquivo .csv. Este arquivo está disponível para download no site de Manning. d3.js oferece uma função fácil para isso. Depois de carregar os dados, você os entrega ao seu função principal do aplicativo na função de retorno de chamada d3.csv.

Além da função principal você tem uma função CreateTable , que você usará para… você adivinhou… criar suas tabelas, conforme mostrado na listagem a seguir.

Listagem 9.2 A função CreateTable

```
var tableTemplate = $([
            "<table class='table table-hover table-condensed table-striped'>",
            "<caption></caption>",
            "<thead><tr/></thead>",
            "<tbody></tbody>",
            "</tabela>"
    ].join('\n'));

    CriarTabela = função(dados,variáveisInTabela,título){
        var tabela = tableTemplate.clone();
        var ths = variáveisInTable.map(function(v) { return $("<th>").text(v) });

        $('legenda', tabela).text(título);
        $('thead tr', tabela).append(ths);
        data.forEach(função(linha) {
            var tr = $("<tr>").appendTo($('tbody', tabela));
            variáveisInTable.forEach(function(varName) { var val = row, keys =
                varName.split('.'); keys.forEach(function(key) { val = val[key] });

                tr.append($("<td>").text(val));
            });
        });
        tabela de retorno;
    }
```

CreateTable() requer três argumentos:

- ÿ dados—Os dados necessários para colocar em uma tabela.
- ÿ variáveisInTable — quais variáveis ele precisa mostrar.
- ÿ Título — O título da tabela. É sempre bom saber o que você está vendo.

CreateTable() usa uma variável predefinida, tableTemplate, que contém nossa visão geral disposição da mesa. CreateTable() pode então adicionar linhas de dados a este modelo.

Agora que você tem seus utilitários, vamos à função principal do aplicativo, como mostrado na listagem a seguir.

Listagem 9.3 Função principal do JavaScript

```
principal = função(dadosdeentrada){

    var medicinaData = inputData ;

    var dateFormat = d3.time.format("%d/%m/%Y"); medicinaData.forEach
    (função (d) {
        d.Dia = dateFormat.parse(d.Data);
    })
    var variáveisInTable =
['MedName','StockIn','StockOut','Stock','Data','LightSen']
    var amostra = medicinaData.slice(0,5);
    varinputTable = $("#inputtable");
```

Nossos dados: normalmente são obtidos de um servidor, mas neste caso nós os lemos de um arquivo .csv local

Coloque as variáveis vamos mostrar no tabela em uma matriz então podemos fazer um loop por eles ao criar código da tabela

Converta a data para o formato correto para que o Crossfilter reconheça a variável de data

Mostrar apenas uma amostra de dados

Criar a tabela

```
                tabela de entrada
                        .vazio()
                        .append(CreateTable(sample,variablesInTable,"A tabela de entrada"));
}
```

Você começa mostrando seus dados na tela, mas de preferência não todos; apenas o as primeiras cinco entradas servirão, conforme mostrado na figura 9.4. Você tem uma variável de data em seus dados e você quer ter certeza de que o Crossfilter o reconhecerá como tal mais tarde, então primeiro analise-o e crie uma nova variável chamada Day. Você mostra o original, Data, para aparecer em tabela por enquanto, mas mais tarde você usará Day para todos os seus cálculos.

The input table

MedName	StockIn	StockOut	Stock	Date	LightSen
Acupan 30 mg	150	-7	143	1/01/2015	No
Acupan 30 mg	5	-6	142	2/01/2015	No
Acupan 30 mg	15	-9	148	3/01/2015	No
Acupan 30 mg	0	-11	137	4/01/2015	No
Acupan 30 mg	10	-8	139	5/01/2015	No

Figura 9.4 Tabela de entrada de medicamentos mostrada no navegador: primeiras cinco linhas

É assim que você acaba: a mesma coisa que você viu no Excel antes. Agora que você sabe que o básico está funcionando, você introduzirá o Crossfilter na equação.

9.2.2 Liberando o Crossfilter para filtrar o conjunto de dados do medicamento

Agora vamos entrar no Crossfilter para usar a filtragem e o MapReduce. Doravante você pode coloque todo o próximo código após o código da seção 9.2.1 dentro da função main() . A primeira coisa que você precisa fazer é declarar uma instância do Crossfilter e iniciá-la com seus dados.

```
CrossfilterInstance = crossfilter(medicineData);
```

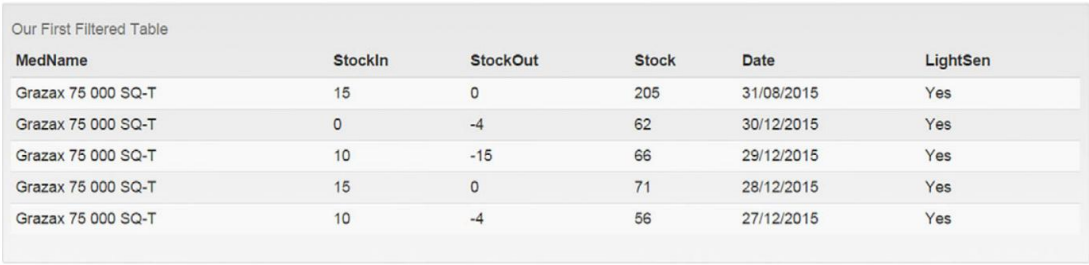
A partir daqui você pode começar a trabalhar. Nesta instância você pode cadastrar dimensões, que são as colunas da sua tabela. Atualmente o Crossfilter está limitado a 32 dimensões. Se você está lidando com dados maiores que 32 dimensões, considere restringi-los para baixo antes de enviá-lo para o navegador. Vamos criar nossa primeira dimensão, a medicina dimensão do nome:

```
var medNameDim = CrossfilterInstance.dimension(function(d) {return d.MedName;});
```

Sua primeira dimensão é o nome dos medicamentos, e você já pode usar isso para filtrar seu conjunto de dados e mostrar os dados filtrados usando nossa função CreateTable() .

```
var dataFiltered= medNameDim.filter('Grazax 75 000 SQ-T')
var tabelafiltrada = $('#tabelafiltrada');
tabela filtrada
.empty().append(CreateTable(dataFiltered.top(5),variablesInTable,'Nosso
Primeira tabela filtrada'));
```

Você mostra apenas as cinco principais observações (figura 9.5); você tem 365 porque você tem os resultados de um único medicamento durante um ano inteiro.



Our First Filtered Table

MedName	StockIn	StockOut	Stock	Date	LightSen
Grazax 75 000 SQ-T	15	0	205	31/08/2015	Yes
Grazax 75 000 SQ-T	0	-4	62	30/12/2015	Yes
Grazax 75 000 SQ-T	10	-15	66	29/12/2015	Yes
Grazax 75 000 SQ-T	15	0	71	28/12/2015	Yes
Grazax 75 000 SQ-T	10	-4	56	27/12/2015	Yes

Figura 9.5 Dados filtrados pelo nome do medicamento Grazax 75 000 SQ-T

Esta tabela não parece ordenada, mas está. A função top() classificou-a em medicina nome. Como você tem apenas um medicamento selecionado, isso não importa. Classificando em date é bastante fácil usando sua nova variável Day . Vamos registrar outra dimensão, a dimensão de data:

```
var DateDim = CrossfilterInstance.dimension(
function(d) {return d.Day;});
```

Agora podemos classificar por data em vez de nome do medicamento:

```
tabela filtrada
.vazio()
.append(CreateTable(DateDim.bottom(5),variablesInTable,'Nossa Primeira Tabela Filtrada'));
```

O resultado é um pouco mais atraente, como mostra a figura 9.6.

Esta tabela oferece uma visão de janela dos seus dados, mas não os resume para você ainda. É aqui que entram os recursos do Crossfilter MapReduce.
gostaria de saber quantas observações você tem por medicamento. A lógica dita que você

Our First Filtered Table

MedName	StockIn	StockOut	Stock	Date	LightSen
Grazax 75 000 SQ-T	65	-12	53	1/01/2015	Yes
Grazax 75 000 SQ-T	15	-11	57	2/01/2015	Yes
Grazax 75 000 SQ-T	5	-9	53	3/01/2015	Yes
Grazax 75 000 SQ-T	5	-4	54	4/01/2015	Yes
Grazax 75 000 SQ-T	0	-14	40	5/01/2015	Yes

Figura 9.6 Dados filtrados pelo nome do medicamento Grazax 75 000 SQ-T e classificados por dia

deveria terminar com o mesmo número para todos os medicamentos: 365, ou 1 observação por dia em 2015.

```
var countPerMed = medNameDim.group().reduceCount(); variáveisInTable = ["chave", "valor"]
tabela filtrada


.empty() .append(CreateTable(countPerMed.top(Infinity),variáveisInTable,'Tabela
Reduzida'));
```

Crossfilter vem com duas funções MapReduce: reduzirCount() e reduzirSum(). Se você quiser fazer algo além de contar e somar, você precisa escrever funções de redução para isso. A variável countPerMed agora contém os dados agrupados pela dimensão do medicamento e uma contagem de linhas para cada medicamento na forma de uma chave e um valor.
Para criar a tabela você precisa endereçar a variável key em vez de medName e valor para a contagem (figura 9.7).

Reduced Table

key	value
Adoport 1 mg	365
Atenolol EG 100 mg	365
Ceftriaxone Actavis 1 g	365
Cefuroxim Mylan 500 mg	365
Certican 0.25 mg	365

Figura 9.7 Tabela MapReduced com o medicamento como grupo e uma contagem de linhas de dados como valor

Ao especificar .top(Infinity) você pede para mostrar todos os 29 medicamentos na tela, mas para economizar papel a figura 9.7 mostra apenas os primeiros cinco resultados. Ok, você pode ficar tranquilo; os dados contêm 365 linhas por medicamento. Observe como o Crossfilter ignorou o filtro em “Grazax”. Se uma dimensão for usada para agrupamento, o filtro não se aplicará a ela. Somente filtros em outras dimensões podem restringir os resultados.

E quanto a cálculos mais interessantes que não vêm junto com o Crossfil-ter, como uma média, por exemplo? Você ainda pode fazer isso, mas precisará escrever três funções e alimentá-las para um método .reduce() . Digamos que você queira saber o estoque médio por medicamento. Como mencionado anteriormente, quase toda a lógica do MapReduce precisa ser escrito por você. Uma média nada mais é do que a divisão da soma por conte, então você precisará de ambos; como você faz isso? Além das funções reduzir-Count() e reduzirSum() , Crossfilter tem a função reduzir() mais geral . Esta função leva três argumentos:

- ÿ *A* função *reduzAdd() — Uma* função que descreve o que acontece quando um valor extra observação é adicionada.
- ÿ *A* função *reduzRemove() —Uma* função que descreve o que precisa acontecer quando uma observação desaparece (por exemplo, porque um filtro é aplicado).
- ÿ *A função reduzInit()* — Esta define os valores iniciais para tudo o que é calculado. Para uma soma e contagem, o ponto de partida mais lógico é 0.

Vejamos as funções de redução individuais que você precisará antes de tentar chamar o Método Crossfilter .reduce() , que toma esses três componentes como argumentos. A A função de redução personalizada requer três componentes: uma iniciação, uma função de adição e uma função de remoção. A função de redução inicial definirá os valores iniciais do objeto p :

```
var reduzInitAvg = function(p,v){
    return {contagem: 0, stockSum: 0, stockAvg:0};
}
```

Como você pode ver, as próprias funções de redução recebem dois argumentos. Eles são alimentados automaticamente pelo método Crossfilter .reduce() : ÿ p é um objeto que

contém a situação de combinação até o momento; persiste acima de tudo observações. Esta variável controla a soma e a contagem para você e, portanto, representa seu objetivo, seu resultado final.

- ÿ v representa um registro dos dados de entrada e tem todas as suas variáveis disponíveis para você. Ao contrário de p, não persiste, mas é substituído por uma nova linha de dados toda vez a função é chamada. O reduzInit() é chamado apenas uma vez, mas reduzAdd() é chamado toda vez que um registro é adicionado e reduzRemove() toda vez que uma linha de os dados são removidos.
- ÿ A função reduzInit() , aqui chamada de reduzirInitAvg() porque você está calcular uma média, basicamente inicializa o objeto p definindo seu componentes (contagem, soma e média) e definindo seus valores iniciais. Vamos veja reduzirAddAvg():

```
var reduzirAdicionarAvg = função(p,v){
    contagem += 1;
    p.stockSum = p.stockSum + Número (v.Stock);
    p.stockAvg = Math.round(p.stockSum / p.count);
    retornar p;
}
```

reduzAddAvg() usa os mesmos argumentos p e v, mas agora você realmente usa v; você não precisa de seus dados para definir os valores iniciais de p neste caso, embora possa, se quiser. Seu estoque é somado para cada registro adicionado e, em seguida, a média é calculada com base na soma acumulada e na contagem de registros:

```
var reduzirRemoverAvg = função(p,v){
        contagem -= 1;
        p.stockSum = p.stockSum - Número(v.Stock); p.stockAvg =
        Math.round(p.stockSum / p.count); retornar p;

}
```

A função reduzRemoveAvg() é semelhante, mas faz o oposto: quando um registro é removido, a contagem e a soma são reduzidas. A média é sempre calculada da mesma forma, portanto não há necessidade de alterar essa fórmula.

A hora da verdade: você aplica esta função MapReduce caseira ao conjunto de dados:

```
dataFiltered = medNameDim.group().reduce(reduceAddAvg,
reduzirRemoverAvg,reduzirInitAvg)

variáveisInTable = ["chave","valor.stockAvg"]
tabela filtrada
        .vazio()
.append(CreateTable(dataFiltered.top(Infinity),variáveisInTable,'Tabela
Reduzida'));
```

Business as usual: desenhe a tabela de resultados.

reduz() usa as 3 funções (reduceInitAvg(), reduzAddAvg() e reduzRemoveAvg()) como argumentos de entrada.

Observe como o nome da sua variável de saída mudou de valor para valor .stockAvg. Como você mesmo definiu as funções de redução , poderá gerar muitas variáveis, se desejar. Portanto, value mudou para um objeto contendo todas as variáveis que você calculou; stockSum e count também estão lá.

Os resultados falam por si, como mostra a figura 9.8. Parece que pegamos emprestado Cimalgex de outros hospitais, entrando em estoque médio negativo.

Este é todo o Crossfilter que você precisa saber para trabalhar com dc.js, então vamos seguir em frente e apresentar esses gráficos interativos.

Reduced Table

key	value.stockAvg
Adoport 1 mg	36
Atenolol EG 100 mg	49
Ceftriaxone Actavis 1 g	207
Cefuroxim Mylan 500 mg	118
Certican 0.25 mg	158
Cimalgex 8 mg	-24

Figura 9.8 Tabela MapReduced com estoque médio por medicamento

9.3 Criando um painel interativo com dc.js

Agora que você conhece o básico do Crossfilter, é hora de dar o passo final: construir o painel. Vamos começar reservando um lugar para seus gráficos na página index.html .
O novo corpo se parece com a listagem a seguir. Você notará que é semelhante à nossa configuração inicial, exceto pelas tags do marcador de posição do gráfico adicionadas e pelo botão de redefinição <botão> tag.

Listagem 9.4 Um index.html revisado com espaço para gráficos gerados por dc.js

```
Disposição:
        Título
        | tabela de                                    | (linha 1)
        entrada | tabela                             | (linha 2)
        filtrada [botão reset]
        | gráfico de ações ao longo do tempo | gráfico de estoque por medicamento | (linha 3)
        | gráfico sensível à luz | | (linha 4)
            (coluna 1)              (coluna 2)

<corpo>
    <classe principal='contêiner'>

        <h1>Capítulo 10: Aplicação de ciência de dados</h1>
        <div class="linha">
            <div class='col-lg-12'>
                <div id="inputtable" class="bem, bem-sm">
</div>                                                          Este é um espaço
                                                                reservado <div> para dados de entrada
            </div>                                              tabela inserida posteriormente.
        </div>
        <div class="linha">
            <div class='col-lg-12'>
                <div id="filteredtable" class="bem, bem-sm">
</div>                                                          Este é um espaço
                                                                reservado <div> para filtrado
            </div>                                              tabela inserida posteriormente.
Isso é  </div>
novo: redefinir
botão.  <button class="btn btn-success">Redefinir filtros</button> <div class="row">
                                                                Isto é novo:
                                                                espaço
            <div class="col-lg-6">                              reservado para gráfico de tempo
                <div id="StockOverTime" class="bem, bem-sm"></div>
Isso é novo: luz
espaço reservado para
gráfico de pizza de sensibilidade.
                <div id="LightSensitiveStock" class="bem, bem-sm"></div>
            </div>
            <div class="col-lg-6">
                <div id="StockPerMedicine" class="bem, bem-sm"></div> </div>

        </div>
    </principal>                                                Isto é novo: estoque por medicamento
                                                                espaço reservado para gráfico de barras.
```

```
<script src="https://code.jquery.com/jquery-1.9.1.min.js"></script> <script src="https://maxcdn.bootstrapcdn.com/
bootstrap
/3.3.0/js/bootstrap.min.js"></script>

<script src="crossfilter.min.js"></script> <script src="d3.v3.min.js"></
script>
<script src="dc.min.js"></script>

<script src="application.js"></script> </body>
```

As bibliotecas Crossfilter, d3 e dc podem ser baixadas de seus respectivos sites.

Nosso próprio código JavaScript do aplicativo.

Prática padrão. As bibliotecas JS são as últimas a acelerar o carregamento da página.

jQuery: interação HTML-JavaScript vital
Bootstrap: CSS e layout simplificados do pessoal do Twitter
Crossfilter: nossa biblioteca JavaScript MapReduce preferida
d3: o script d3, necessário para executar dc.js
DC: nossa biblioteca de visualização
aplicação: nossa aplicação de ciência de dados; aqui armazenamos toda a lógica
***.min.js denota JavaScript minificado para nossas bibliotecas de terceiros**

Temos a formatação do Bootstrap em andamento, mas os elementos mais importantes são os três tags com IDs e o botão. O que você quer construir é uma representação do estoque total ao longo do tempo, , com possibilidade de filtragem de medicamentos, , e se eles são sensíveis à luz ou não, . Você também quer um botão para redefina todos os filtros, <button class="btn btn-success">Redefinir filtros</button>. Este elemento do botão de redefinição não é obrigatório, mas é útil.

Agora volte sua atenção para application.js. Aqui você pode adicionar todo o código futuro em sua função main() como antes. Há, no entanto, uma exceção à regra: dc.renderAll(); é o comando de dc para desenhar os gráficos. Você precisa colocar isso render comando apenas uma vez, na parte inferior da sua função main() . O primeiro gráfico o que você precisa é o “estoque total ao longo do tempo”, conforme mostrado na listagem a seguir. Você já tem a dimensão de tempo declarada, então tudo que você precisa é somar seu estoque por tempo dimensão.

Listagem 9.5 Código para gerar o gráfico “estoque total ao longo do tempo”

```
var SummatedStockPerDay =
DateDim.group().reduceSum(function(d){return d.Stock;})

var minDate = DateDim.bottom(1)[0].Dia;
var maxDate = DateDim.top(1)[0].Dia;
var StockOverTimeLineChart = dc.lineChart("#StockOverTime");

Gráfico StockOverTimeLine
        .width(null) // null significa tamanho para caber no contêiner .height(400)

        .dimension(DateDim)
        .group(SomatedStockPerDay)
```

Dados de estoque ao longo do tempo

Gráfico de linha

Entregas por dia gráfico

```
Entregas
por dia
gráfico                .x(d3.time.scale().domain([minDate,maxDate]))
                       .xAxisLabel("Ano 2015")
                       .yAxisLabel("Estoque")
                       .margins({esquerda: 60, direita: 50, superior: 50, inferior: 50})

dc.renderAll();                    Renderizar tudo
                                   gráficos
```

Veja tudo o que está acontecendo aqui. Primeiro você precisa calcular o intervalo do seu eixo x então dc.js saberá onde começar e terminar o gráfico de linhas. Em seguida, o gráfico de linhas é inicializado e configurado. Os métodos menos autoexplicativos aqui são .group() e .dimensão(). .group() assume a dimensão do tempo e representa o eixo x. .dimension() é sua contraparte, representando o eixo y e considerando seus dados somados como entrada. A Figura 9.9 parece um gráfico de linhas enfadonho, mas as aparências enganam.

Figura 9.9 Gráfico dc.js: soma do estoque de medicamentos no ano de 2015

As coisas mudam drasticamente quando você introduz um segundo elemento, então vamos criar uma linha gráfico que representa o estoque médio por medicamento, conforme listagem a seguir.

Listagem 9.6 Código para gerar o gráfico "estoque médio por medicamento"

```
var AverageStockPerMedicineRowChart = dc.rowChart("#StockPerMedicine"); var AvgStockMedicine =
medNameDim.group().reduce(reduceAddAvg,
reduzirRemoveAvg, reduzirInitAvg);                                          Estoque médio
                                                                            por medicamento
Gráfico de estoque médio por linha de medicamentos                          gráfico de linhas
                        .largura(nulo)          Nulo significa "tamanho
                        .altura(1200)           para caber no recipiente"
```

```
.dimension(medNameDim)
.group(MédiaEstoqueMedicina)
.margins({superior: 20, esquerda: 10, direita: 10, inferior: 20})
.valueAccessor(função (p) {return p.value.stockAvg;});
```

Isso deve ser familiar porque é uma representação gráfica da tabela que você criou mais cedo. Um grande ponto de interesse: como você usou uma função reduzir() personalizada desta vez, dc.js não sabe quais dados representar. Com o .valueAccessor() método, você pode especificar p.value.stockAvg como o valor de sua escolha. A linha dc.js a cor da fonte do rótulo do gráfico é cinza; isso torna seu gráfico de linhas um pouco difícil de ler. Você Você pode remediar isso substituindo seu CSS no arquivo application.css:

```
.dc-chart g.row texto {preenchimento: preto;}
```

Uma simples linha pode fazer a diferença entre um gráfico claro e um obscuro (figura 9.10).

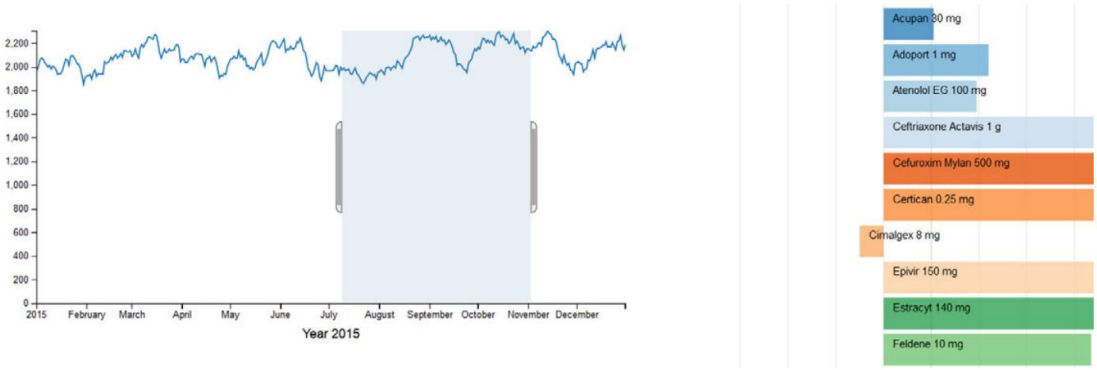


Figura 9.10 Gráfico de linhas dc.js e interação do gráfico de linhas

Agora, quando você seleciona uma área no gráfico de linhas, o gráfico de linhas é automaticamente adaptado para representar os dados do período de tempo correto. Inversamente, você pode selecionar um ou vários medicamentos no gráfico de linhas, fazendo com que o gráfico de linhas se ajuste de acordo. Finalmente, vamos adicionar a dimensão da sensibilidade à luz para que o farmacêutico possa distinguir entre estoque de medicamentos fotossensíveis e não fotossensíveis, conforme listagem a seguir.

Listagem 9.7 Adicionando a dimensão de sensibilidade à luz

```
        var lightSenDim = CrossfilterInstance.dimension(
função(d){return d.LightSen;}); var SummatedStockLight
        = lightSenDim.group().reduceSum(
função(d) {return d.Estoque;});

        var LightSensitiveStockPieChart = dc.pieChart("#LightSensitiveStock");
```

```
Gráfico LightSensitiveStockPie
        .width(null) // null significa tamanho para caber no contêiner
        .altura(300)
        .dimension(lightSenDim)
        .raio(90)
        .group(SomatedStockLight)
```

Ainda não havíamos introduzido a dimensão da luz, então você precisa registrá-la no seu Instância de filtro cruzado primeiro. Você também pode adicionar um botão de redefinição, que faz com que todos os filtros sejam redefinidos, conforme mostrado na listagem a seguir.

Listagem 9.8 O botão de redefinição de filtros do painel

```
resetFiltros=função(){
        StockOverTimeLineChart.filterAll();
        LightSensitiveStockPieChart.filterAll();
        AverageStockPerMedicineRowChart.filterAll(); dc.redrawAll();

}
$('.btn-sucesso').click(resetFilters);
```

Quando um elemento com classe btn-success é clicado (nosso botão de reset), redefinirFiltros() é chamado.

A função resetFilters() irá redefinir nossos dados dc.js e redesenhar gráficos.

O método .filterAll() remove todos os filtros em uma dimensão específica; dc.redraw-All() aciona manualmente todos os gráficos DC para redesenhar.

O resultado final é um dashboard interativo (figura 9.11), pronto para ser utilizado pelos nossos farmacêutico para obter informações sobre o comportamento de seu estoque.

Figura 9.11 Painel dc.js totalmente interativo sobre medicamentos e seu estoque na farmácia hospitalar

9.4 Ferramentas de desenvolvimento de dashboard

Já temos nosso glorioso painel, mas queremos encerrar este capítulo com um breve visão geral (e longe de ser exaustiva) das escolhas alternativas de software quando se trata para apresentar seus números de uma forma atraente.

Você pode optar por pacotes de software comprovados e verdadeiros de desenvolvedores renomados, como Tableau, MicroStrategy, Qlik, SAP, IBM, SAS, Microsoft, Spotfire e assim por diante. Todas essas empresas oferecem ferramentas de painel que valem a pena investigar. Se você trabalha em uma grande empresa, há boas chances de você ter pelo menos uma dessas ferramentas pagas à sua disposição. Desenvolvedores também pode oferecer versões públicas gratuitas com funcionalidade limitada. Definitivamente, verifique o Tab-leau, se ainda não o fez, em http://www.tableausoftware.com/public/download.

Outras empresas fornecerão pelo menos uma versão de teste. No final você tem que pagar a versão completa de qualquer um desses pacotes, e pode valer a pena, especialmente para uma empresa maior que pode pagar por isso.

No entanto, o foco principal deste livro está nas ferramentas gratuitas. Ao procurar ferramentas gratuitas de visualização de dados, você rapidamente acaba no mundo HTML , que prolifera com ferramentas gratuitas Bibliotecas JavaScript para plotar quaisquer dados que você desejar. A paisagem é enorme:

- *HighCharts — Uma* das bibliotecas gráficas baseadas em navegador mais maduras. O a licença gratuita se aplica apenas a atividades não comerciais. Se você quiser usá-lo em um contexto comercial, os preços variam entre US$ 90 e US$ 4.000. Veja http://loja .highsoft.com/highcharts.html.
- *Chartkick — Uma* biblioteca de gráficos JavaScript para fãs de Ruby on Rails. Veja http://ank-ane.github.io/chartkick/.
- *Google Charts – A* biblioteca de gráficos gratuita do Google. Tal como acontece com muitos produtos do Google, seu uso é gratuito, até mesmo comercialmente, e oferece uma ampla variedade de gráficos. Ver https://developers.google.com/chart/.
- d3.js — Esta é uma opção estranha porque não é uma biblioteca gráfica, mas uma biblioteca de visualização de dados. A diferença pode parecer sutil, mas as implicações não são. Enquanto bibliotecas como HighCharts e Google Charts se destinam a desenhar determinados gráficos predefinidos, o d3.js não estabelece tais restrições. d3.js é atualmente a biblioteca de visualização de dados JavaScript mais versátil disponível. Você precisa apenas de um dê uma olhada rápida nos exemplos interativos no site oficial para entender o diferença de uma biblioteca regular de construção de gráficos. Consulte http://d3js.org/.

Claro, outros estão disponíveis que não mencionamos.

Você também pode obter bibliotecas de visualização que vêm apenas com um período de teste e sem edição comunitária gratuita, como Wijmo, Kendo e FusionCharts. Eles valem olhando porque eles também fornecem suporte e garantem atualizações regulares.

Você tem opções. Mas por que ou quando você consideraria construir o seu próprio interface com HTML5 em vez de usar alternativas como BusinessObjects da SAP , SAS JMP, Tableau, Clickview ou um dos muitos outros? Aqui estão algumas razões:

- *Sem orçamento – quando* você trabalha em uma startup ou outra pequena empresa, o licenciamento os custos que acompanham este tipo de software podem ser elevados.

- *Alta acessibilidade – O* aplicativo de ciência de dados destina-se a liberar resultados para qualquer tipo de usuário, especialmente pessoas que podem ter apenas um navegador à disposição – seus próprios clientes, por exemplo. A visualização de dados em HTML5 funciona fluentemente em dispositivos móveis.

- *Grandes reservas de talentos* disponíveis: embora não haja muitos desenvolvedores do Tableau, muitas pessoas têm habilidades de desenvolvimento web. Ao planejar um projeto, é importante levar em consideração se você pode contratar pessoal.
- *Liberação rápida — percorrer* todo o ciclo de TI pode demorar muito para você empresa e você deseja que as pessoas aproveitem sua análise rapidamente. Uma vez que sua interface esteja disponível e sendo usada, a TI pode levar todo o tempo que quiser para industrializar o produto.

- *Prototipagem – Quanto* melhor você mostrar à TI seu propósito e do que ela deve ser capaz, mais fácil será para eles construir ou comprar um aplicativo sustentável que faz o que você quer que faça.
- *Personalização—Embora* os pacotes de software estabelecidos sejam ótimos no que fazer, um aplicativo nunca poderá ser tão personalizado como quando você mesmo o cria.

E por que você não faria isso?

- *Política da empresa – Esta* é a maior delas: não é permitido. As grandes empresas têm Equipes de backup de TI que permitem o uso de apenas um determinado número de ferramentas para que possam manter seu papel de apoio sob controle.
- *Você tem uma equipe experiente de repórteres à sua disposição. Você* estaria fazendo o trabalho deles, e eles podem vir atrás de você com forcados.
- *Sua ferramenta permite personalização suficiente para se adequar ao seu* gosto — Várias das plataformas maiores são interfaces de navegador com JavaScript rodando nos bastidores. Quadro, BusinessObjects Webi, SAS Visual Analytics e assim por diante possuem interfaces HTML ; sua tolerância à personalização pode aumentar com o tempo.

O front-end de qualquer aplicativo pode conquistar o coração da multidão. Todo o trabalho duro você investe na preparação de dados e as análises sofisticadas que você aplicou valem apenas tanto quanto você pode transmitir para aqueles que o usam. Agora você está no caminho certo para alcançar esse. Com esta nota positiva concluiremos este capítulo.

9.5 Resumo

- Este capítulo enfocou a última parte do processo de ciência de dados e nosso objetivo era construir um aplicativo de ciência de dados onde o usuário final recebe um painel interativo. Depois de passar por todas as etapas do processo de ciência de dados, somos apresentados a dados limpos, muitas vezes compactados ou com muitas informações. Dessa forma, podemos consultar menos dados e obter os insights que desejamos.
- No nosso exemplo, os dados do estoque da farmácia são considerados completamente limpos e preparado e este deve ser sempre o caso no momento em que a informação chega ao usuário final.

ÿ Painéis baseados em JavaScript são perfeitos para conceder acesso rápido aos resultados de ciência de dados porque exigem apenas que o usuário tenha um navegador da web. Existem alternativas, como o Qlik (capítulo 5). ÿ

Crossfilter é uma biblioteca MapReduce, uma das muitas bibliotecas JavaScript MapReduce, mas provou sua estabilidade e está sendo desenvolvida e usada pela Square, uma empresa que faz transações monetárias. A aplicação do MapReduce é eficaz, mesmo em um único nó e em um navegador; aumenta a velocidade de cálculo. ÿ dc.js é uma

biblioteca de gráficos criada com base em d3.js e Crossfilter que permite uma rápida construção do painel do navegador.

ÿ Exploramos o conjunto de dados de uma farmácia hospitalar e construímos um painel interativo para farmacêuticos. A força de um painel é sua natureza *de autoatendimento :* eles nem sempre precisam de um repórter ou cientista de dados para fornecer os insights que desejam.

ÿ Alternativas de visualização de dados estão disponíveis e vale a pena dedicar algum tempo para encontrar aquela que melhor atende às suas

necessidades. ÿ Há vários motivos pelos quais você criaria seus próprios relatórios personalizados em vez de optar pelas ferramentas da empresa (geralmente mais caras): – *Sem* orçamento – as startups nem sempre podem pagar por todas as ferramentas – *Alta acessibilidade* – *Todo mundo* tem um navegador – Talento *disponível* — Acesso (comparativamente) fácil aos desenvolvedores de JavaScript — *Liberação rápida* — Os ciclos de TI podem demorar um pouco — *Prototipagem* — *Um* aplicativo protótipo pode fornecer e deixar tempo para a TI construir a versão de produção — *Personalização* — *Às vezes* você só quer que ele seja exatamente como seu sonhos imaginem isso.

ÿ É claro que existem razões contra o desenvolvimento de seu próprio aplicativo:

– Política *da empresa* – A proliferação de aplicações não é uma coisa boa e a empresa pode querer evitar isso restringindo o desenvolvimento local.

– *Equipe de reportagem madura* – *Se* você tem um bom departamento de reportagem, por que ainda se preocuparia?

– *A personalização é satisfatória* – *nem* todo mundo quer coisas brilhantes; lata básica seja suficiente.

Parabéns! Você chegou ao final deste livro e ao verdadeiro início de sua carreira como cientista de dados. Esperamos que você tenha se divertido bastante lendo e trabalhando nos exemplos e estudos de caso. Agora que você tem uma visão básica do mundo da ciência de dados, cabe a você escolher um caminho. A história continua e todos desejamos a você muito sucesso em sua busca para se tornar o maior cientista de dados que já existiu! Que possamos nos encontrar novamente algum dia. ;)