Visualization in Industry

Kevin Lynagh
Keming Labs @lynaghk

2012 October 17
IEEE VisWeek
Seattle, Washington

Agenda

Agenda

4

Cool stuff we made this year

Agenda

2

Cool stuff we're working on

Agenda

4

Cool stuff we made this year

Agenda

Cool stuff we made this year

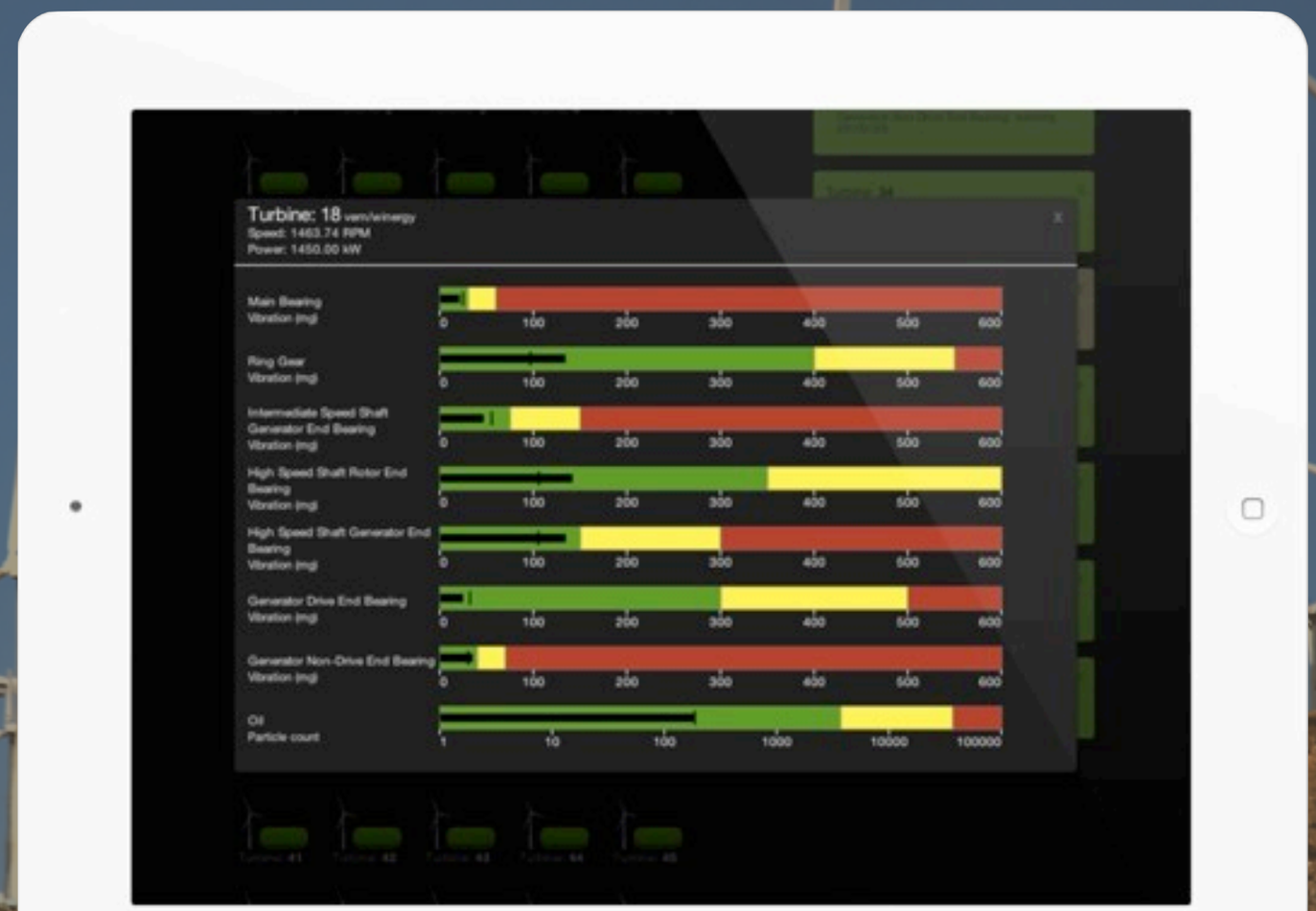
Algorithm 1: Business

```
while people have (problems + money) do
    find them;
    if you can solve their problem then
        convince them;
        solve their problem;
        take their money;
    end
end
```

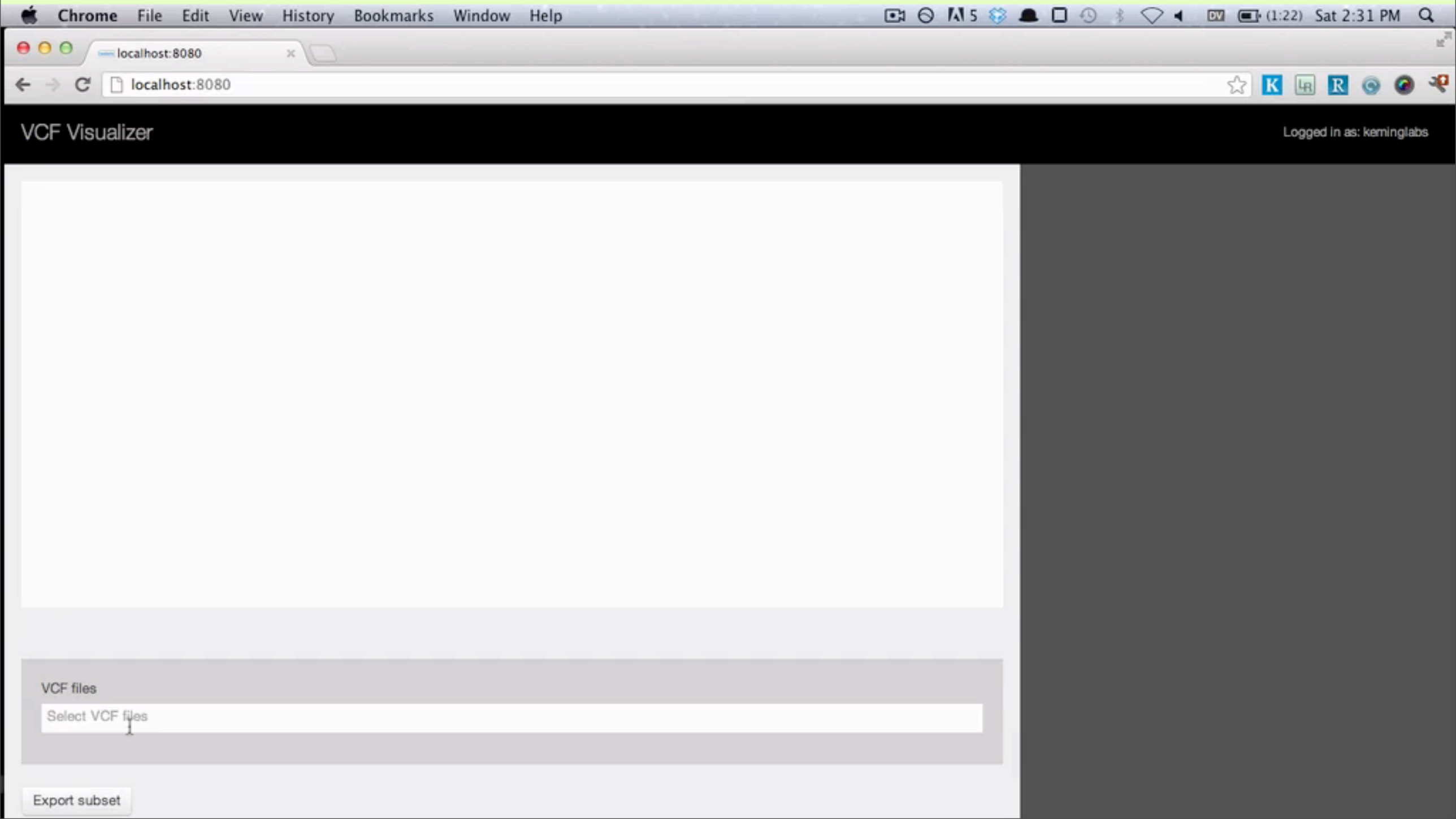
Algorithm 1: Business

while *people have (problems + money)* **do**
    find them;
    if *you can solve their problem* **then**
        convince them;
        solve their problem;
        take their money;
    end
end

Wind energy


Turbine: 18 vem/winergy
Speed: 1463.74 RPM
Power: 1450.00 kW
Main Bearing Vibration (mg)
Ring Gear Vibration (mg)
Intermediate Speed Shaft Generator End Bearing Vibration (mg)
High Speed Shaft Rotor End Bearing Vibration (mg)
High Speed Shaft Generator End Bearing Vibration (mg)
Generator Drive End Bearing Vibration (mg)
Generator Non-Drive End Bearing Vibration (mg)
Oil Particle count
0
100
200
300
400
500
600
1
10
100
1000
10000
100000

Bioinformatics



Doc & patient, *meet* Data

Algorithm 1: Business

while *people have (problems + money)* **do**
    find them;
    if *you can solve their problem* **then**
        convince them;
        solve their problem;
        take their money;
    end
end

(They don't care how)

We use

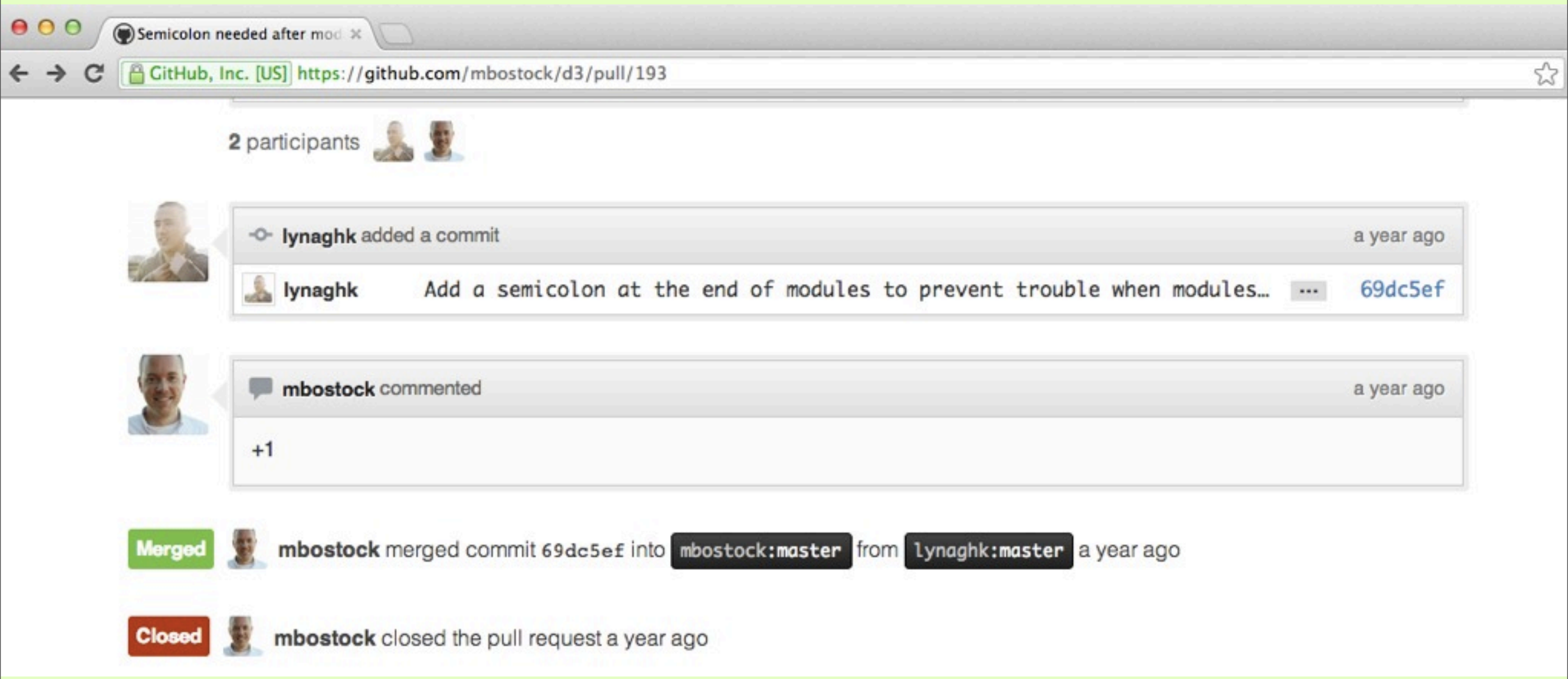
Clojure

We use

The Internets

(❤ Google Chrome in particular)

github is Awesome



Agenda

2

Cool stuff we're working on

Agenda

2

AKA

Research

Bespoke

Photo by **Scott Schuman**,
The Sartorialist

D3: Data Driven Documents (2011)

Mike Bostock Jeffrey Heer Vadim Ogievetsky

$\mathbb{D}^3$: Data-Driven Documents

Michael Bostock, Vadim Ogievetsky and Jeffrey Heer

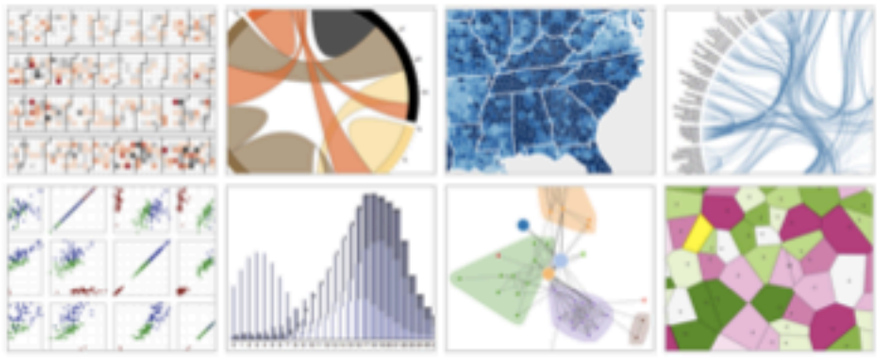


Fig. 1. Interactive visualizations built with D3, running inside Google Chrome. From left to right: calendar view, chord diagram, choropleth map, hierarchical edge bundling, scatterplot matrix, grouped & stacked bars, force-directed graph clusters, Voronoi tessellation.

Abstract—Data-Driven Documents (D3) is a novel representation-transparent approach to visualization for the web. Rather than hide the underlying scenegraph within a toolkit-specific abstraction, D3 enables direct inspection and manipulation of a native representation: the standard document object model (DOM). With D3, designers selectively bind input data to arbitrary document elements, applying dynamic transforms to both generate and modify content. We show how representational transparency improves expressiveness and better integrates with developer tools than prior approaches, while offering comparable notational efficiency and retaining powerful declarative components. Immediate evaluation of operators further simplifies debugging and allows iterative development. Additionally, we demonstrate how D3 transforms naturally enable animation and interaction with dramatic performance improvements over intermediate representations.

Index Terms—Information visualization, user interfaces, toolkits, 2D graphics.

1 Introduction

When building visualizations, designers often employ multiple tools simultaneously. This is particularly true on the web, where interactive visualizations combine varied technologies: HTML for page content, CSS for aesthetics, JavaScript for interaction, SVG for vector graphics, and so on. One of the great successes of the web as a platform is the (mostly) seamless cooperation of such technologies, enabled by a shared representation of the page called the *document object model* (DOM). The DOM exposes the hierarchical structure of page content, such as paragraph and table elements, allowing reference and manipulation. In addition to programming interfaces, modern browsers include powerful graphical tools for developers that display the element tree, reveal inherited style values, and debug interactive scripts.

Unfortunately, this blissful interoperability is typically lost with visualization toolkits due to encapsulation of the DOM with more specialized forms. Rather than empowering direct manipulation of the existing model, such toolkits [2, 9, 18] supplant it with custom scenegraph abstractions. This approach may provide substantial gains in *efficiency*—reducing the effort required to specify a visualization—but it incurs a high opportunity cost: it ignores developers' knowledge of standards, and the tools and resources that augment these standards.

The resulting cost to *accessibility*—the difficulty of learning the representation—may trump efficiency gains, at least for new users. Scarcity of documentation and ineffectual debugging exacerbate the problem, impeding users from gaining deeper understanding of toolkit abstractions and limiting the toolkit's potential. Systems with intermediate scenegraph abstractions and delayed property evaluation can be particularly difficult to debug: internal structures are exposed only when errors arise, often at unexpected times.

Furthermore, intermediate representations may diminish *expressiveness*—the diversity of possible visualizations—and introduce substantial runtime overhead. Certain tasks that could be offloaded to a more suitable tool, such as specifying fonts via CSS, may be stymied by encapsulation. Similarly, while graphical features such as clipping may be supported by the underlying representations, they may not be exposed by the toolkit. Even if extensibility is available as a means for greater expression, it requires in-depth knowledge of toolkit internals and poses a substantial barrier to the average user.

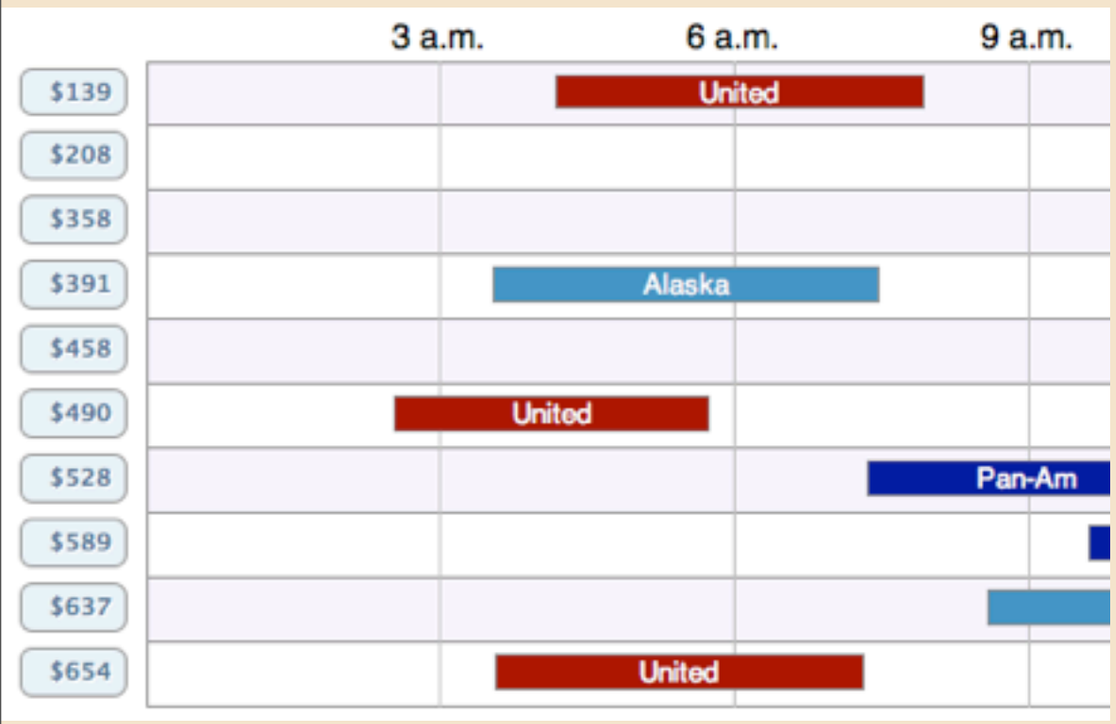
Our awareness of these issues comes in part from thousands of user observations over the two years since releasing Protovis [2], despite our attempt to balance expressiveness, efficiency and accessibility. We now refine these three goals with specific objectives:

Compatibility. Tools do not exist in isolation, but within an ecosystem of related components. Technology reuse utilizes prior knowledge and reference materials, improving accessibility. Offloading a subset of tasks to specialized tools can improve efficiency, avoiding the generality and complexity of a monolithic approach. And, full access to

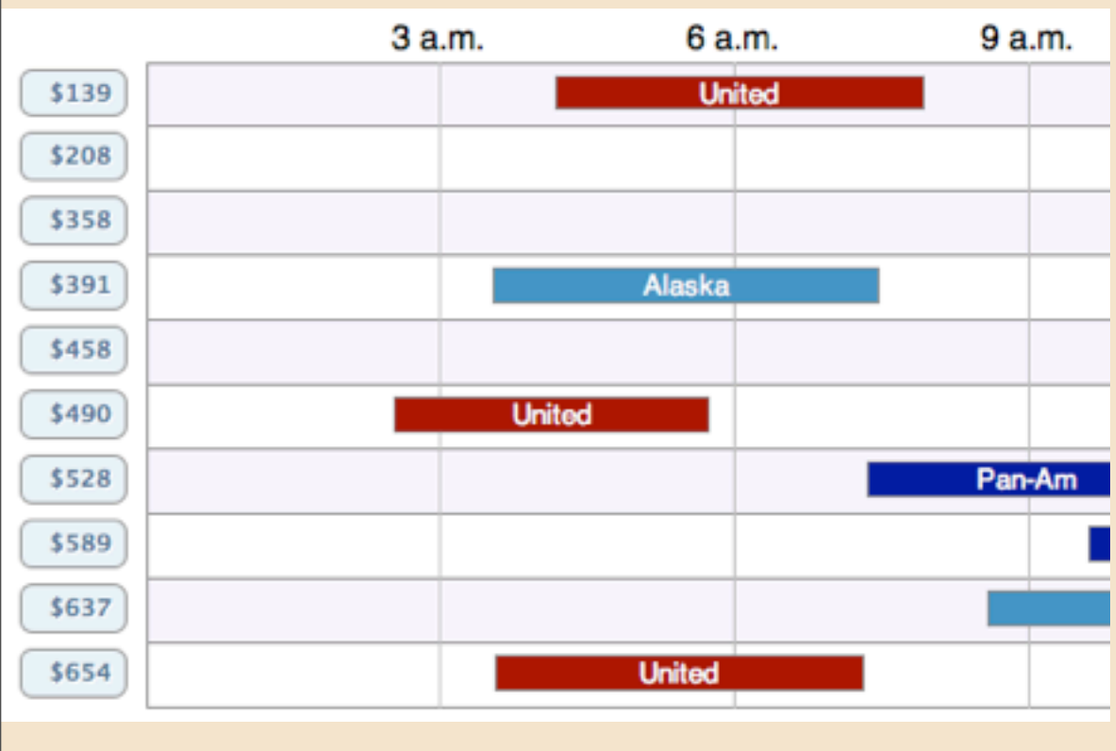
• *The authors are with the Computer Science Department of Stanford University, Stanford, CA 94305.*
Email: {mbostock,vad,jheer}@stanford.edu.

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

Bespoke

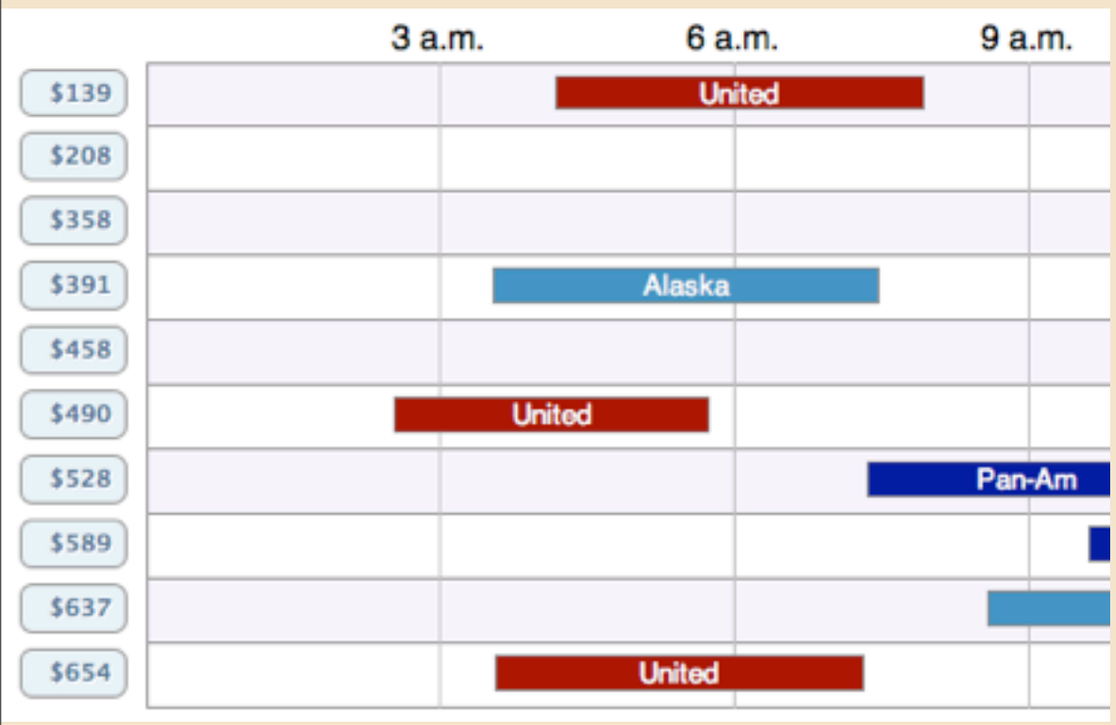

3 a.m.
6 a.m.
9 a.m.
$139
$208
$358
$391
$458
$490
$528
$589
$637
$654
United
Alaska
United
Pan-Am
United

Bespoke



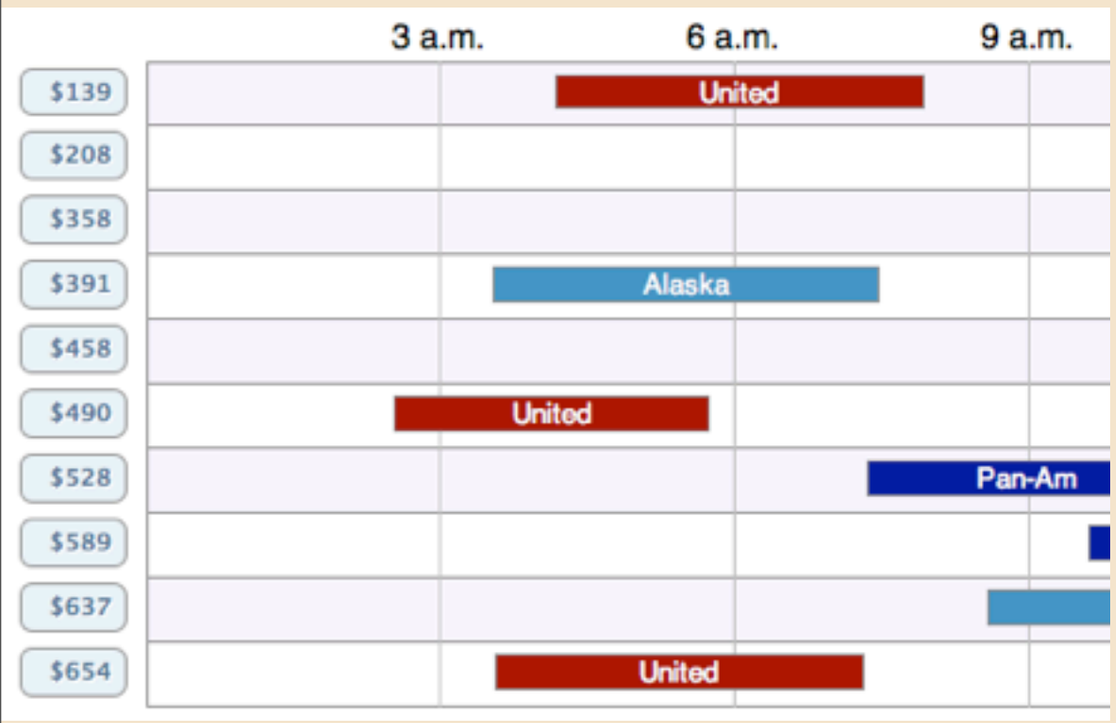

```
(map flight-data
  (fn [{:keys [price carrier depart arrive]}]
      [:div.row
        [:button.price (str "$" price)]
        [:div.flight
          {:style {:left  (time-scale depart)
                   :width (time-scale (- arrive depart))}
           :carrier carrier}
          [:span carrier]]]))
```

Bespoke




```
(map flight-data
  (fn [{:keys [price carrier depart arrive]}]
      [:div.row
        [:button.price (str "$" price)]
        [:div.flight
          {:style {:left  (time-scale depart)
                   :width (time-scale (- arrive depart))}
           :carrier carrier}
          [:span carrier]]]))

(map [[3 "3 a.m."] [6 "6 a.m."] [9 "9 a.m."]
      [12 "Noon"] [15 "3 p.m."] [18 "6 p.m."]]
     (fn [[t label]]
        [:div.tick {:style {:left (time-scale t)}}
          [:div.grid-line]
          [:span.label label]]))
```

Bespoke




```
(map flight-data
  (fn [{:keys [price carrier depart arrive]}]
      [:div.row
        [:button.price (str "$" price)]
        [:div.flight
          {:style {:left  (time-scale depart)
                   :width (time-scale (- arrive depart))}
           :carrier carrier}
          [:span carrier]]]))
```

```
(map [[3 "3 a.m."] [6 "6 a.m."] [9 "9 a.m."]
      [12 "Noon"] [15 "3 p.m."] [18 "6 p.m."]]
  (fn [[t label]]
    [:div.tick {:style {:left (time-scale t)}}
      [:div.grid-line]
      [:span.label label]]))
```

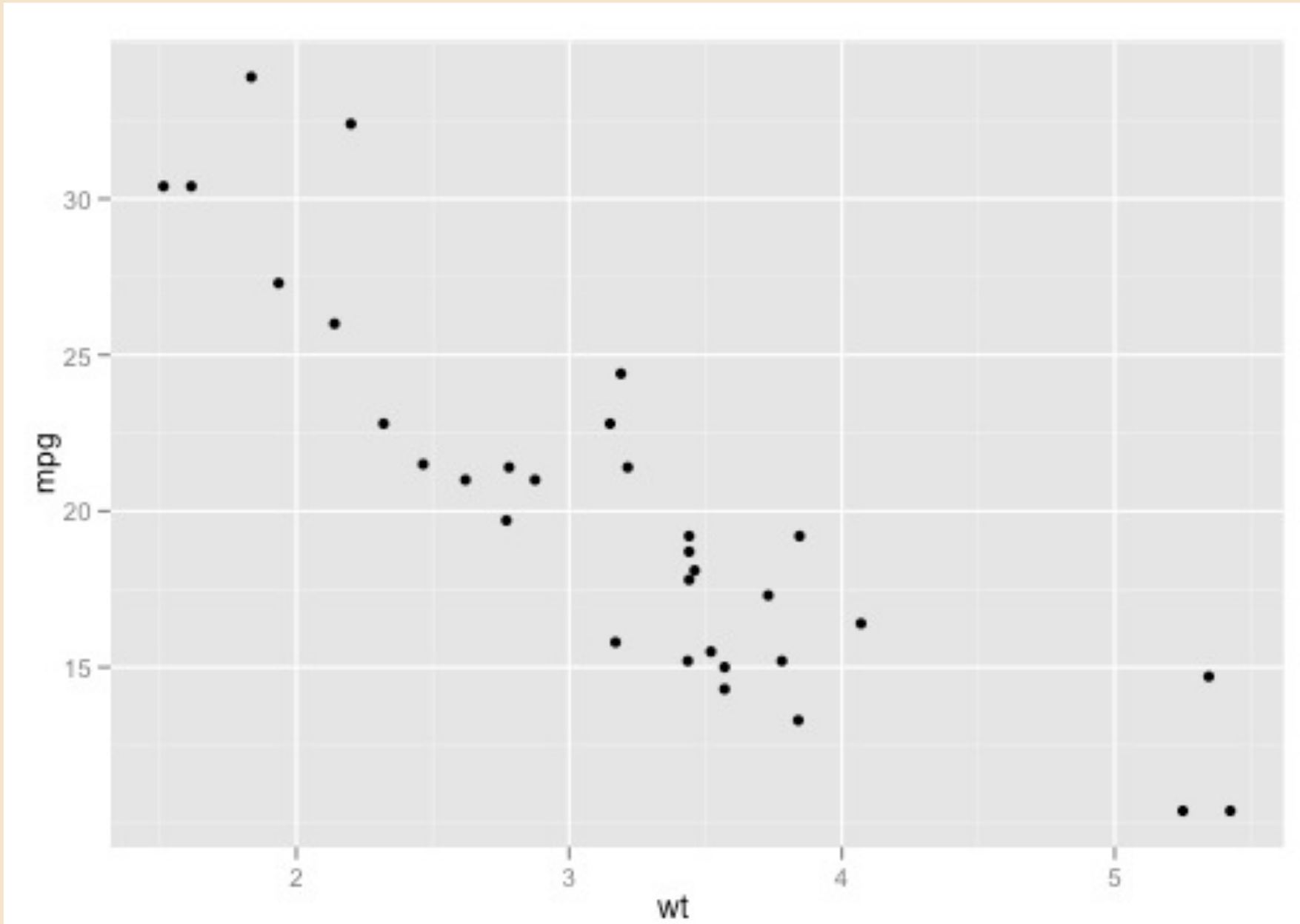
```
#axis
  .tick
    position: absolute
    height: 100%
    .label
      display: block
      position: relative
      top: -1.3em
      left: -2.5em
      width: 5em
      text-align: center
      font-family: sans-serif
    .grid-line
      position: absolute
      width: 1px
      height: 100%
      top: 0px
      background-color: hsl(0,0%,70%)
      z-index: 1
```

```
#flights
  flight
    z-index: 2
    -webkit-transi
ease-out
.price
  position: absolu
  text-align: righ
  width: 50px
  height: 80%
  top: 10%
  left: -60px

.flight
  position: absolu
  height: 60%
  top: 20%
  text-align: cent
  border: 1px soli
  span
    color: white
    font-size: 0.8
    font-family: s

@mixin carrier-col
$color)
  flight[carrier=
  background-col
```

Exploratory

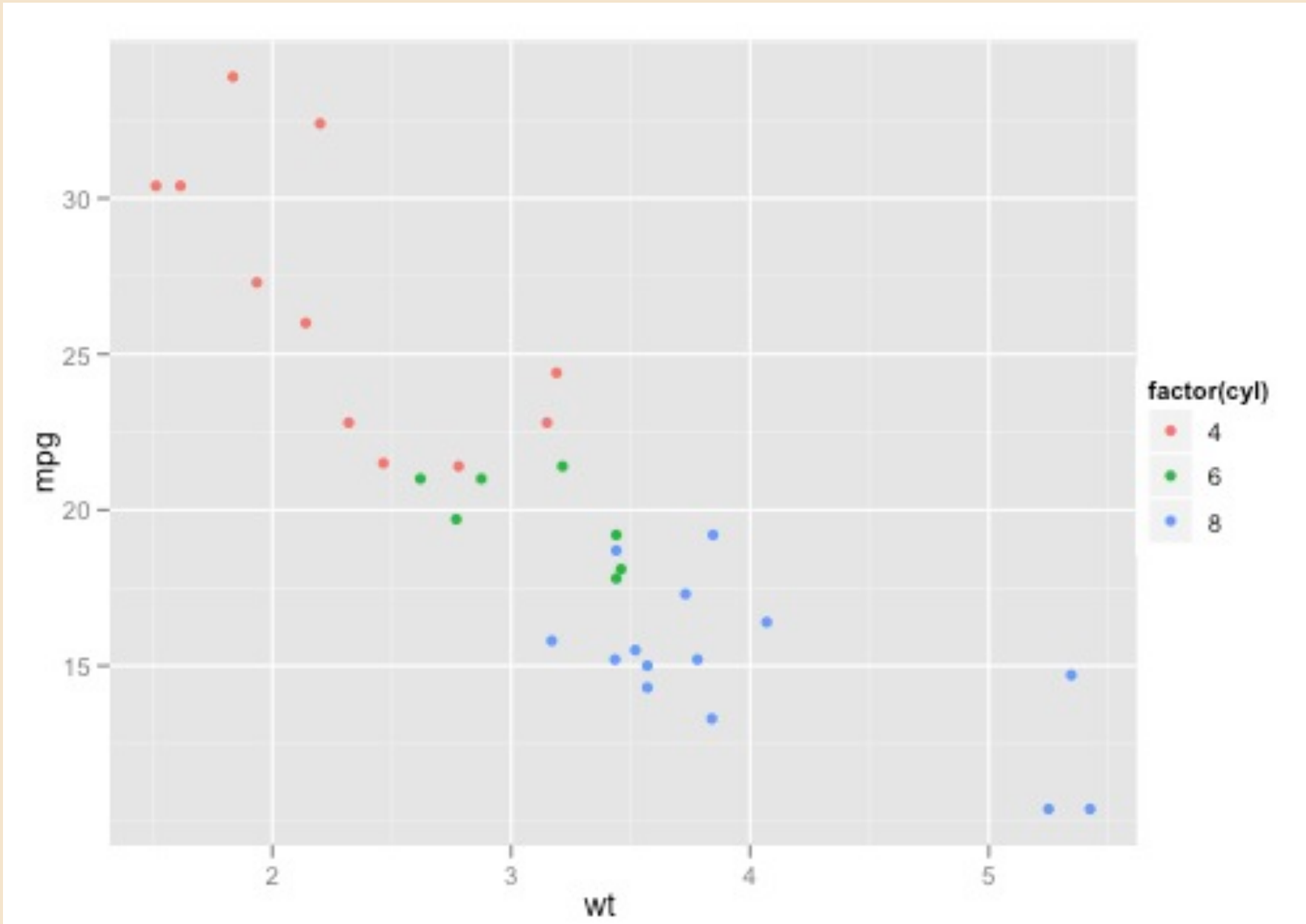
Exploratory

```
ggplot(mtcars, aes(wt, mpg))
  + geom_point()
```



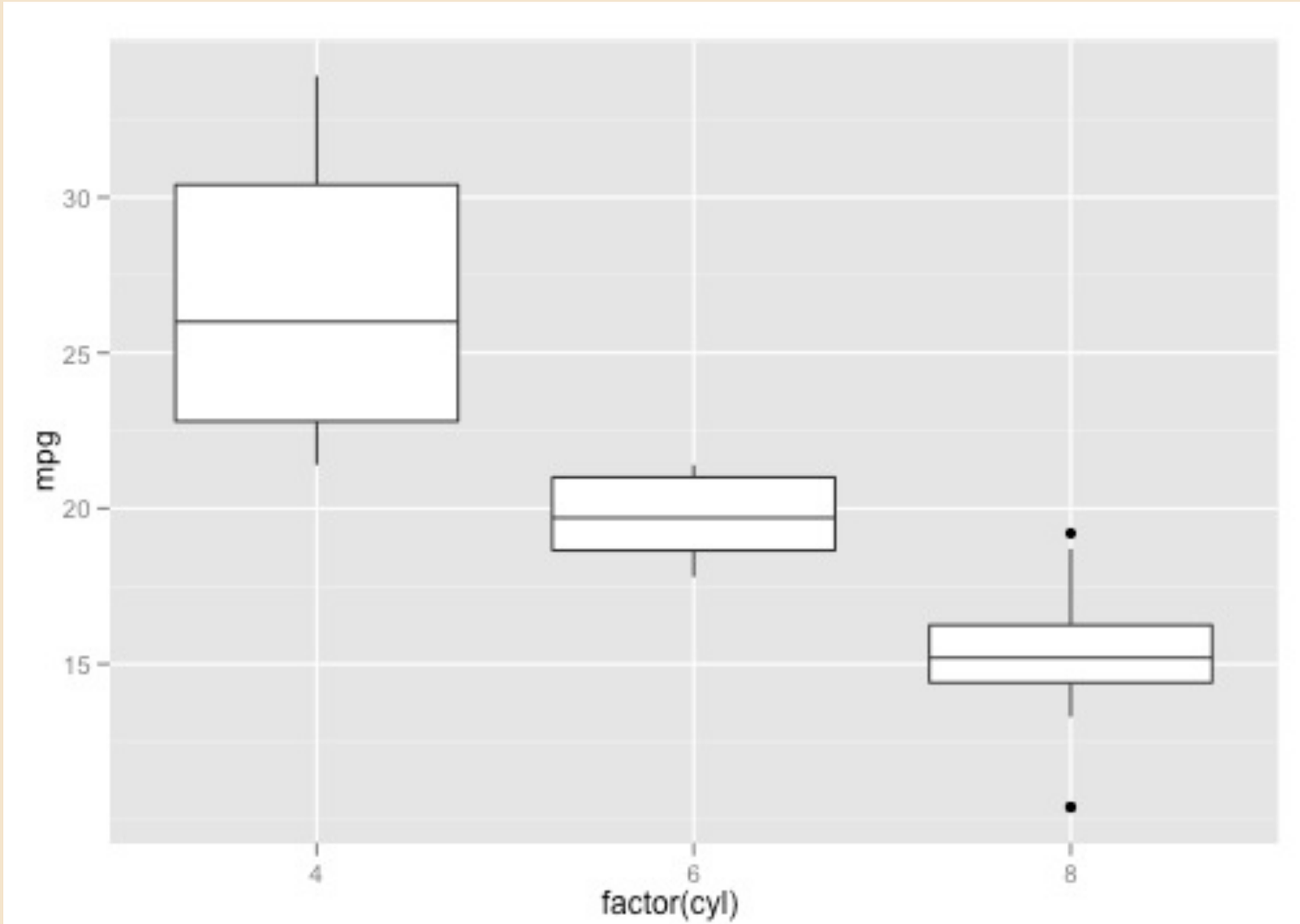
Exploratory

```
ggplot(mtcars, aes(factor(cyl), mpg))
  + geom_point(aes(colour = factor(cyl)))
```



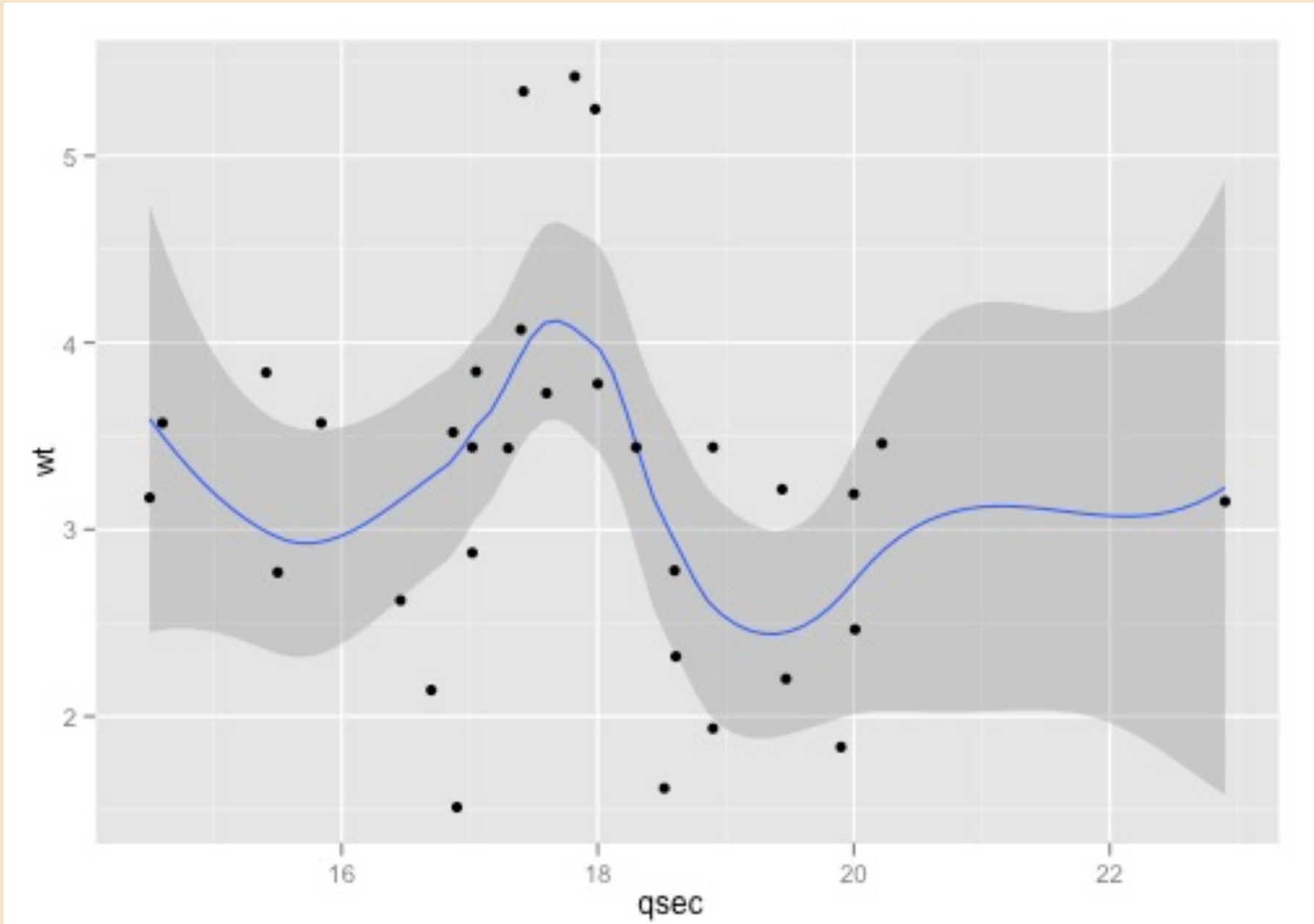
Exploratory

```
ggplot(mtcars, aes(wt, mpg))
  + geom_boxplot()
```



Exploratory

```
ggplot(mtcars, aes(qsec, wt))
  + stat_smooth() + geom_point()
```




Java


python


Scala


jetty://


JRuby


hadoop

API Support

```
{:data mtcars
 :mapping {:x :wt :y :mpg}
 :geom :point}
```

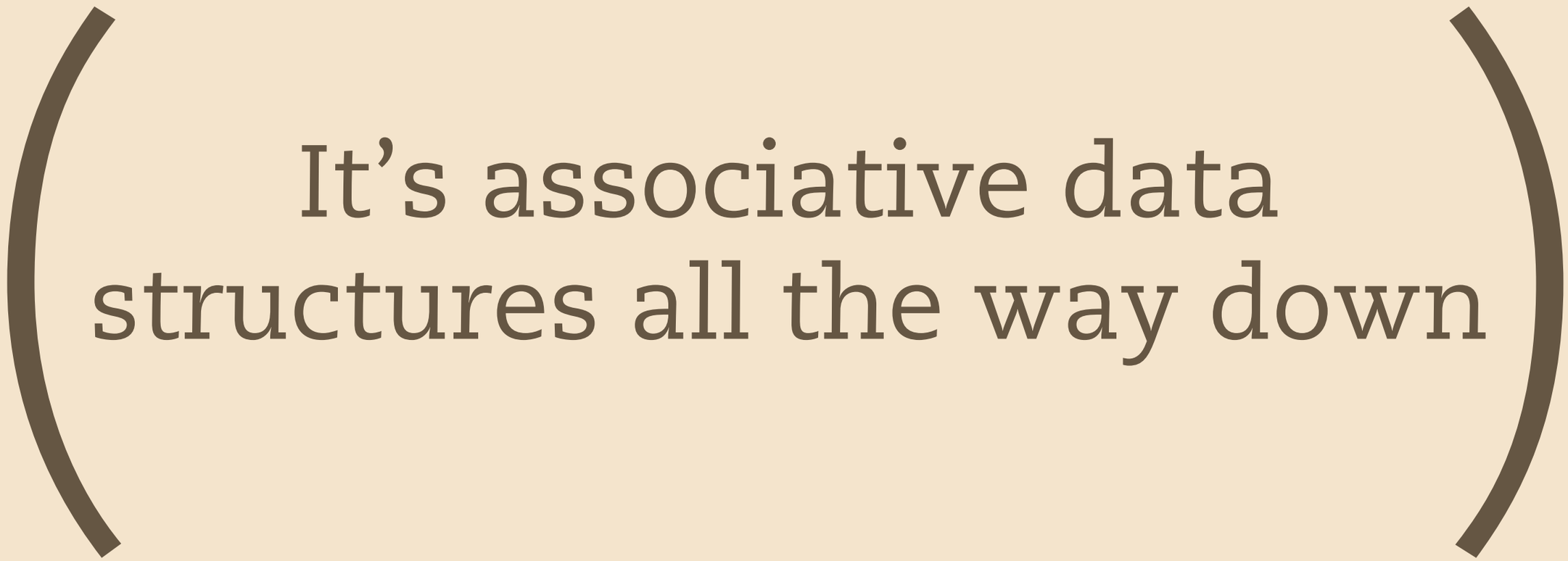
API Support

```
{:data => mtcars,
 :mapping => {:x => :wt, :y => :mpg},
 :geom => :point}
```

API Support

```
{"data": mtcars,
 "mapping": {"x": "wt", "y": "mpg"},
 "geom": "point"}
```

API Support



De-complect

```
{:data diamonds
 :geom :bar
 :mapping {:x (comp first :group) :y :sum
           :width :group/width}
 :group (group/bin :dimension :carat
                   :num-bins 10)
 :stat :sum}
```

Pattern rewrites

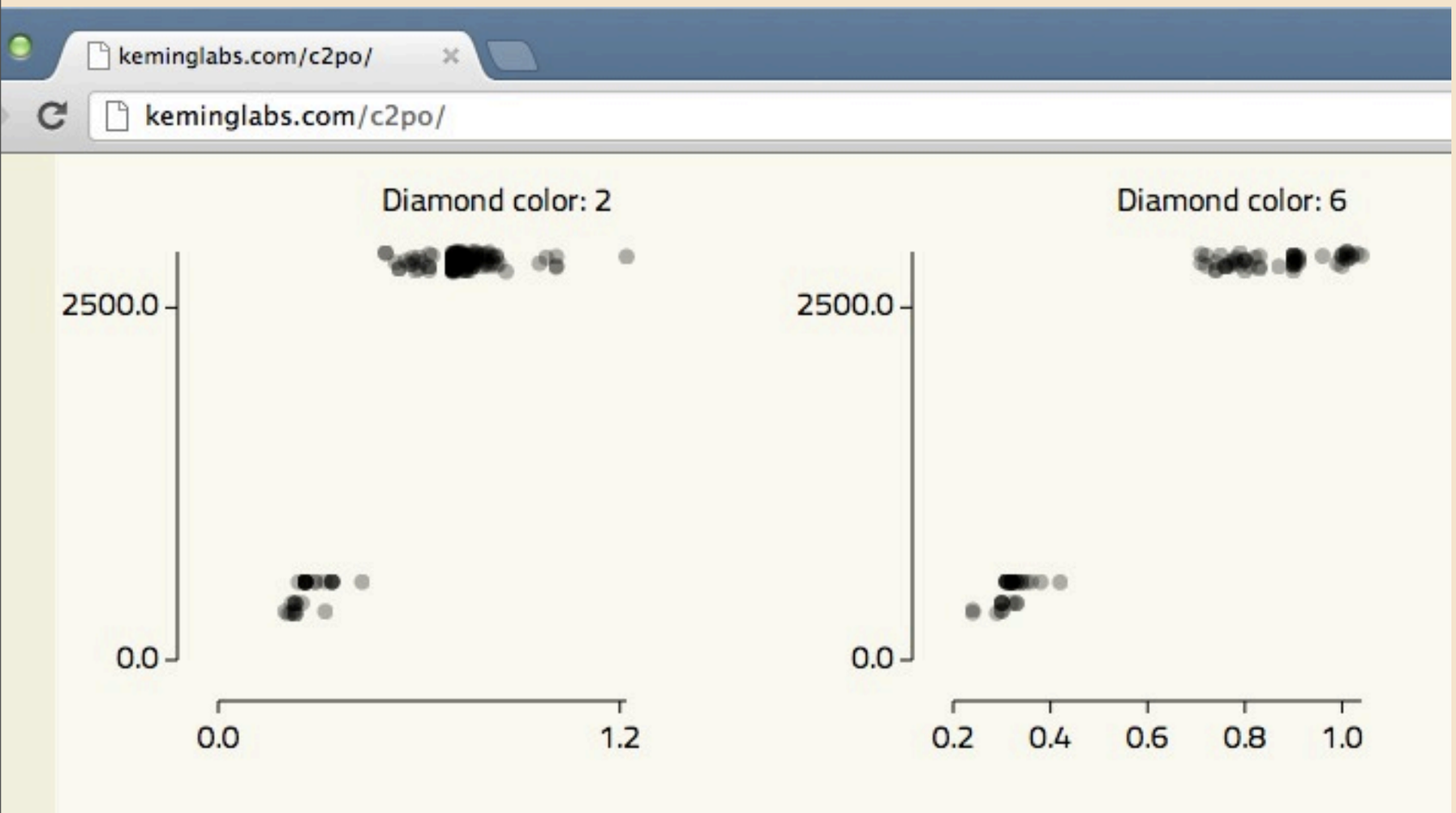
```
{:data diamonds :geom :bar
 :mapping {:x :carat :y :sum}}
```

declarative rules

```
{:data diamonds :geom :bar
 :mapping {:x (comp first :group) :y :sum
           :width :group/width}
 :group (group/bin :dimension :carat
                   :num-bins 10)
 :stat :sum}
```

Recursion

```
{:data diamonds
 :mapping {:title #(str "Diamond color: " (% :group))}
 :group :color
 :geom (geom/plot
         :width 200 :height 200
         :geom (geom/point :radius 4 :opacity 0.3)
         :mapping {:x :carat :y :price})
 :scales {:group :facet}}
```



plot-as-geom idea from unpublished paper by:

Garrett Grolemund
Hadley Wickham

thanks!

Kevin Lynagh

Keming Labs @lynaghk