Interaktive Webgraphiken mit ggvis

von Alexander Lunge

16.7.2016

Proseminar: Programmierung in R

Betreuer: Jakob Lüttgau

Abstract

Im ersten Kapitel beschreibe ich was ggvis überhaupt ist und wofür wir es benutzen können. Danach gehen wir in Kapitel zwei dazu über, wie die Grafiken im Web dargestellt werden und was wir dazu beachten sollten. In Kapitel 3 werden kurz die Bibliotheken beleuchtet, die von ggvis benutzt werden.
Ab Kapitel 4 nähern wir uns den Syntax und den Beispieldatensatz mit einfachen Code Beispielen. Nachdem wir dann mit den Syntax vertraut sind kommen wir zum aussehen der Grafiken in Kapitel 5, wo wir anhand von praktischen Beispielen das Aussehen der Grafiken verändern.
In Kapitel 6 schauen wir uns die Besonderheit von ggvis genauer an, undzwar die Interaktivität. Der Aufbau der Grafiken wird in Kapitel 7 beschrieben und exemplarisch an ein paar Beispielen verdeutlicht.
Kapitel 8 beschäftigt sich wieder mit der Anpassung der Grafiken, hier kommen jedoch kompliziertere Beispiele, da wir dann schon ein besseres Verständnis von ggvis besitzen.
Im Letzten Kapitel 9 wird noch auf die technischen Besonderheiten von den Zuweisungen der Eigenschaften aufmerksam gemacht.

1. Was ist ggvis?

Ggvis ist ein Paket, um Daten zu visualisieren. Dabei können nicht nur statische Grafiken erstellt werden, sondern auch interaktive. Das bedeutet, es lassen sich zum Beispiel Informationen einblenden, wenn man mit der Maus über einen Punkt fährt. Zudem lässt sich auch das Aussehen der Grafiken sowie die Verarbeitung der Daten verändern.

Dabei werden alle Grafiken im Webbrowser dargestellt, was die Plattform übergreifende Programmierung erleichtert. Desweiteren gibt es bereits Bibliotheken, die die Visualisierung der Daten erleichtert, wodurch viel Entwicklungszeit von ggvis gespart werden kann.

Das Ziel von ggvis ist es, das Beste aus R und Web zu verbinden.[1] Das bedeutet, man kombiniert die Darstellungsmöglichkeiten des Webbrowsers mit der schnellen Datenverarbeitung von R.

2. Verbindung zum Web

Sobald wir eine Grafik darstellen wollen, öffnet sich der Standart Webbrowser und zeigt die Grafik an. Was sehr komfortabel ist, da wir nicht erst eine HTML Datei suchen und öffnen müssen.

Es gibt zwei Typen von Grafiken: statische und dynamische.
Bei den statischen handelt es sich um einfache HTML Datein mit den Daten für die Grafik, die wir auch im Internet veröffentlichen können.

Bei den dynamischen handelt es sich um Grafiken mit interaktiven Bedienelementen, wie zum Beispiel ein Schieberegler. Diese benötigen immer einen Webserver im Hintergrund, von dem neue Daten abgefragt werden können. Es wird eine neue Abfrage gestartet, sobald wir ein Bedienelement verändern. R startet diesen Server für uns im Hintergrund, deshalb müssen wir uns nicht um diesen kümmern. Wollen wir jedoch die Grafik im Internet veröffentlichen, müssen wir daran denken, einen Server für die Datenabfragen bereit zu stellen.

3. Plotten im Webbrowser

Ggvis benutzt zwei Bibliotheken: Shiny und Vega.

[1] vgl. http://ggvis.rstudio.com/

Shiny ist ein Web Framework und kümmert sich um den Datenaustausch zwischen R und dem Webbrowser und um das zur Verfügungstellen des HTML codes.

Vega ist zuständig für das Visualisieren der Daten vom JSON Format zu einer Grafik.

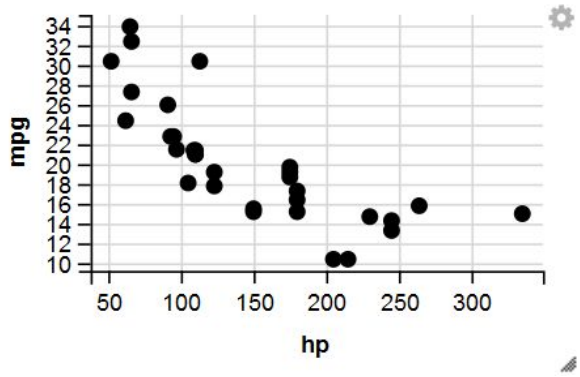
4. Syntax

Der Syntax ist angelehnt an ggplot2. Jedoch sind die Gemeinsamkeiten nur einfach zu erkennen, wenn man mit den Syntax von ggplot2 gut vertraut ist.

Als erstes werden wir für die Beispiele den eingebauten Datensatz “mtcars” verwenden, da dieser sich sehr gut eignet, um sich Beispielgrafiken zu generieren. Mtcars steht für “motor tend cars”, also wie viel Gewicht oder PS ein Auto hat.

	mpg (Miles/Gallon)	cyl (Zylinder)	hp (PS)	wt (Gewicht)
Mazda RX4	21.0	6	110	2.620

Hier sehen wir ein erstes Syntax Beispiel. Als erstes Argument wird mtcars übergeben, also unser Datensatz. Als zweites und drittes Argument wird übergeben, welche Werte aus den Datensatz an welche Achse kommen. Layer_points ist dann dafür zuständig, das Ganze als Grafik darzustellen. Die Tilde vor den Variablennamen gibt an, dass es sich um eine Variable in dem Datensatz handelt und nicht um eine Variable, die vorher im R Code definiert wurde.



Code:[2]

layer_points(ggvis(mtcars, x = ~hp, y = ~mpg)

[2] http://ggvis.rstudio.com/ggvis-basics.html Basierend auf der Quelle, viele Beispiele sind abgeändert

In ggvis verwendet man in der Regel Pipes, um den Code besser lesen zu können. Dabei wird der Wert vor dem %>% als erstes Argument in die Funktion dahinter übergeben.

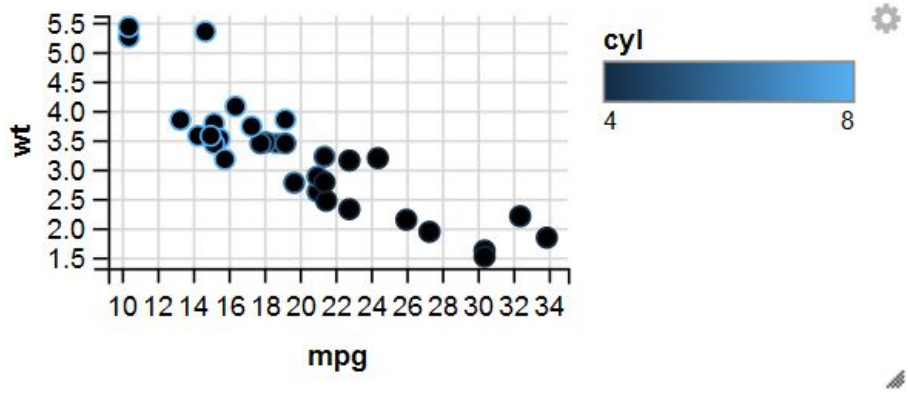
Code:[3]
mtcars %>%
ggvis(~hp, ~mpg) %>%
layer_points()

5. Grundlegenes Anpassen von Graphiken

Oftmals möchte man die Grafiken noch visuell aufbereiten. Dafür gibt es in ggvis zahlreiche Funktionen, um die Grafiken anzupassen. Im Folgenden stelle ich die wichtigsten Eigenschaften vor.

Im nächsten Codebeispiel sehen wir, dass nun ein dritter Parameter hinzugekommen ist. Nach dem Angeben der Variablen für die x- und y-Achse können wir weitere Eigenschaften angeben, um die Grafik noch weiter zu verändern. Hier haben wir nun die Eigenschaft “stroke” hinzugefügt, um den Datenpunkten eine Kontur zu geben. Ggvis erstellt für uns automatisch eine Legende und legt eine Farbe fest, die wir auch noch weiter spezifizieren könnten.

stroke - Kontur
Code:[4]
mtcars %>% ggvis(~mpg, ~wt, stroke = ~cyl) %>% layer_points()



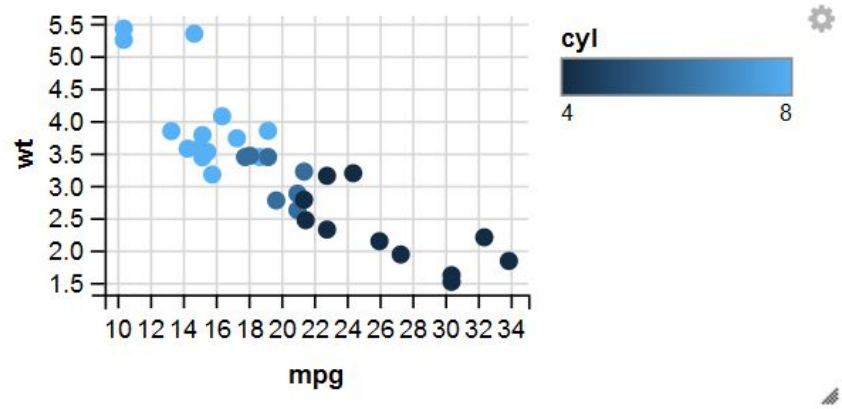
[3] http://ggvis.rstudio.com/cookbook.html
[4] http://ggvis.rstudio.com/ggvis-basics.html

In diesen Beispiel ersetzen wir nun die “stroke” Eigenschaft durch “fill”. Damit können wir die ganzen Punkte einfärben. Wir sehen nun schon, dass es mit ggvis sehr einfach ist, verschiedene Eigenschaften zu benutzen.

fill - Ganze Farbe

Code:[5]

mtcars %>% ggvis(~mpg, ~wt, fill = ~cyl) %>% layer_points()

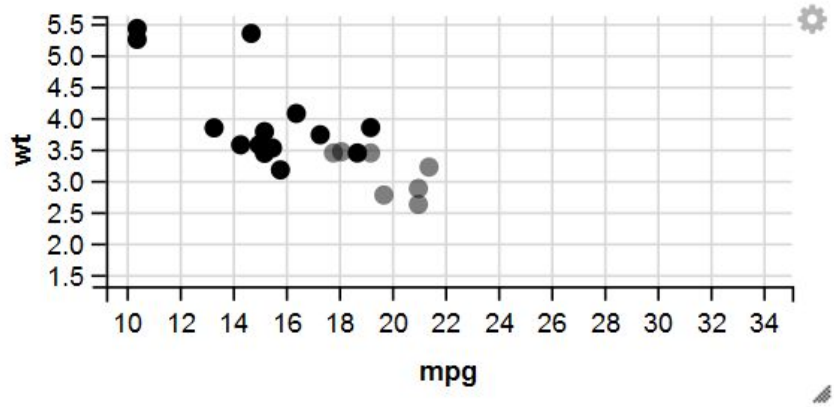


Es eignet sich jedoch nicht jede Eigenschaft, wie in diesem Beispiel. Hier verwenden wir die “opacity” Eigenschaft, um die Punkte durchsichtig zu machen. Jedoch sind die Punkte mit den wenigsten Zylindern komplett durchsichtig, wodurch sie nicht mehr sichtbar sind .

opacity - Durchsichtigkeit

Code:[6]

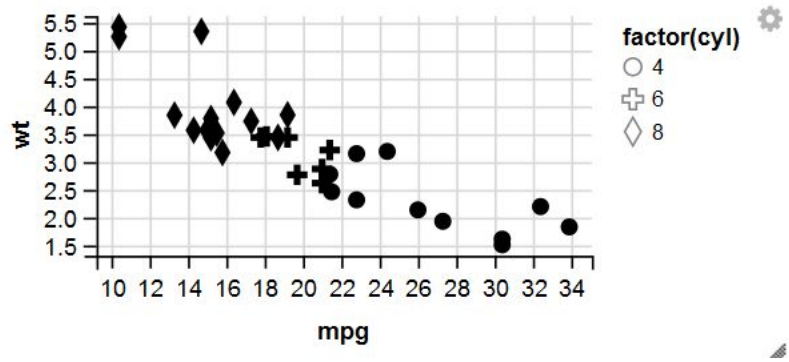
mtcars %>% ggvis(~mpg, ~wt, opacity = ~cyl) %>% layer_points()



[5] http://ggvis.rstudio.com/ggvis-basics.html
[6] http://ggvis.rstudio.com/ggvis-basics.html

Bei der “shape” Eigenschaft gibt es eine Besonderheit. Hier müssen wir die “factor” Funktion verwenden. Diese geht über alle Variablen eines Vektors und gibt uns ein Vektor mit den unterschiedlichen Werten, wodurch wir keine doppelten Werte mehr im Vektor haben. Dadurch bekommt jedes Fahrzeug ein unterschiedliches Symbol, falls sie eine unterschiedliche Anzahl an Zylindern haben.

shape - Form
Code:[7]
mtcars %>% ggvis(~mpg, ~wt, shape = ~factor(cyl)) %>% layer_points()

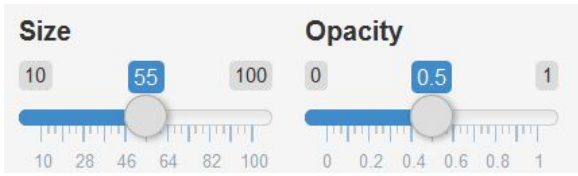


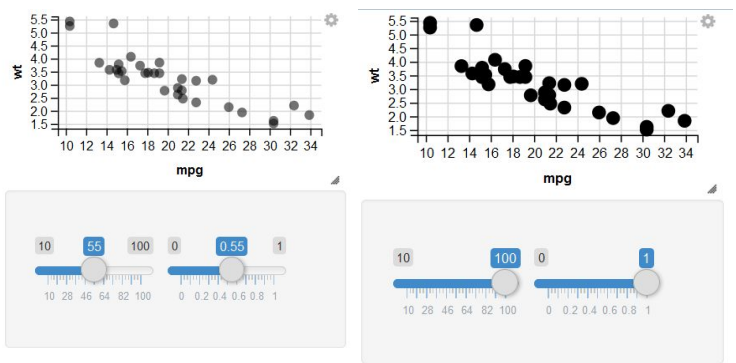
6. Interaktive Graphiken

Sobald wir eine interaktive Grafik öffnen, wird ein Server im Hintergrund gestartet, was auch daran zu erkennen ist, dass die Kommandozeile von R blockiert bis wir mit ESC den Server stoppen. Danach kann sich unsere Grafik auch nicht weiter verändern.

Hier sehen wir ein Beispiel. Wir haben wieder unseren vertrauten Datensatz, nun aber mit zwei Schiebereglern. Diesen können wir mit dem Parameter Label auch noch einen Namen geben, um zu wissen, was dieses Element verändert.
Code:[8]

```
mtcars %>%
  ggvis(~mpg, ~wt,
    size := input_slider(10, 100, label = “Size”),
    opacity := input_slider(0, 1, label = “Opacity”)
  ) %>%
  layer_points()
```





[7] http://ggvis.rstudio.com/ggvis-basics.html
[8] http://ggvis.rstudio.com/interactivity.html

7. Ebenen (Layers)

Alle Grafiken in ggvis sind in Ebenen aufgebaut. Dies ermöglicht uns verschiedene Darstellungsweisen übereinander zu legen. Es gibt zwei verschiedene Typen von Ebenen: Simple und Compound.

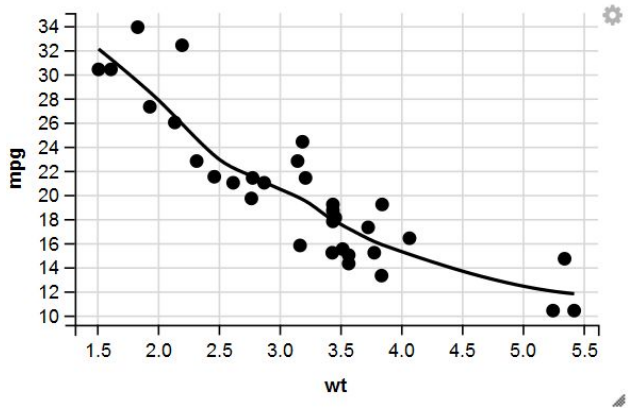
Bei den Simple Layers handelt es sich um Ebenen, die die Daten direkt auf eine Funktion von Vega abbilden. Das bedeutet, die Daten werden nicht vorher verändert.

Bei den Compound Layers handelt es sich um Ebenen, die die Daten vorher manipulieren. So wird zum Beispiel bei der “layer_smooths” Funktion ein Pfad gebildet, der vor der Darstellung noch geglättet wird.

Hier sehen wir ein Beispiel mit der “layer_smooths” Funktion.

Code:[9]

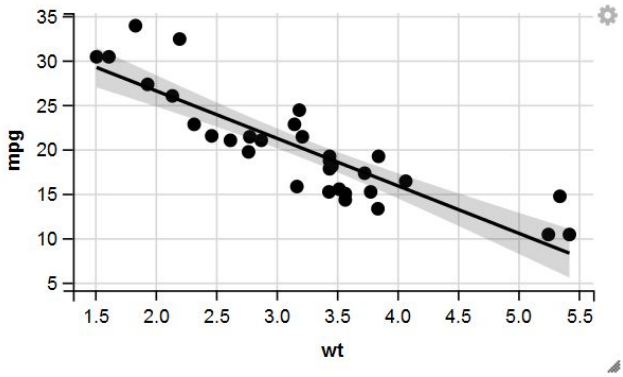
```
mtcars %>%
ggvis(~wt, ~mpg) %>%
layer_points() %>%
layer_smooths()
```



Ein weiteres Beispiel ist die “layer_model_predictions” Funktion bei der eine Trendlinie eingezeichnet wird. Als ersten Parameter haben wir “lm” benutzt, um eine lineare Trendlinie zu erhalten. Der zweite Parameter ist auf true gesetzt, wodurch wir in Grau hinterlegt die Abweichung zum Trend angezeigt bekommen.

Code:[10]

```
mtcars %>%
  ggvis(~wt, ~mpg) %>%
  layer_points() %>%
  layer_model_predictions(model = "lm",
se = TRUE)
```

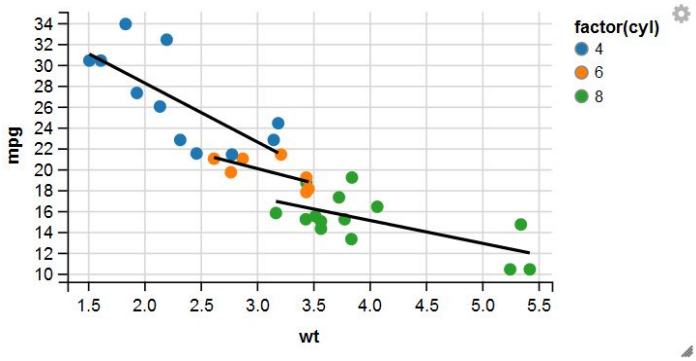


[9] http://ggvis.rstudio.com/cookbook.html
[10] http://ggvis.rstudio.com/cookbook.html

Wir können und auch mehrere Trendlinien einzeichnen lassen, indem wir den Datensatz vorher gruppieren. Dies geschieht mit der “group_by” Funktion. Dadurch erhalten wir drei verschiedene Gruppen in diesem Beispiel, wobei jede Gruppe eine eigene Trendlinie erhält. Desweiteren haben wir in diesem Beispiel wieder die “fill” Eigenschaft verwendet, jedoch in Kombination mit der “factor” Funktion. Deshalb bekommen wir keinen Farbverlauf, wie am Anfang gezeigt, sondern gut zu unterscheidende Farben.

Code:[11]

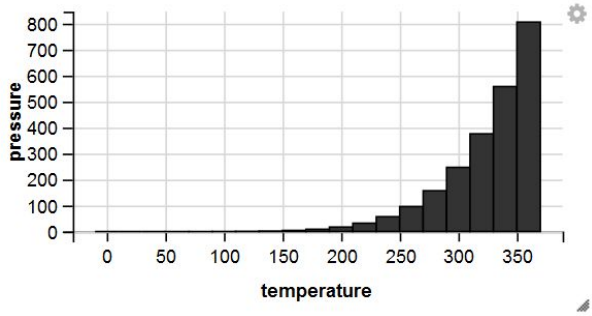
```
mtcars %>%
  ggvis(~wt, ~mpg, fill = ~factor(cyl)) %>%
  layer_points() %>%
  group_by(cyl) %>%
  layer_model_predictions(model = "lm")
```



In dem letzten Beispiel für die Ebenen habe ich einen anderen Datensatz verwendet, da sich dieser sehr viel besser für die “layer_bars” Funktion eignet. Dieses Beispiel verdeutlicht nochmal, dass wir durch einfaches Ändern der Ebene sehr unterschiedliche Grafiken erzeugen können.

Code:[12]

```
pressure %>%
  ggvis(~temperature, ~pressure) %>%
  layer_bars()
```



[11] http://ggvis.rstudio.com/cookbook.html
[12] http://ggvis.rstudio.com/cookbook.html

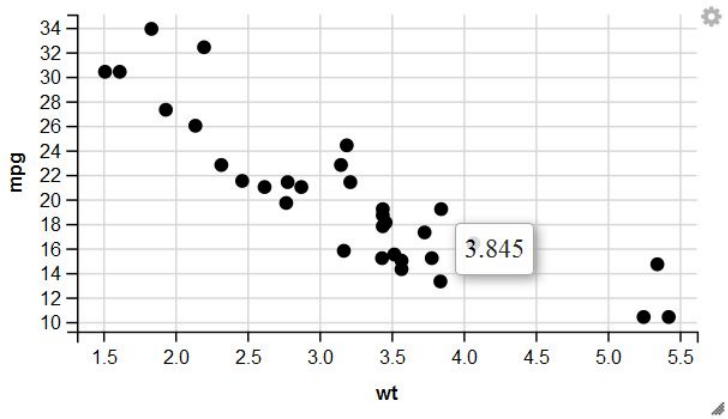
8. Erweitertes Anpassen von Graphiken

Mit ggvis lässt sich noch sehr viel mehr als nur die Farbe und Form von Grafiken verändern. Im folgenden möchte ich ein paar Beispiele zu diesen erweiterten Anpassungen zeigen.

Es lassen sich sehr einfach tooltips hinzufügen. Dazu fügen wir nach der layer Funktion einfach die “add_tooltip” Funktion ein. Als ersten Parameter wird eine Funktion akzeptiert, die als ersten Parameter den Datenpunkt bekommt über den die Maus gefahren ist. Als Rückgabewert greifen wir über das $ Symbol auf die Variablen des Datenpunktes zu und geben in diesen Fall das Gewicht des Autos zurück.

Code:[13]

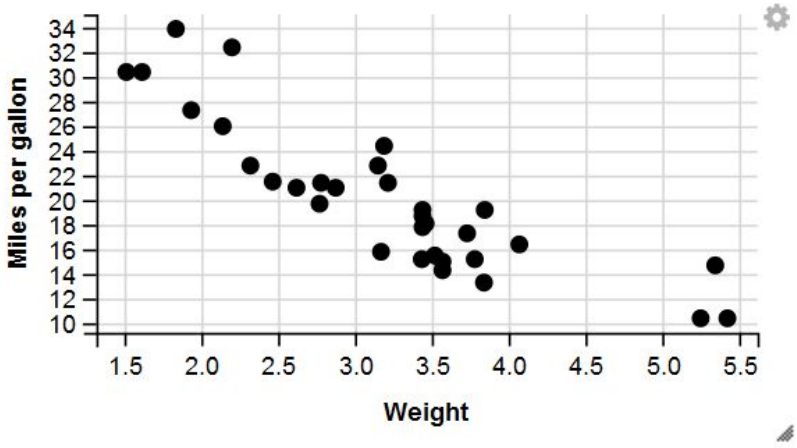
```
mtcars %>% ggvis(~wt, ~mpg) %>%
  layer_points() %>%
  add_tooltip(function(df) df$wt)
```



Falls man die Grafik im Internet veröffentlichen möchte, dann bietet es sich an, die Achsen zu beschriften. Dies erfolgt sehr einfach mit der “add_axis” Funktion. Als ersten Parameter wird hier die Achse übergeben und als zweiten Parameter übergeben wir den Titel. Man könnte auch noch weitere Eigenschaften an die Funktion übergeben, um weiter Anpassungen an der Achsenbeschriftung vorzunehmen.

Code:[14]

```
mtcars %>% ggvis(~wt, ~mpg) %>%
layer_points() %>%
  add_axis("x", title = "Weight") %>%
  add_axis("y", title = "Miles per
gallon")
```

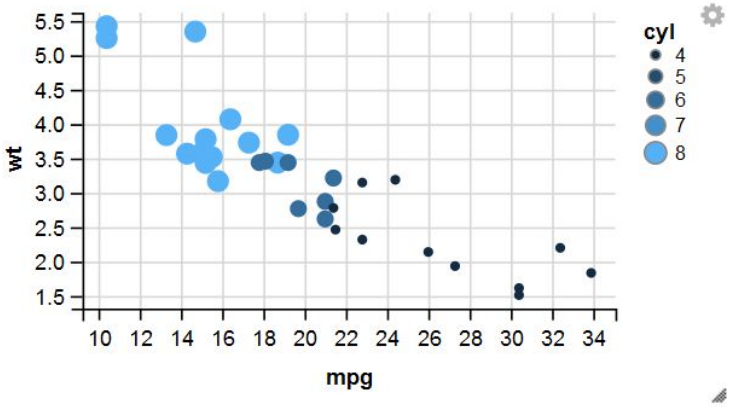


[13] http://ggvis.rstudio.com/ggvis-basics.html
[14] http://ggvis.rstudio.com/axes-legends.html

In den letzten Beispiel möchte ich noch eine Besonderheit vorstellen. Bei der “add_legend” Funktion ist es möglich, verschiedene Legenden zu kombinieren. Würden wir die Funktion hier nicht verwenden, so bekämen wir zwei Legenden , einmal für die Größe und einmal für die Farbe. Mithilfe der “c” Funktion erstellen wir einen Vektor, wodurch wir die Legenden “size” und “fill” kombinieren und wir nun nur noch eine Legende haben mit Kreisen verschiedener Größe und Farbe.

Code:[15]

```
mtcars %>%
ggvis(~mpg, ~wt, size = ~cyl, fill = ~cyl)
%>%
layer_points() %>%
add_legend(c("size", "fill"))
```



9. Skalierung und Eigenschaften

Eine Besonderheit, die einem nicht sofort auffällt, ist die Zuweisung der Eigenschaften. Eigenschaften sind alle Parameter, die zum Beispiel an die “ggvis” oder an die Layer Funktionen übergeben werden, somit sind auch “x” und “y” Eigenschaften.
Wird eine Eigenschaft über ein “=” Zeichen zugewiesen bedeutet das, dass der Wert skaliert wird, also verschiedene Werte für verschiedene Datenpunkte annimmt.
“=” -> Skalierung

Code: [16]

```
mtcars %>%
ggvis(x = ~hp, y = ~mpg) %>%
layer_points()
```

Wird eine Eigenschaft über ein “:=” Zeichen zugewiesen bedeutet das, dass der Wert für alle Datenpunkte der Gleiche ist. So lassen sich zum Beispiel alle Punkte der gleichen Farbe zuordnen oder wie in diesen Beispiel den Input Sildern.
“:=” -> Konstanter Wert

Code: [17]

```
mtcars %>% ggvis(~mpg, ~wt,
size := input_slider(10, 100, label = "Size"),
opacity := input_slider(0, 1, label = "Opacity") ) %>%
layer_points()
```

[15] http://ggvis.rstudio.com/axes-legends.html
[16] http://ggvis.rstudio.com/cookbook.html
[17] http://ggvis.rstudio.com/interactivity.html

Zusammenfassung

Zusammenfassend stellen wir fest, dass ggvis ein sehr flexibles Paket zum Plotten für R ist. Es lassen sich einfach Grafiken erstellen und verändern. Desweiteren erledigt es die komplexe Arbeit unsere Daten in einen Webbrowser darzustellen und bietet uns eine einheitliche Schnittstelle für statische und dynamische Grafiken.

Der Syntax ist übersichtlich und überzeugt durch seine Austauschbarkeit der Funktionen, wie wir es bei den Ebenen gesehen haben. Das Aussehen der Grafiken ist leicht anspassbar und ermöglicht es mit wenig aufwand Interaktive grafiken zu erstellen und veröffentlichen.

Als Fazit finde ich, dass ggvis einen großen Vorteil gegenüber gerenderter Bilder besitzt und einfach zur lernen ist, da sich Grafiken durch zum Beispiel durch Tooltips übersichtlicher gestalten lassen und der Nutzer mehr Freiheiten dadurch bekommt.

Literatur

http://www.r-statistics.com/2014/08/simpler-r-coding-with-pipes-the-present-and-future-of-the-magrittr-package/

http://ggvis.rstudio.com/

Mtcars Erläuterung - https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html

https://cran.r-project.org/web/packages/ggvis/ggvis.pdf