Grafiken in R mit ggplot2

Benjamin Schlegel

29. Februar 2016

ggplot2 ist eine mächtige Bibliothek zum Erstellen von Grafiken in R. Dieser Artikel gibt eine Einführung in ggplot2. Er deckt aber bei weitem nicht alle Möglichkeiten dieser Bibliothek ab. Der Artikel setzt voraus, dass man die wichtigsten Elemente von R schon beherrscht. Wer noch nie mit R gearbeitet hat, kann zuerst den Artikel R Grundlagen lesen.

In einem ersten Schritt wird ein Zufallsdatensatz erstellt, der nachher für die Beispiele gebraucht wird. Das verstehen des folgenden Codes ist keine Voraussetzung, um ggplot2 zu verstehen.

```
c=c(rep(1,10),rep(2,90),rep(3,50),rep(4,30),rep(5,120), rep(6,80),
    rep(7,160),rep(8,60),rep(9, 150),rep(10,40),rep(11,110),rep(12,100))
c=sample(c, 1000)
data1 = data.frame(a=1:1000,b=runif(1000,0, 100), c=c)
data1$d = ifelse(data1$b%%2==0,"good","bad")
```

Als nächstes wird die Bibliothek geladen.

```
library(ggplot2)
```

Eine Grafik mit ggplot2 fängt immer mit der Funktion ggplot() an. Dieser Funktion wird der Datensatz übergeben und optional globale Werte, welche für die ganze Grafik gelten.

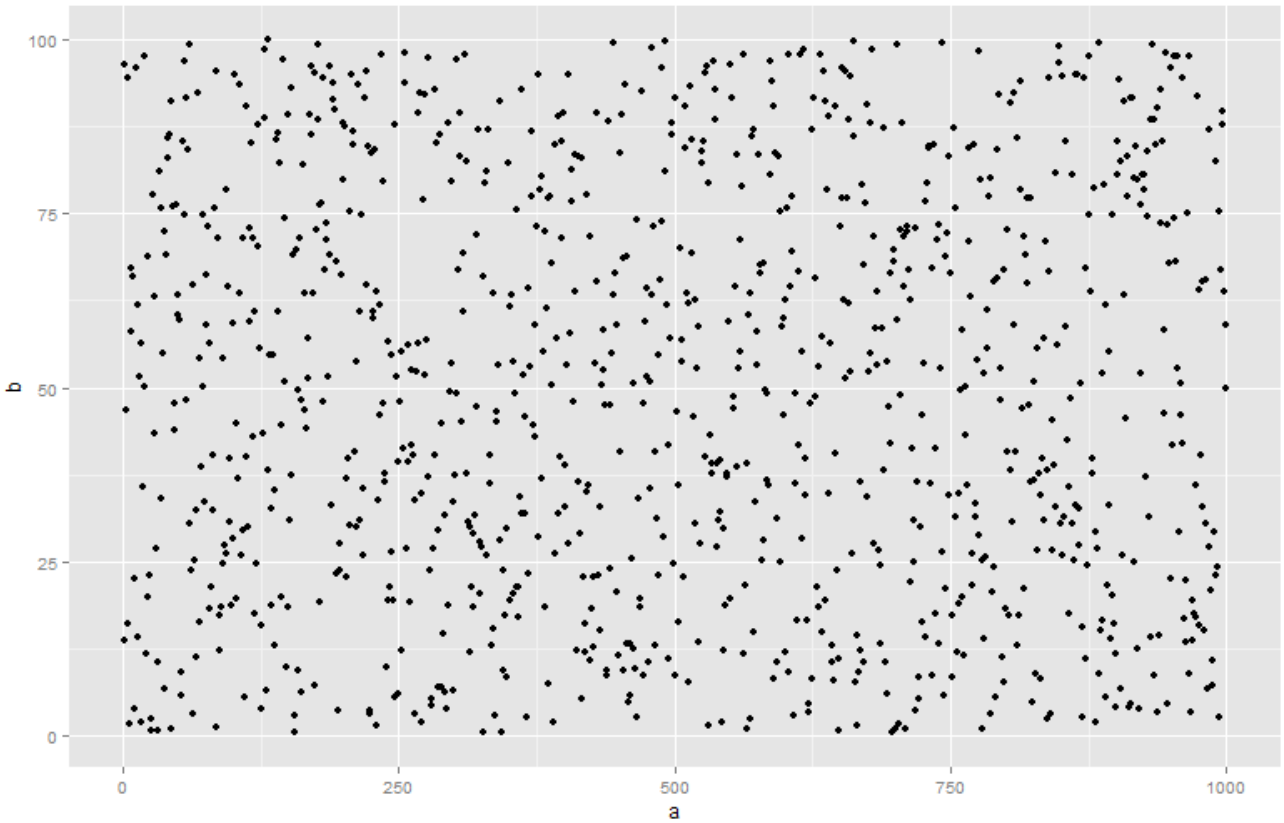
```
p = ggplot(data1)
```

Die Grafik wird in der Variable p gespeichert. Die Grafik kann jedoch noch nicht gezeichnet werden. Um eine Grafik zu produzieren, braucht ggplot2 mindestens einen Layer. Erst dann bekommt man eine Grafik. Ohne Layer bekommt man die Fehlermeldung Error: No layers in plot.

Punkte

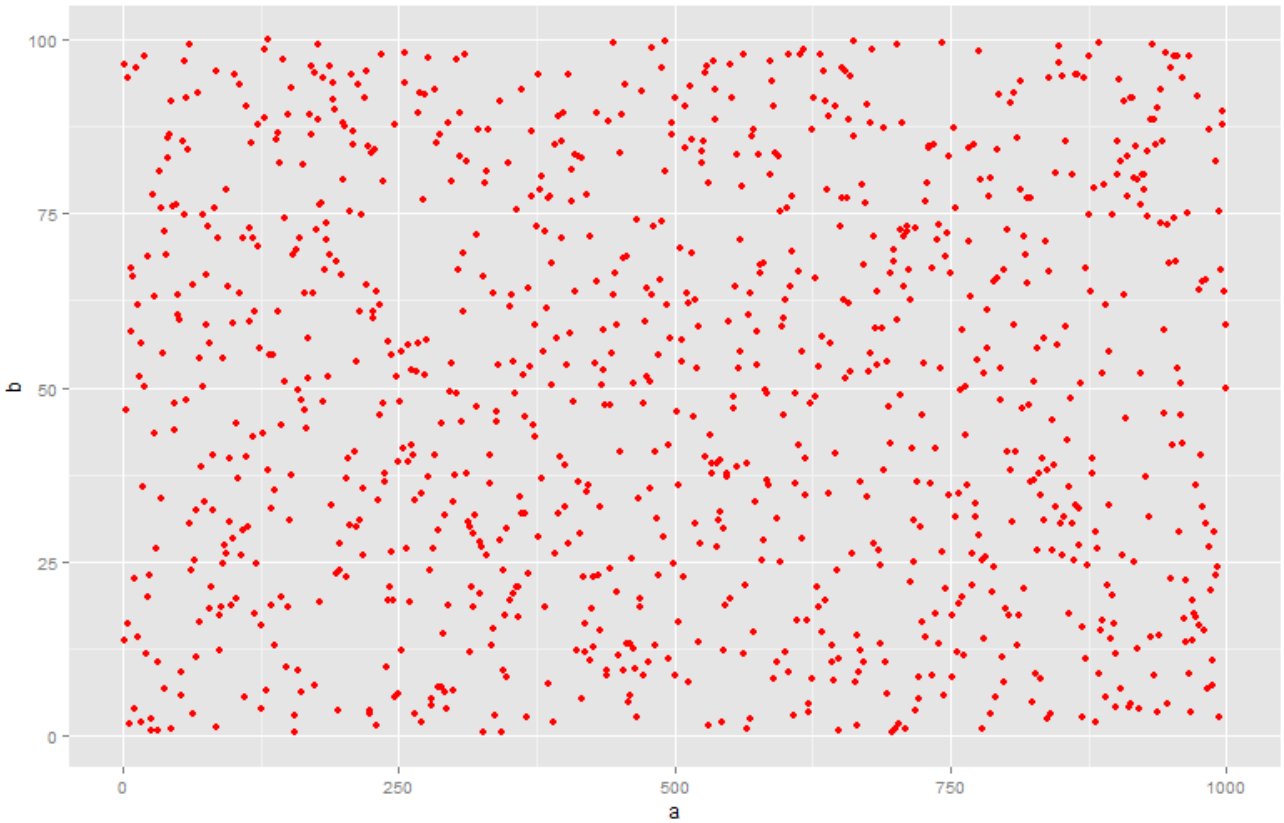
Will man eine Grafik mit Punkten, kann der Layer geom_point() verwendet werden. Die Funktion braucht ein x und ein y. Diese kann entweder direkt in der Funktion übergeben werden oder kann global aus dem ggplot() stammen. Der Unterschied ist, dass die Werte in der Funktion geom_point() nur für diesen Layer gelten und Werte in der Funktion ggplot() für alle Layer gelten, sollte mehr als einer definiert werden. Die Layers und weitere Elemente werden mit + hinzugefügt.

```
p + geom_point(aes(x=a,y=b))
```



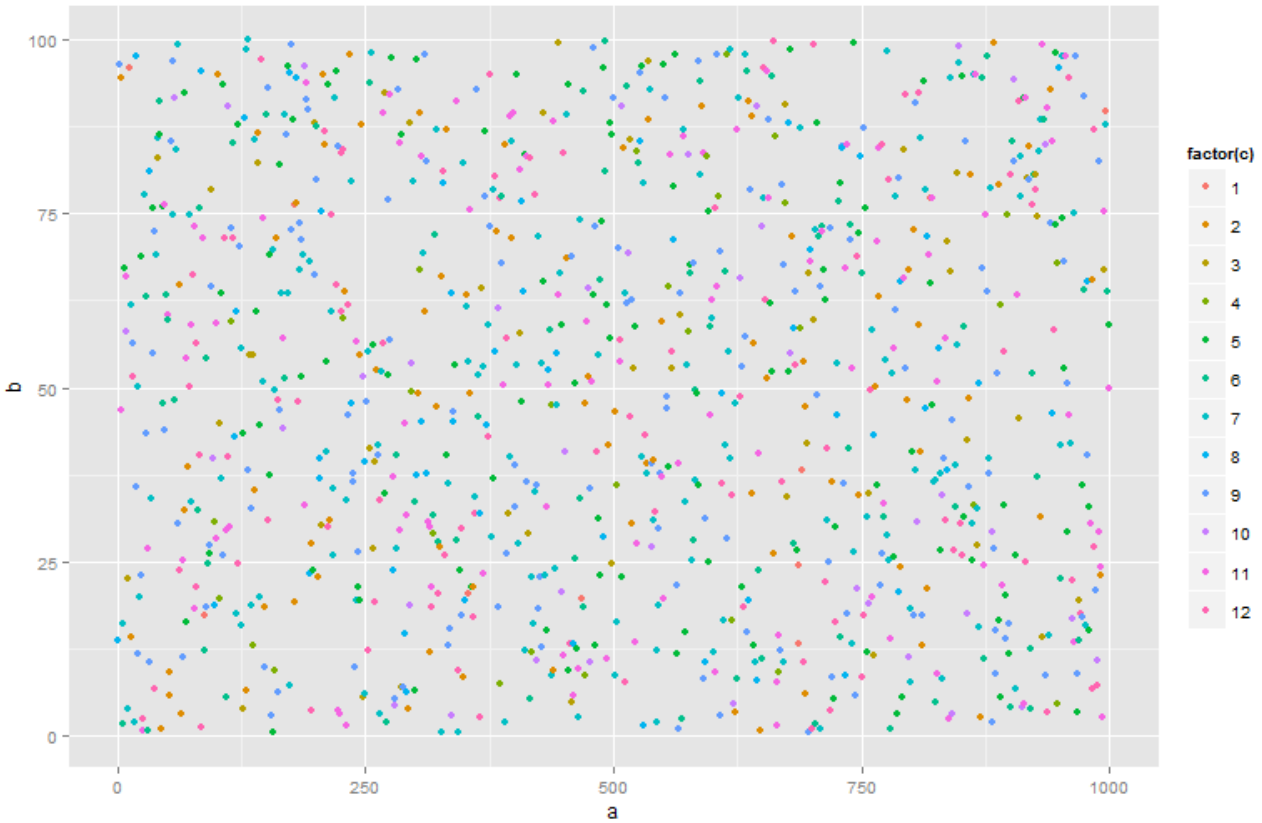
Die Werte für x und sind in der Funktion aes() definiert worden. In aes() werden alle Werte definiert, welche vom Datensatz abhängen. Ausserhalb dieser Funktion werden hingegen alle Werte definiert, welche unabhängig vom Datensatz sind. So können zum Beispiel alle Punkte rot gefärbt werden.

```
p + geom_point(color="red",aes(x=a,y=b))
```



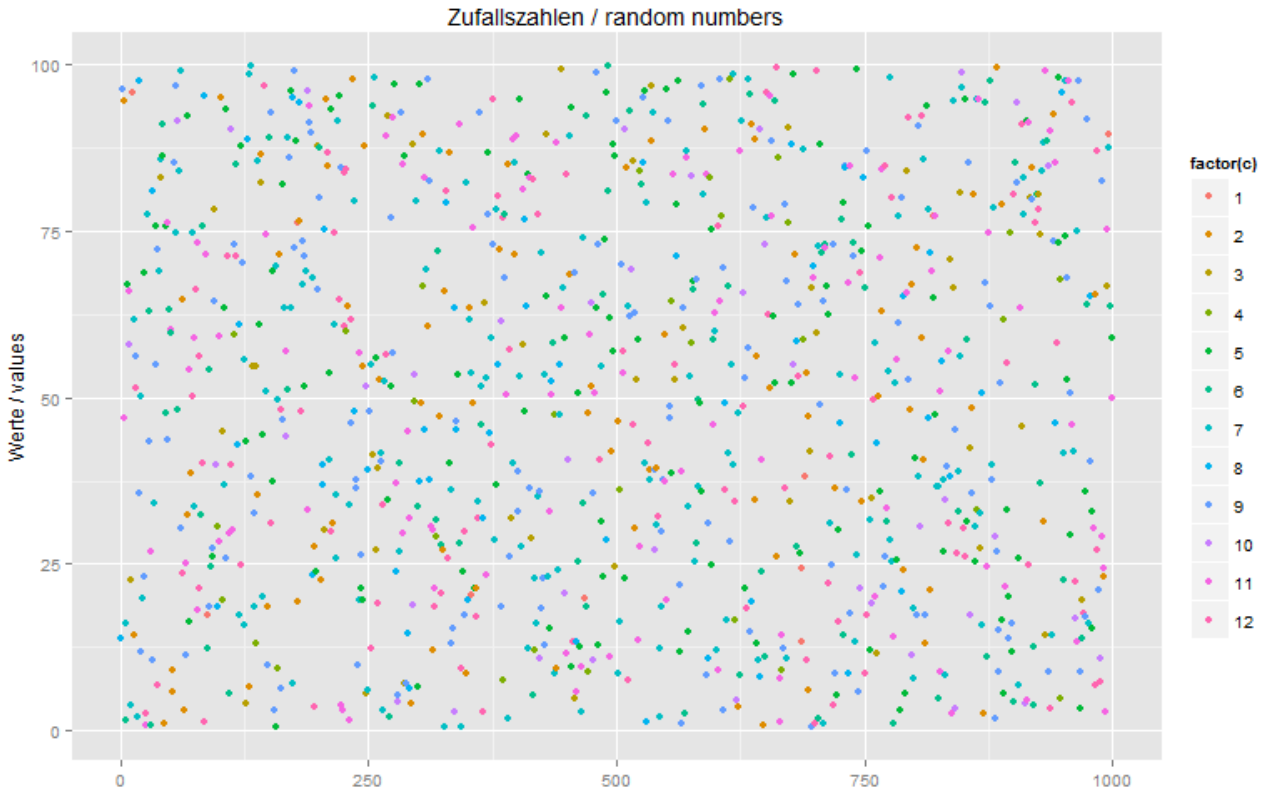
Will man hingegen die Farben nach spezifischen Werten im Datensatz definieren, so schreibt man es innerhalb von aes().

```
p + geom_point(aes(x=a,y=b,color=factor(c)))
```



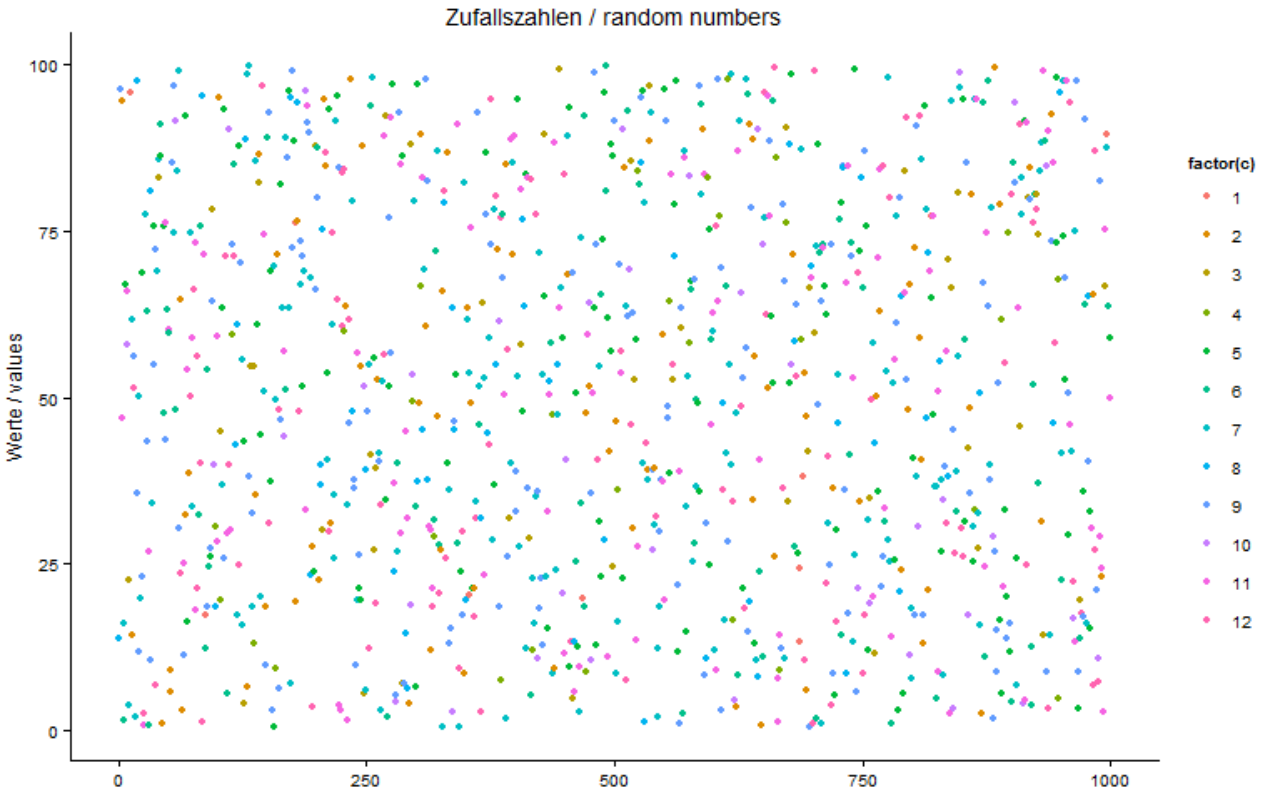
Nun sieht die Grafik noch nicht sehr schön aus. Im nächsten Schritt werden die Achsen (xlab() und ylab()) beschriftet und ein Titel gesetzt (ggtitle()).

```
p + geom_point(aes(x=a,y=b,color=factor(c))) +
  ylab("Werte / values") + xlab("") + ggtitle("Zufallszahlen / random numbers")
```



Das Standard Layout ist nicht besonders schön. Besser sind die Layouts theme_bw() und theme_classic().

```
p + geom_point(aes(x=a,y=b,color=factor(c))) +
  ylab("Werte / values") + xlab("") + ggtitle("Zufallszahlen / random numbers") +
  theme_classic()
```

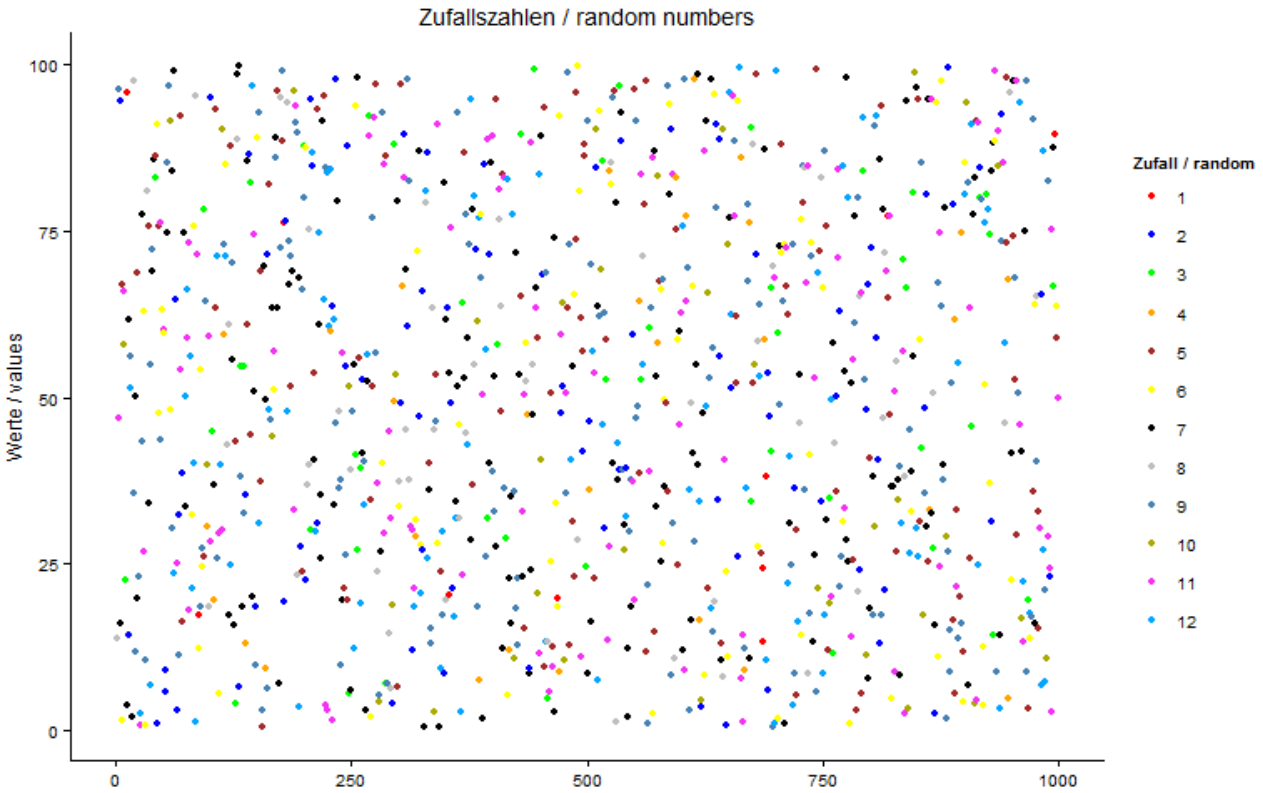


Im nächsten Schritt bekommt die Legende einen richtigen Titel mit scale_color_discrete(). Wäre die Variable, welche die Farben definiert stetig, so verwendet man die Funktion scale_color_continuous().

```
p + geom_point(aes(x=a,y=b,color=factor(c))) +
  ylab("Werte / values") + xlab("") + ggtitle("Zufallszahlen / random numbers") +
  theme_classic()
```

Mit der Funktion scale_color_manual können die Farben definiert werden. Die wird grundsätzlich Empfohlen, da die Standardfarben zu ähnlich sind.

```
p + geom_point(aes(x=a,y=b,color=factor(c))) +
  ylab("Werte / values") + xlab("") +
  ggtitle("Zufallszahlen / random numbers") +  theme_classic() +
  scale_color_manual(values=c("red","blue","green","orange",
    "brown","yellow","black","grey","steelblue","#aaaa00",
    "#f333f3","#05a5ff"), name="Zufall / random")
```



Linien

Wie der Punkte-Layer, braucht auch der Linien-Layer (geom_line()) Werte für x und y.

```
ggplot(subset(data1,a
```