Bootcamp 7: Extend discrete color palettes

Lucien Baumgartner

11/6/2018

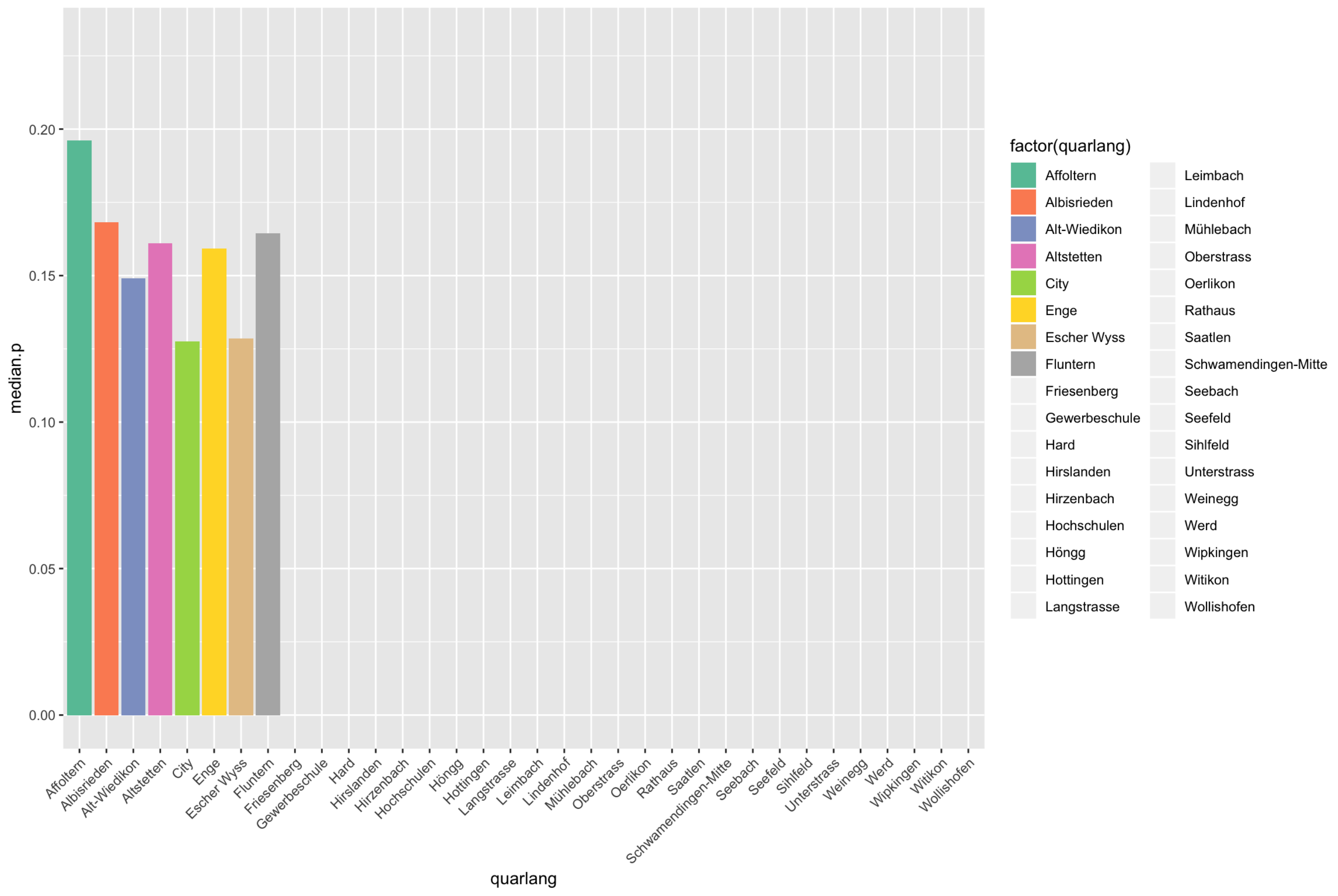
1 Problem

```
library(gplots)
library(ggplot2)
library(dplyr)
library(RColorBrewer)
library(viridis)
library(extrafont) # different fonts

rm(list=ls())

source('~/r-helpers/ggplot/ggplot-helper.R')

# set WD
setwd('~/ddj18/output/')

# load pop data
load('01-bevoelkerung-clean.RData')
```

If we plot the data without extending the color palette, we end up with the following result:

```
# aggregate data
kids <- df %>% mutate(has.kids=ifelse(anzahlkinder>0,1,0)) %>%
  filter(stichtagdatjahr%in%2012:2017) %>%
  group_by(quarlang, stichtagdatjahr, has.kids) %>%
  summarise(n=n()) %>%
  mutate(p=n/sum(n)) %>%
  ungroup %>%
  group_by(quarlang, has.kids) %>%
  summarise(median.p=median(p))

# simple bar plot
kids %>%
  filter(has.kids==1) %>%
  ggplot() +
  geom_bar(aes(quarlang, median.p, fill=factor(quarlang)), stat='identity')
  scale_fill_brewer(palette="Set2") +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```

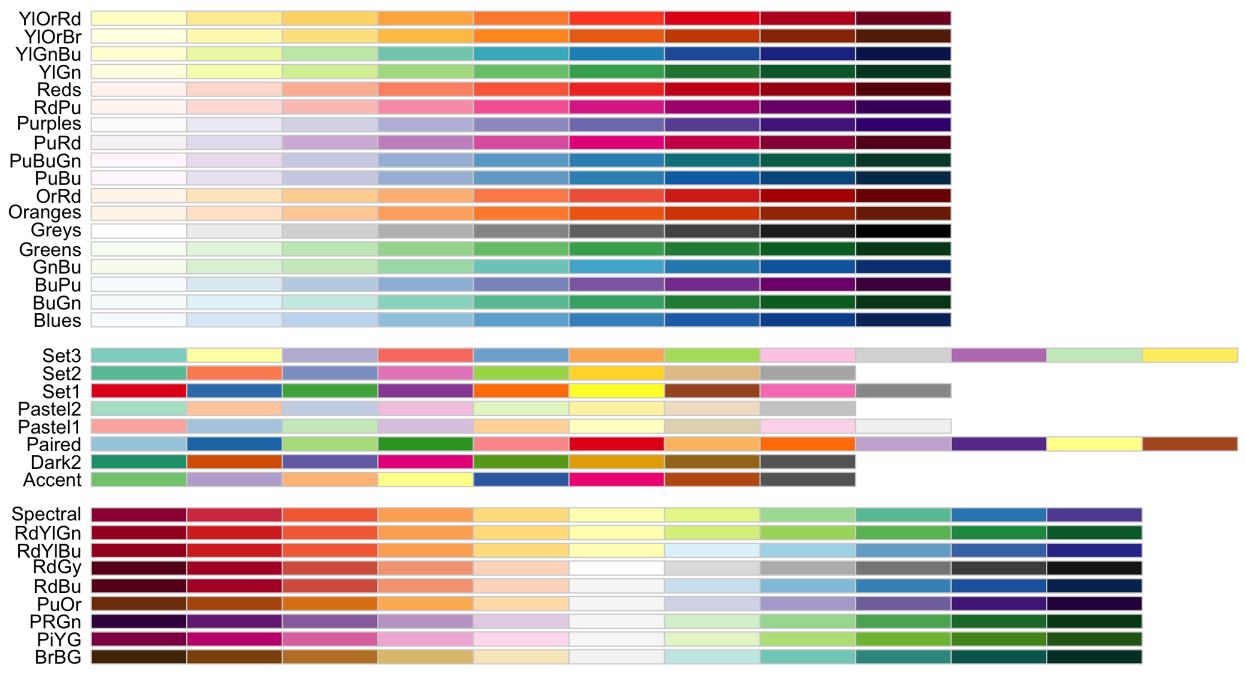
```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximu
## Returning the palette you asked for with that many colors
```



2 Brewer palettes

Now, to extend the color palette of your choice, you don't need anything else than the palette (dah), and the `colorRampPalette()` function native to the graphic devices.

```
# brewer palettes
display.brewer.all()
```



```
# define the number of discrete colors you need (== factor levels of the fi
c.count <- length(unique(kids$quarlang))

# feed the palette to the function to create a pre-parametrized function
get.palette <- colorRampPalette(brewer.pal(9, "Set1"))

# function
get.palette
```

```
## function (n)
## {
##     x <- ramp(seq.int(0, 1, length.out = n))
##     if (ncol(x) == 4L)
##         rgb(x[, 1L], x[, 2L], x[, 3L], x[, 4L], maxColorValue = 255)
##     else rgb(x[, 1L], x[, 2L], x[, 3L], maxColorValue = 255)
## }
## <bytecode: 0x7fec446dd638>
## <environment: 0x7fec446da2e8>
```

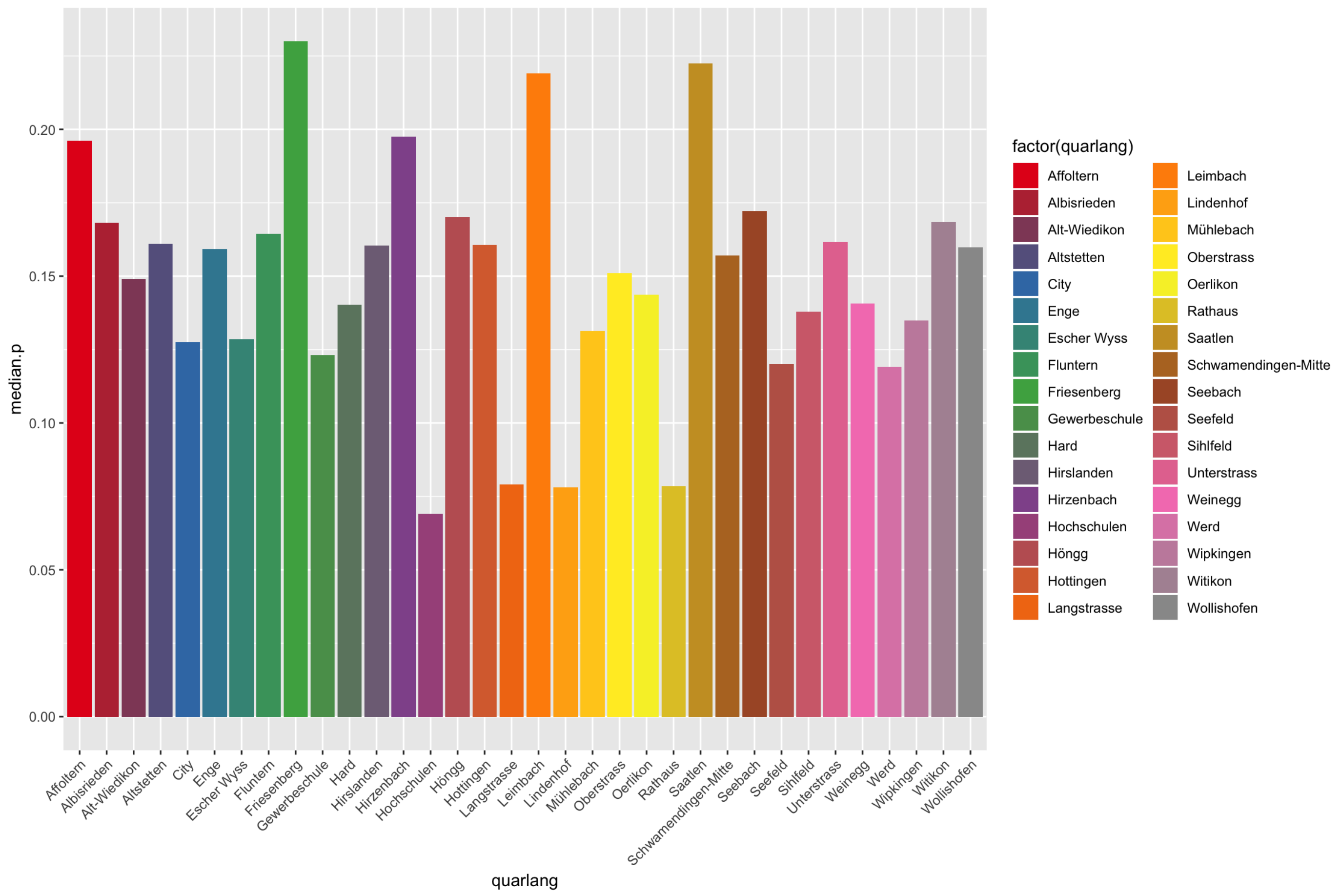
```
# function fed with the umber of colours needed
get.palette(c.count)
```

```
##  [1] "#E41A1C" "#BA3241" "#904A67" "#66628D" "#3C7AB3" "#3B88A0" "#41948
##  [8] "#46A06B" "#4BAC50" "#5A9D5A" "#6C856F" "#7F6E85" "#91569A" "#A7558
## [15] "#C06162" "#D96D3B" "#F27913" "#FF8E06" "#FFAD12" "#FFCC1E" "#FFEB2
## [22] "#F6EF32" "#E1C62F" "#CB9D2C" "#B6742A" "#AA5831" "#BE6355" "#D26D7
## [29] "#E5779E" "#F481BD" "#DD87B4" "#C68DAB" "#AF93A2" "#999999"
```

```
# is everything of the same length?
length(get.palette(c.count))==c.count&c.count==length(unique(kids$quarlang)
```

```
## [1] TRUE
```

```
kids %>%
  filter(has.kids==1) %>%
  ggplot() +
  geom_bar(aes(quarlang, median.p, fill=factor(quarlang)), stat='identity')
  theme(legend.position="right") +
  scale_fill_manual(values=get.palette(c.count))  +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



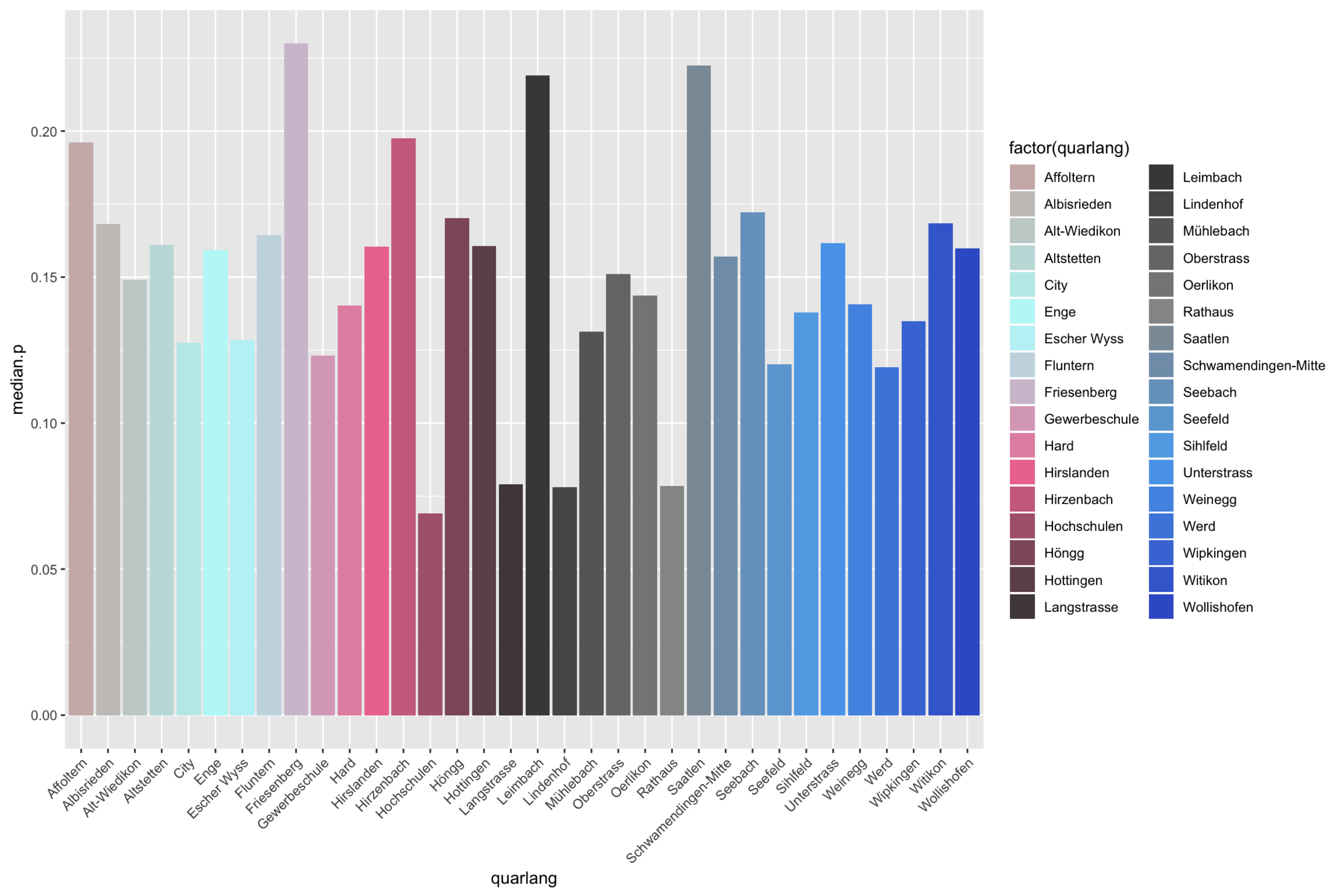
3 Custom colors: via expanding

Another example with custom colors:

```
# draw some random colors
set.seed(727678)
c <- sample(colors(), 7) %>% col2hex
c
```

```
## [1] "#CDB7B5" "#BBFFFF" "#EE799F" "#404040" "#969696" "#5CACEE" "#3A5FCD"
```

```
get.palette <- colorRampPalette(c)

kids %>%
  filter(has.kids==1) %>%
  ggplot() +
  geom_bar(aes(quarlang, median.p, fill=factor(quarlang)), stat='identity')
  # HERE you specify the colors:
  scale_fill_manual(values=get.palette(c.count))  +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



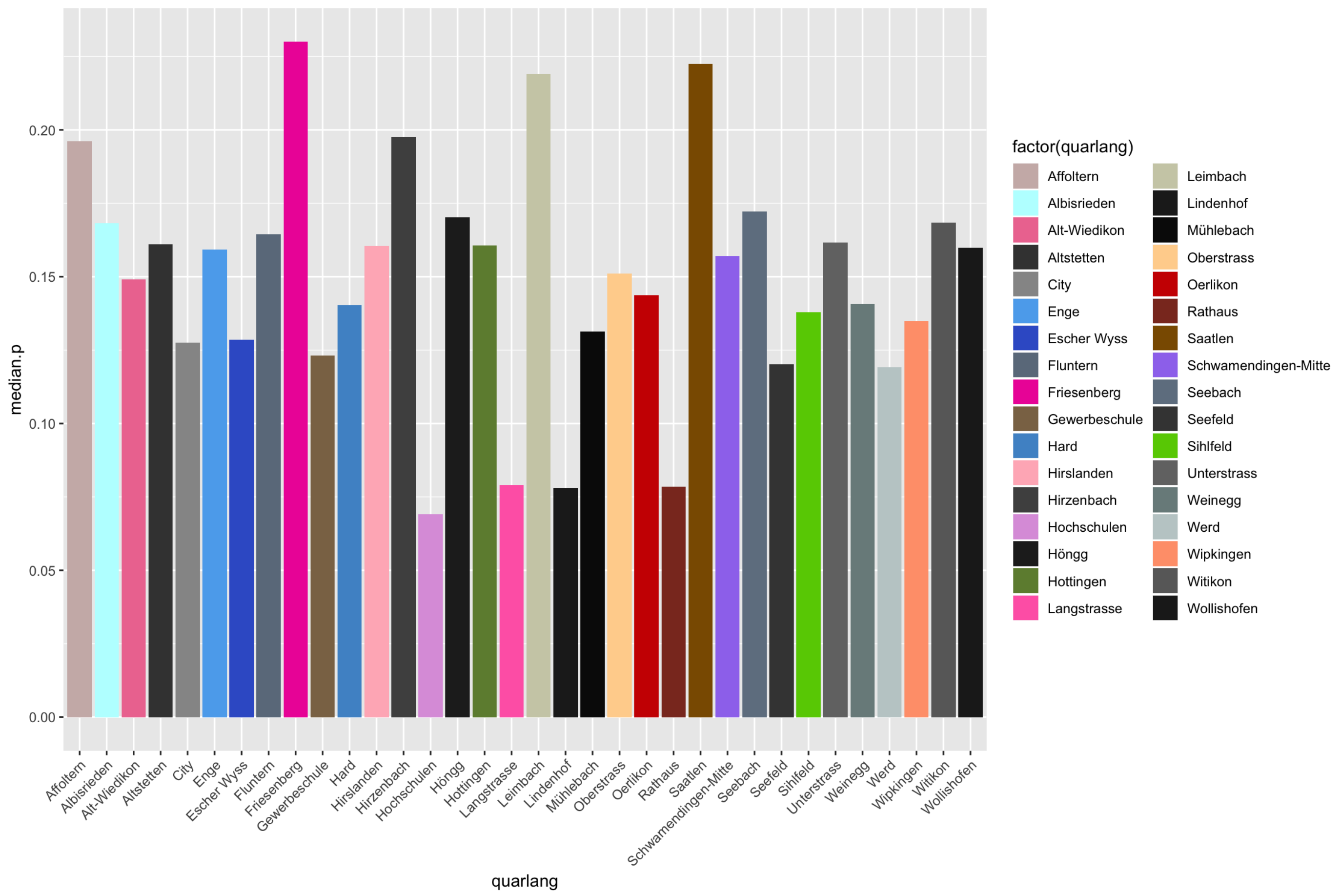
4 Custom colors: direct approach

Alternative example with custom colors:

```
# draw some random colors
set.seed(727678)
c <- sample(colors(), c.count) %>% col2hex
c
```

```
##  [1] "#CDB7B5" "#BBFFFF" "#EE799F" "#404040" "#969696" "#5CACEE" "#3A5FC
##  [8] "#6C7B8B" "#EE30A7" "#8B7355" "#4F94CD" "#FFB6C1" "#4F4F4F" "#DDA0D
## [15] "#242424" "#6E8B3D" "#FF69B4" "#CDCDB4" "#212121" "#0A0A0A" "#FFD39
## [22] "#CD0000" "#8B3626" "#8B5A00" "#9F79EE" "#708090" "#424242" "#66CD0
## [29] "#737373" "#7A8B8B" "#C1CDCD" "#FFA07A" "#696969" "#1F1F1F"
```

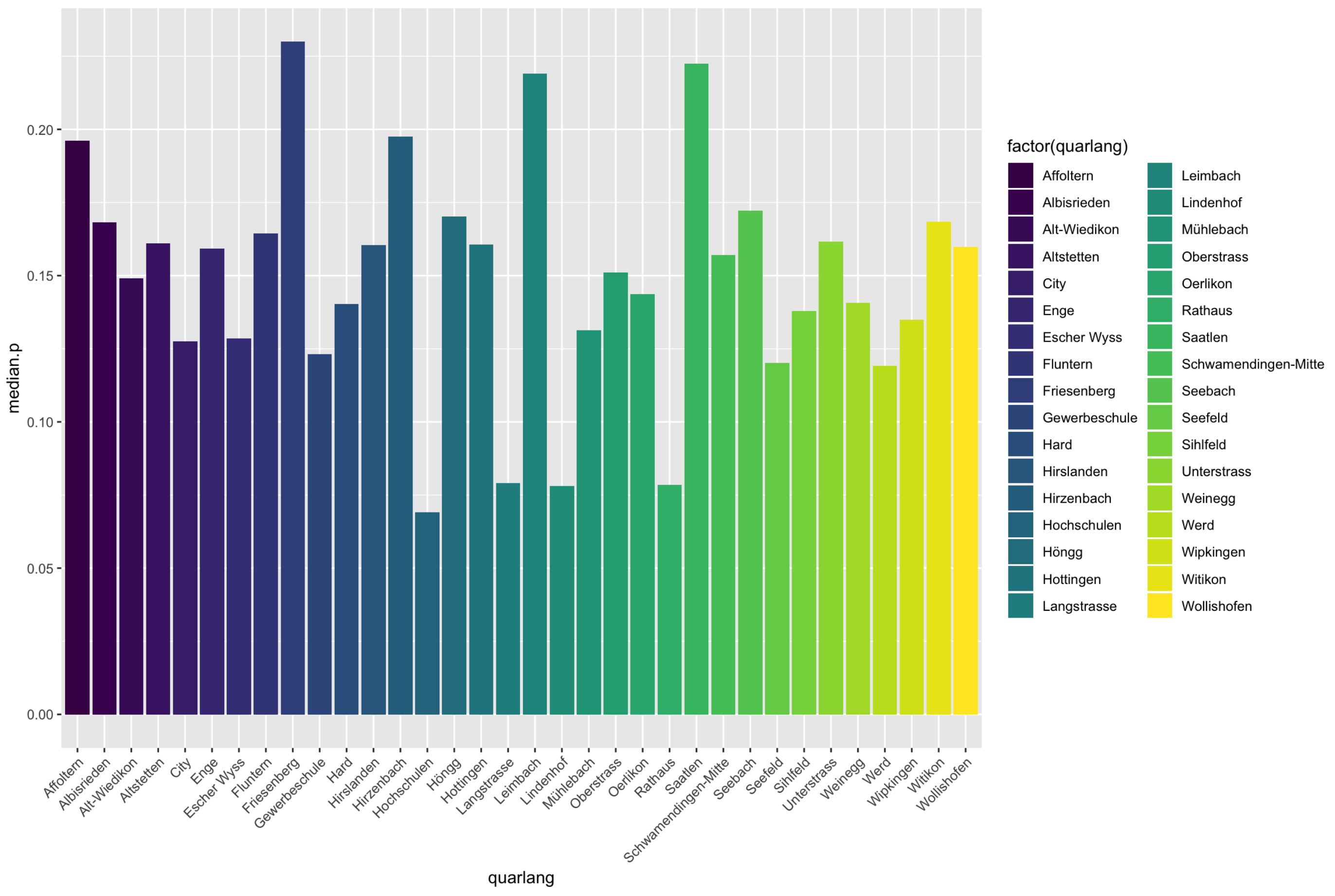
```
kids %>%
  filter(has.kids==1) %>%
  ggplot() +
  geom_bar(aes(quarlang, median.p, fill=factor(quarlang)), stat='identity')
  scale_fill_manual(values=c) +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



5 Viridis

Alternative example with viridis:

```
kids %>%
  filter(has.kids==1) %>%
  ggplot() +
  geom_bar(aes(quarlang, median.p, fill=factor(quarlang)), stat='identity')
  # HERE you specify discreteness
  scale_fill_viridis(discrete = T) +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```


median.p
quarlang
Affoltern
Albisrieden
Alt-Wiedikon
Altstetten
City
Enge
Escher Wyss
Fluntern
Friesenberg
Gewerbeschule
Hard
Hirslanden
Hirzenbach
Hochschulen
Höngg
Hottingen
Langstrasse
Leimbach
Lindenhof
Mühlebach
Oberstrass
Oerlikon
Rathaus
Saatlen
Schwamendingen-Mitte
Seebach
Seefeld
Sihlfeld
Unterstrass
Weinegg
Werd
Wipkingen
Witikon
Wollishofen
factor(quarlang)