Spatial Analysis mit R (I)

Methodenwoche

Till Straube
straube@geo.uni-frankfurt.de

Institut für Humangeographie
Goethe-Universität Frankfurt

20.–21. September 2021

Inhaltsverzeichnis

Zeitplan

Alle Sitzungen finden über Zoom statt.

Zeit	Montag	Dienstag
10:00–11:30	**(1) Getting started (LAS)**	**(5) Geodaten verschneiden: FTR**
11:30–11:45	*Kaffeepause*	*Kaffeepause*
11:45–13:15	**(2) Daten visualisieren (FTR, HOS)**	**(6) Geodaten verschneiden: HOS, SYW**
13:15–14:15	*Mittagspause*	*Mittagspause*
14:15–15:45	**(3) Karten erstellen (FTR)**	**(7) Nach Hilfe fragen und und publizieren (FTR)**
15:45–16:15	*Kaffeepause*	*Kaffeepause*
16:15–18:00	**(4) Karten erstellen (HOS, SYW)**	**(8) Looking back, looking ahead (LAS)**

1 Getting started

1.1 Formales

1.1.1 Keine reguläre Anrechnung des Workshops

- Die Methodenwoche ist eine außercurriculare Veranstaltung
- Keine Anrechnung als Prüfungsleistung für das reguläre Studium
- Alle Teilnehmer*innen erhalten Methodenzentrum eine Bescheinigung über die erbrachte Leistung (Ende 2021/Anfang 2022)

1.1.2 Methodenzertifikat

- Nur für Bachelor-Studierende der Fachbereiche 02–05
- Kann mit 5 CP beantragt werden
- z. B. Teilnahme Workshop (2 CP) + Leistungsnachweis (3 CP)
- Maximal 20% Fehlzeit zulässig für Teilnahmenachweis
- Schriftliche Leistungsnachweise mit maximal vierwöchiger Abgabefrist
- Alle Fragen dazu bitte an hiwis-methodenzentrum@uni-frankfurt.de

1.2 Inhaltliches

1.2.1 Lernziele der Veranstaltung

Sie können…

- Datenvisualisierungen nachvollziehen und selbst gestalten.
- Geodaten einlesen, transformieren und verschneiden.
- Geodaten kartographisch darstellen.
- Reproducible examples erstellen um nach Hilfe zu fragen.
- Berichte in Rmarkdown verfassen und rendern.

1.2.2 Seminarkonzept

- Kompetenter Umgang mit Geodaten als Kernziel
- Aber nicht im luftleeren Raum
- Das Drumherum ist mindestens genauso wichtig
- Unterlagen sind Auszüge aus einem zweisemestrigen Seminar

1.2.3 “Opinionated...”

- package choices:
  - `tidyverse`
  - `sf`
- coding style:
  - Functional
  - Pipes %>%
- workflow:
  - Incremental commands
  - Rmarkdown (reproducible research)

1.3 Didaktisches

1.3.1 Herausforderungen in der IT-Didaktik

- Unterschiedliche Erfahrungen, Kompetenzen und Herangehensweisen
- Die eine Hälfte versteht gar nichts, die andere langweilt sich
- Kleinster gemeinsamer Nenner: Schritt-für-Schritt-Anleitungen
- In der Praxis wertlos

1.3.2 Everyone fails

- In der Praxis stoßen alle ständig an die Grenzen ihrer technischen Kompetenz.
- Es geht darum, sich am Limit einigermaßen wohl zu fühlen und die Grenzen zu verschieben.
- Gute Angewohnheiten (Strukturen, Formate, Stil) helfen dabei!

1.3.3 Mein Ansatz in der Lehre

- Strategische Überforderung durch schwierige Aufgaben?
- Lösungsorientierte Didaktik!
- Die affektive Seite (Spaß, Frust) ernst nehmen und thematisieren
- Frustrationsschwelle trainieren

1.3.4 “Schattenkompetenzen”

- Über Code reden
- Fehlermeldungen lesen
- Gezielt googlen (und Antworten auswählen)
- Copy, paste, customize
- Gute Fragen (online) stellen

1.3.5 Dieser Workshop findet in verschiedenen Modi statt:

1.3.5.1 Listen and share (LAS)

- Ich rede (mit Folien oder ohne) oder moderiere eine Diskussion.
- Sie hören mir und Ihren Kommiliton*innen aufmerksam zu.
- Sie “melden” sich für Redebeiträge oder Fragen (Zoom-Funktion).

1.3.5.2 Follow the recipe (FTR)

- Ich teile ein unvollständiges Beispielprojekt.
- Wir gehen die Teilschritte nach und nach durch.
- Ich "habe den Plan", stelle aber immer wieder Fragen ans Plenum.
- Sie vollziehen die Schritte an Ihrer eigenen Kopie des Projekts nach.
- Sie unterbrechen mich mit Nachfragen oder Problemen.

1.3.5.3 Hands-on session (HOS)

- Sie bearbeiten praktische Aufgabenstellungen alleine.
- Dabei sind sie in zufälligen Dreier-Konstellationen (Breakout-Session).
- Bei Fragen oder Problemen wenden Sie sich zunächst an Ihre Kleingruppe.
- Falls Sie nicht weiterkommen, fordern Sie Hilfe an (Zoom-Funktion).
- Ich reagiere auf Hilfegesuche oder "mache die Runde".

1.3.5.4 Share your work (SYW)

- Ich wähle eine Teilnehmer*in zufällig aus.
- Die Person teilt ihren Bildschirm und berichtet von ihrer Bearbeitung eines Problems.
- Alle anderen unterstützen solidarisch durch aktives Nachvollziehen, Nachfragen und Hinweise.

1.4 Technisches

1.4.1 Arbeitsplatz

- Challenge: Zoom (meinen Bildschirm) und R gleichzeitig sehen
- Am allerbesten: Zweiter Bildschirm
- Auch gut: Zweites Gerät (Tablet)

1.4.2 Workshopunterlagen

- Bookdown (statt OLAT)
- ~~Können~~ Werden sich verändern
- Am besten neu laden mit Strg+Umschalt+R
- https://tiny.gu/mwsa

1.4.3 RStudio Cloud

- Grundsätzliche Empfehlung: R und RStudio lokal installieren
- Wir nutzen im Rahmen des Workshops die RStudio Cloud
  - für einfaches Teilen von Code
  - für ein homogenes Setup

2 Daten visualisieren

2.1 Lernziele dieser Sitzung

Sie können…

- einfache Befehle zur Visualisierung in Base R anwenden.
- die Grammatik von `ggplot2` für Visualisierungen in Grundzügen wiedergeben und anwenden.
- eigene Ideen für Visualisierungen entwickeln und umsetzen.

2.2 Voraussetzungen

Für diese Lektion benötigen wir das Paket `tidyverse`:

```
library(tidyverse)
```

Und einen Datensatz, der in Form eines tibble vorliegt. Der Beispieldatensatz `diamonds` wird mitgeliefert:

```
diamonds
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

Wenn wir mögen, können wir ihn mit der Funktion `data()` explizit in unser Environment laden:

```
data(diamonds)
```

2.3 Überblick

Einen ersten Überblick kriegen wir zum Einen durch den Befehl `str()`, der uns die Typen in den Spalten anzeigt:

```
str(diamonds)
## tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Zum Anderen gibt die Hilfefunktion Auskunft über den Datensatz und die einzelnen Variablen (Metadaten):

```
?diamonds
```

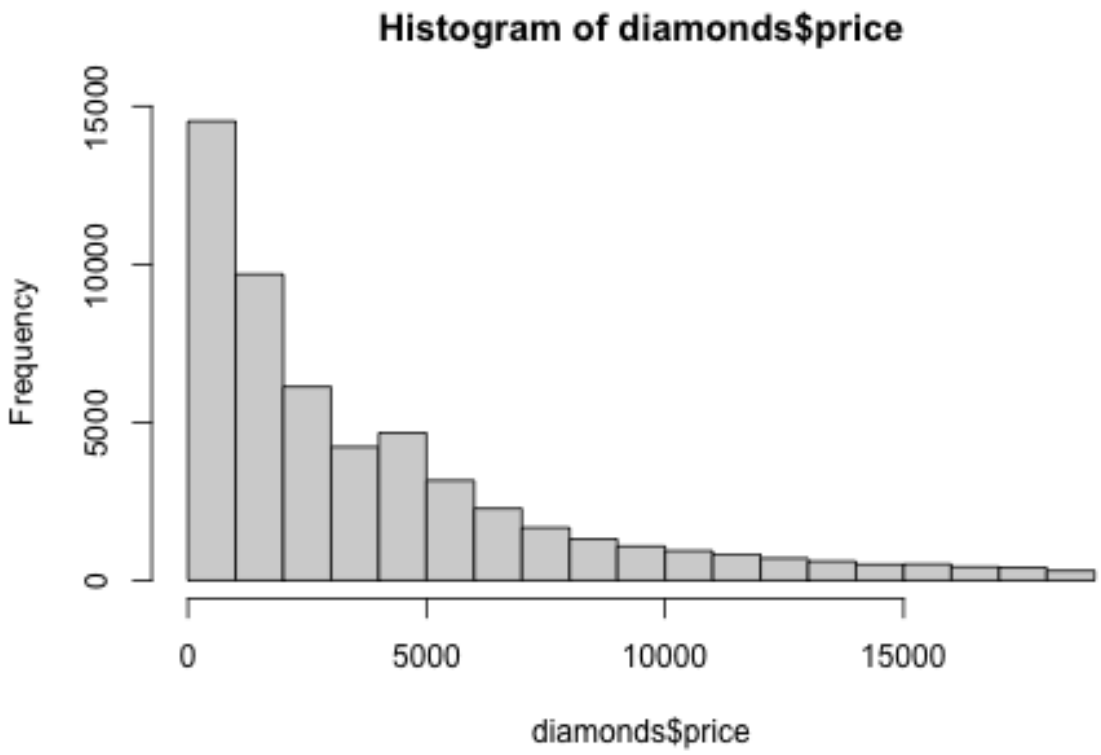
Einen Überblick über die wichtigsten statistischen Parameter erhalten wir mit:

```
summary(diamonds)
##      carat               cut        color        clarity          depth
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00
##  Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80
##  Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50
##  Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00
##                                     J: 2808   (Other): 2531
##      table           price             x                y
##  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720
##  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710
##  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735
##  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540
##  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900
##
##        z
##  Min.   : 0.000
##  1st Qu.: 2.910
##  Median : 3.530
##  Mean   : 3.539
##  3rd Qu.: 4.040
##  Max.   :31.800
##
```

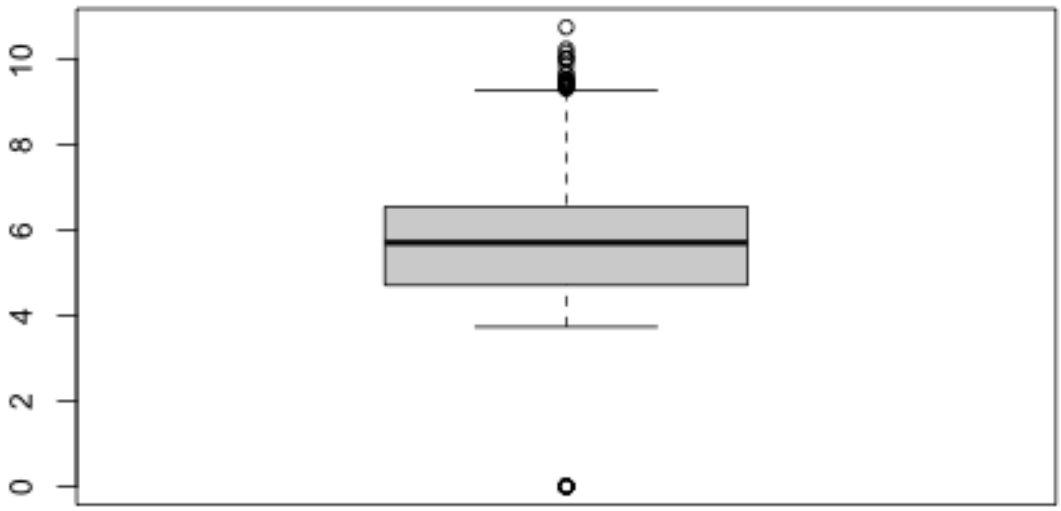
2.4 Visualisierung mit dem Standardpaket

Es gibt in R mehrere grundlegend verschiedene Möglichkeiten, Daten zu visualisieren. Für einen schnellen Überblick sind z.B. `hist()` und `boxplot()` hilfreich:

```
hist(diamonds$price)
```




```
boxplot(diamonds$x)
```



2.5 Visualisierung mit `ggplot()`

Das Paket `ggplot2` ist Teil vom `tidyverse`. Hiermit lassen sich sehr flexible Graphiken gestalten. Wir werden ausschließlich mit diesem System arbeiten.

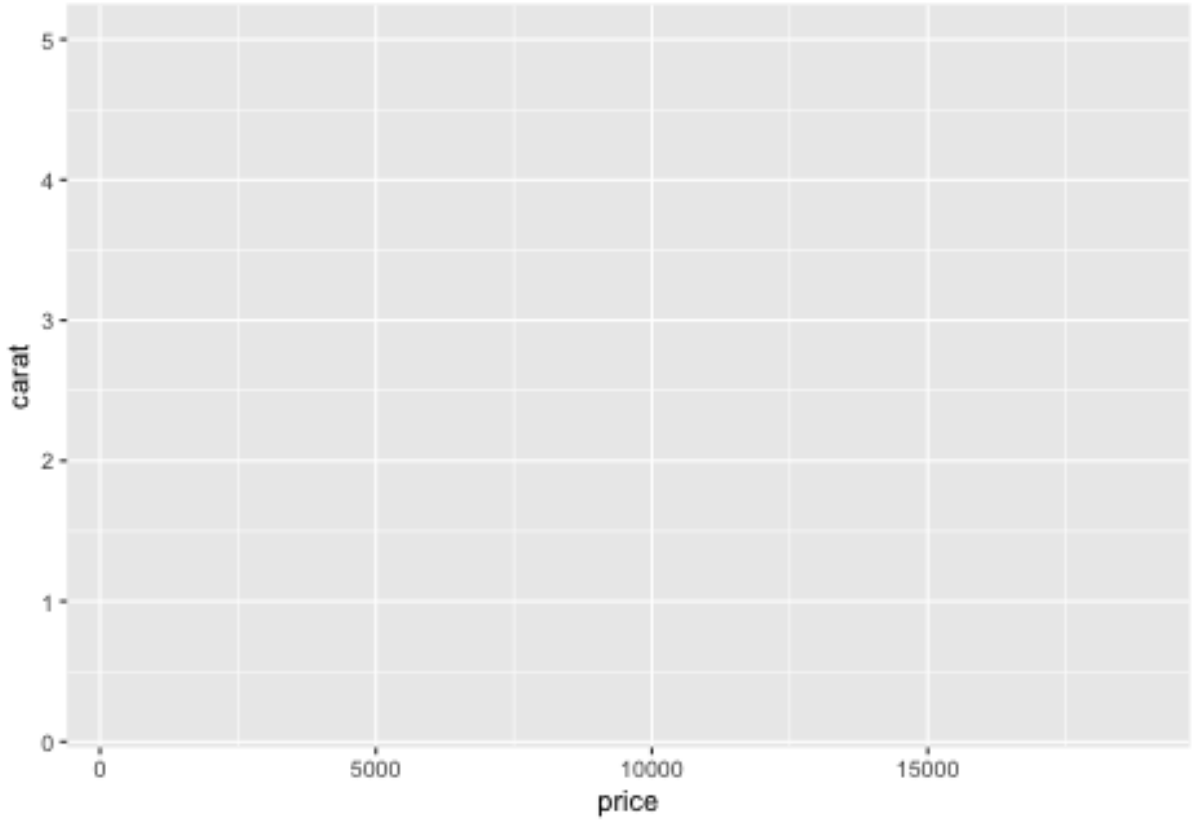
Die Syntax ist dabei auf den ersten Blick etwas komplexer.

Am Anfang steht der Befehl `ggplot(x)` mit dem Datensatz als Parameter

```
ggplot(data = diamonds)
```

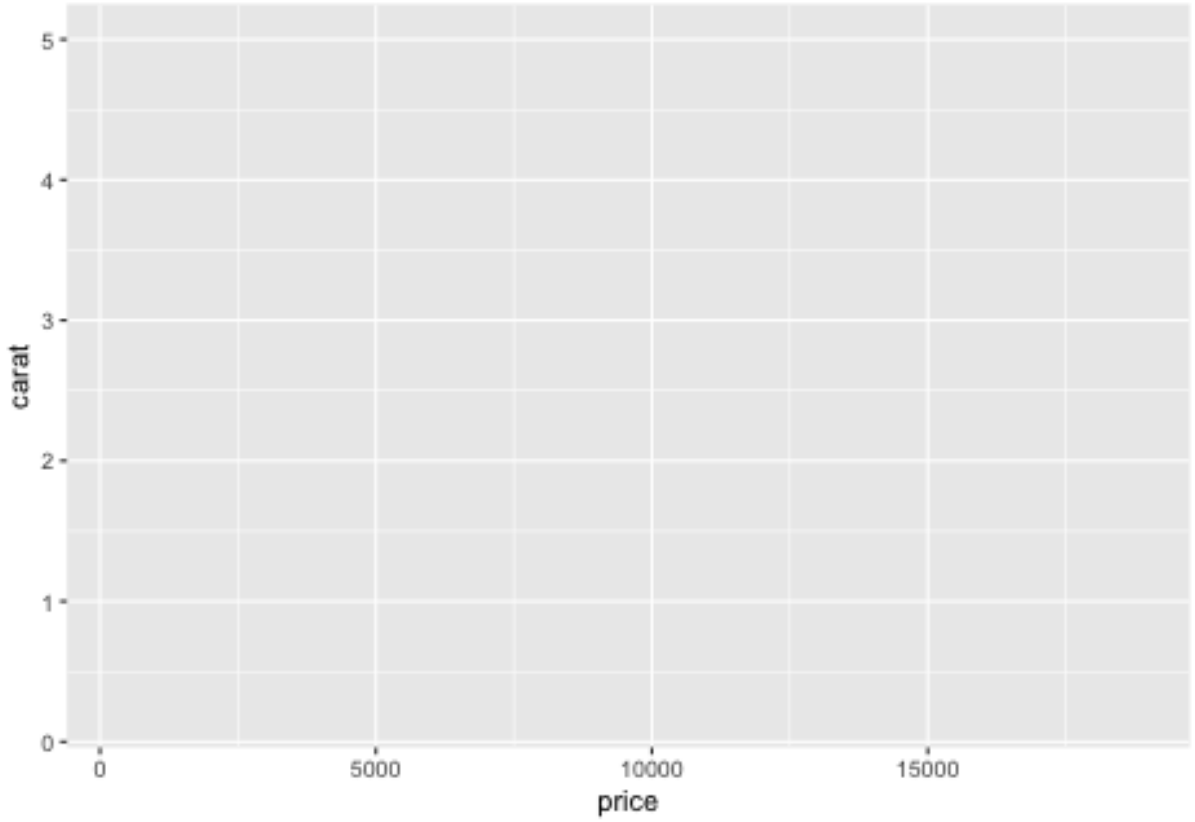
Mit einem Mapping-Parameter legen wir die Dimensionen fest:

```
ggplot(data = diamonds, mapping = aes(x = price, y = carat))
```



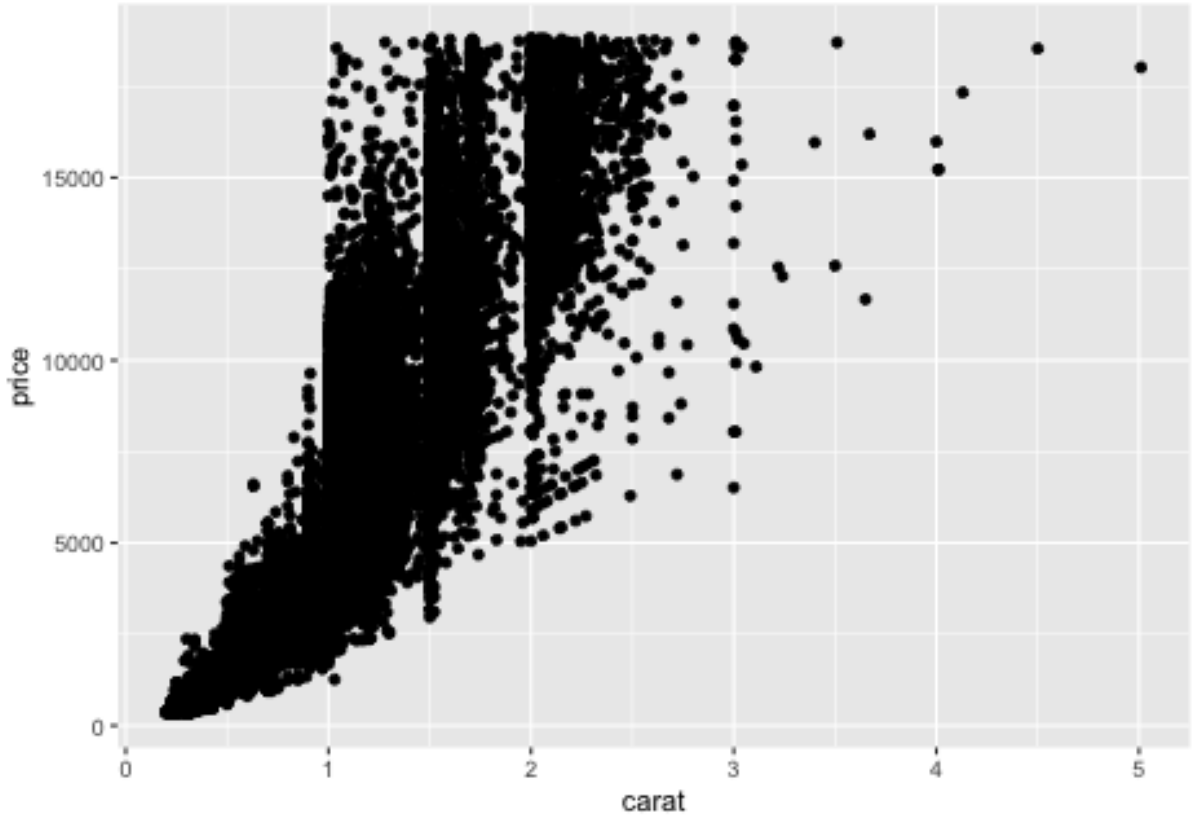
Das gleiche ohne Parameternamen:

```
ggplot(diamonds, aes(price, carat))
```



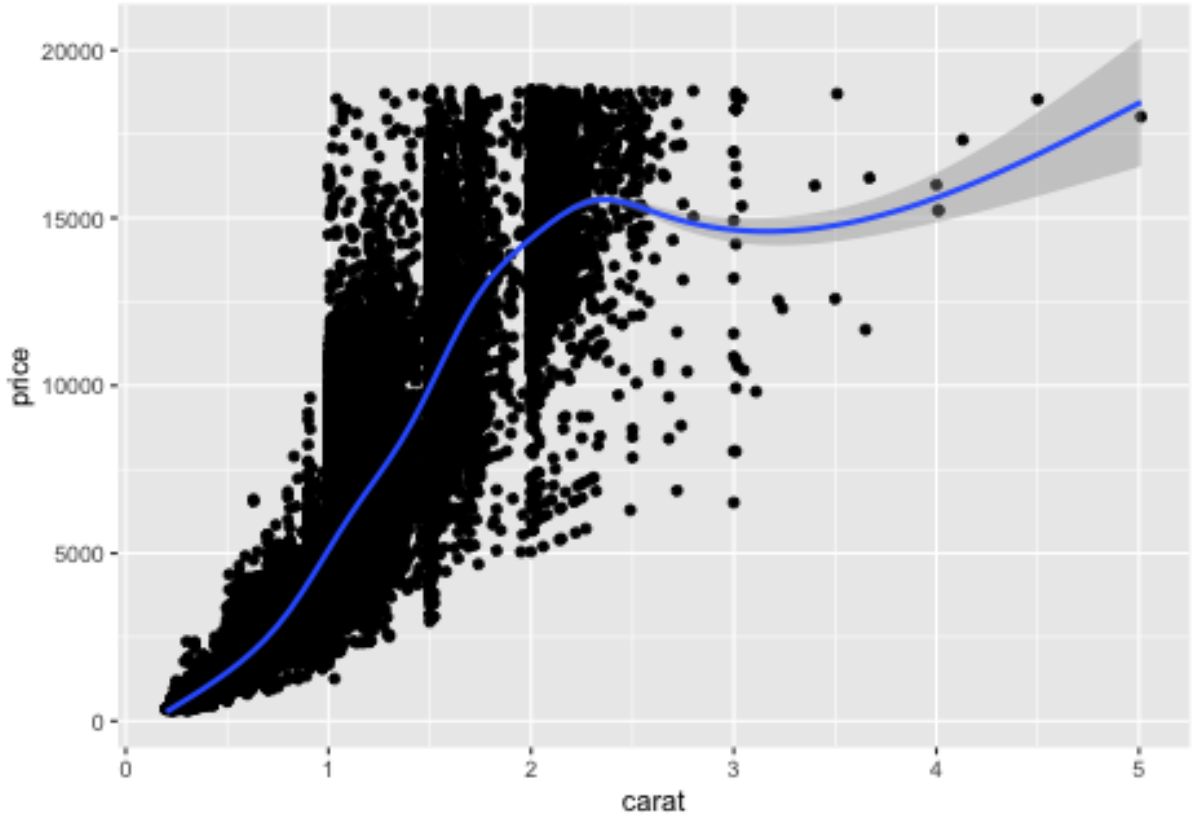
Nun kann mit dem +-Operator ein “geometrischer” Layer hinzugefügt werden:

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point()
```



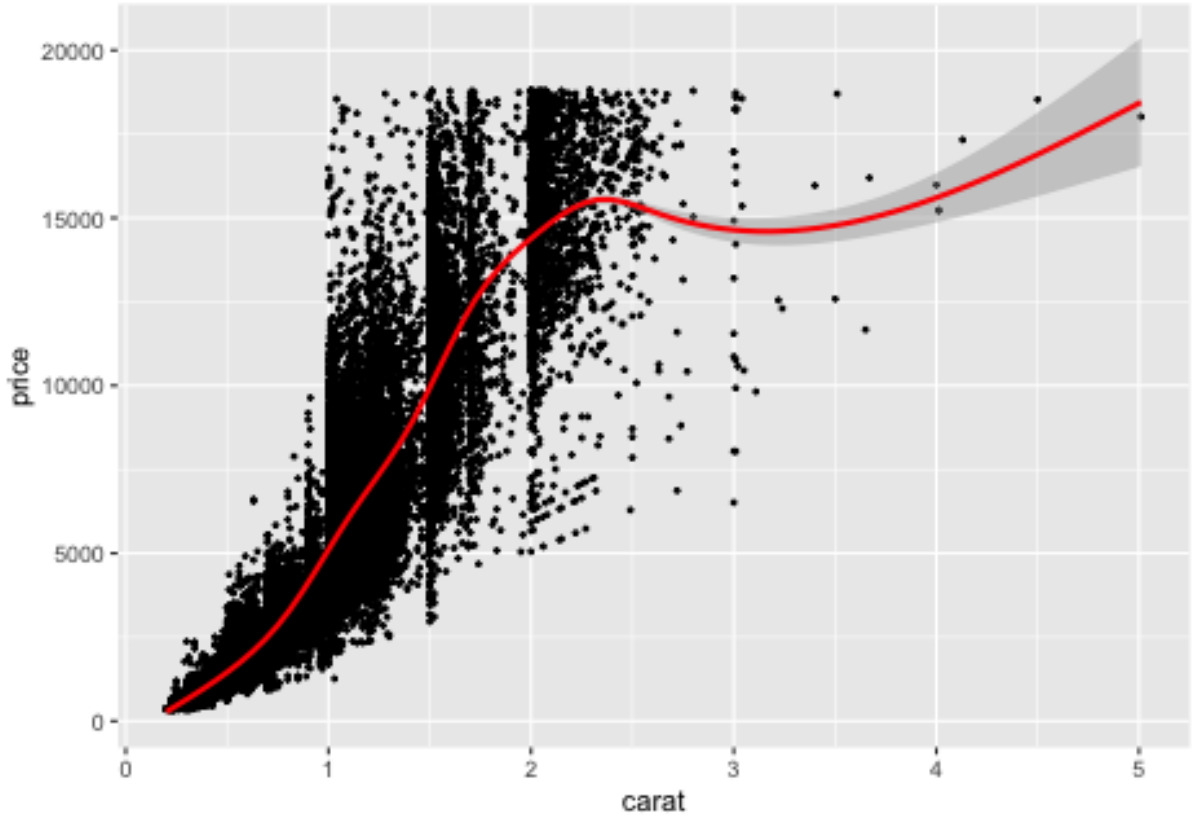
Weitere `geom`-Layer lassen sich mit dem +-Operator hinzufügen:

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth()
```



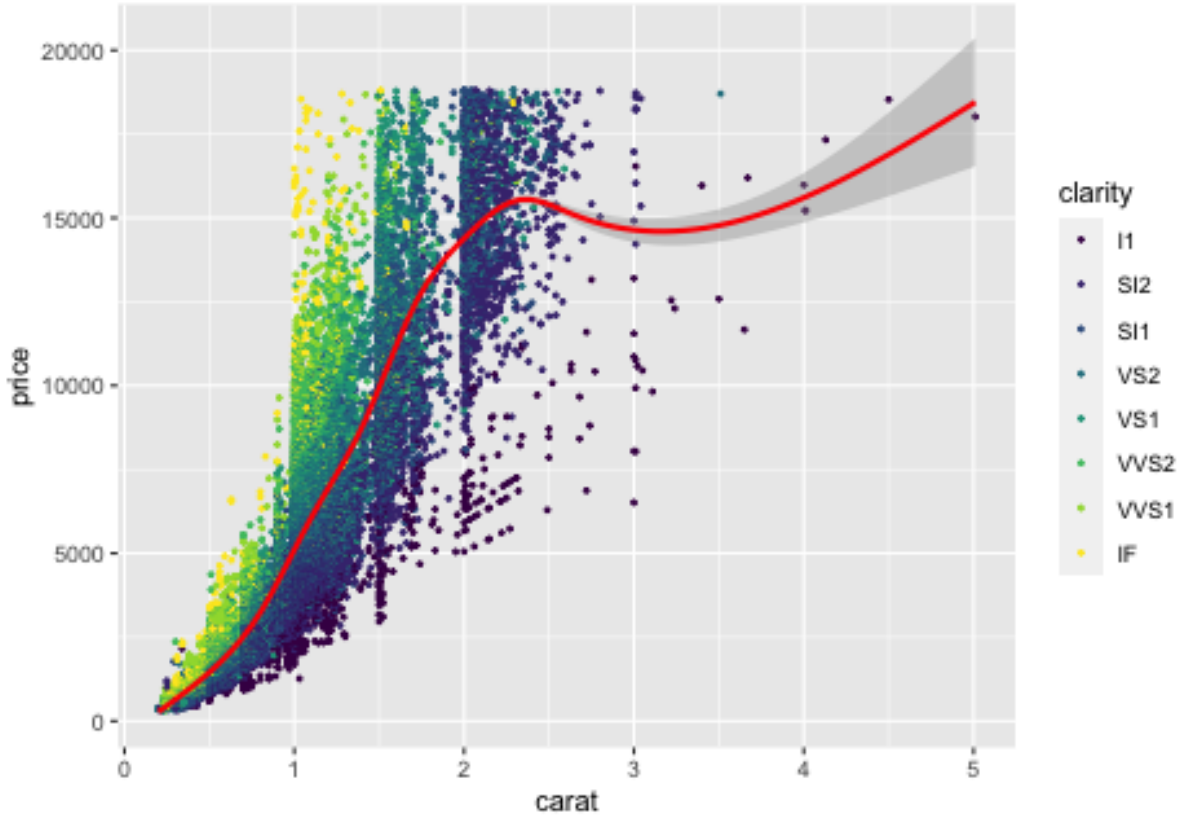
Die Layer-Funktionen können durch Parameter angepasst werden:

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(size = 0.5) +
  geom_smooth(color = "red")
```



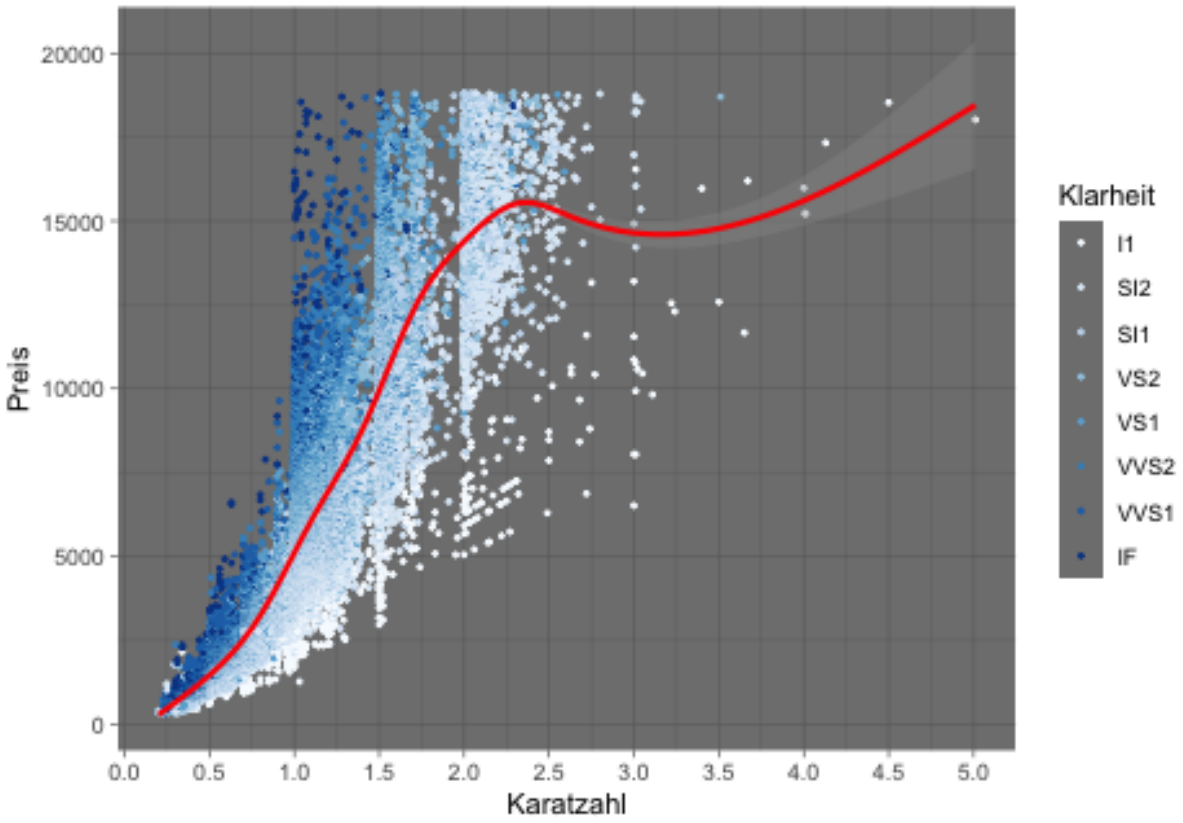
Dabei lassen sich in den einzelnen Layers mappings hinzufügen oder verändern:

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(aes(color = clarity), size = 0.5) +
  geom_smooth(color = "red")
```



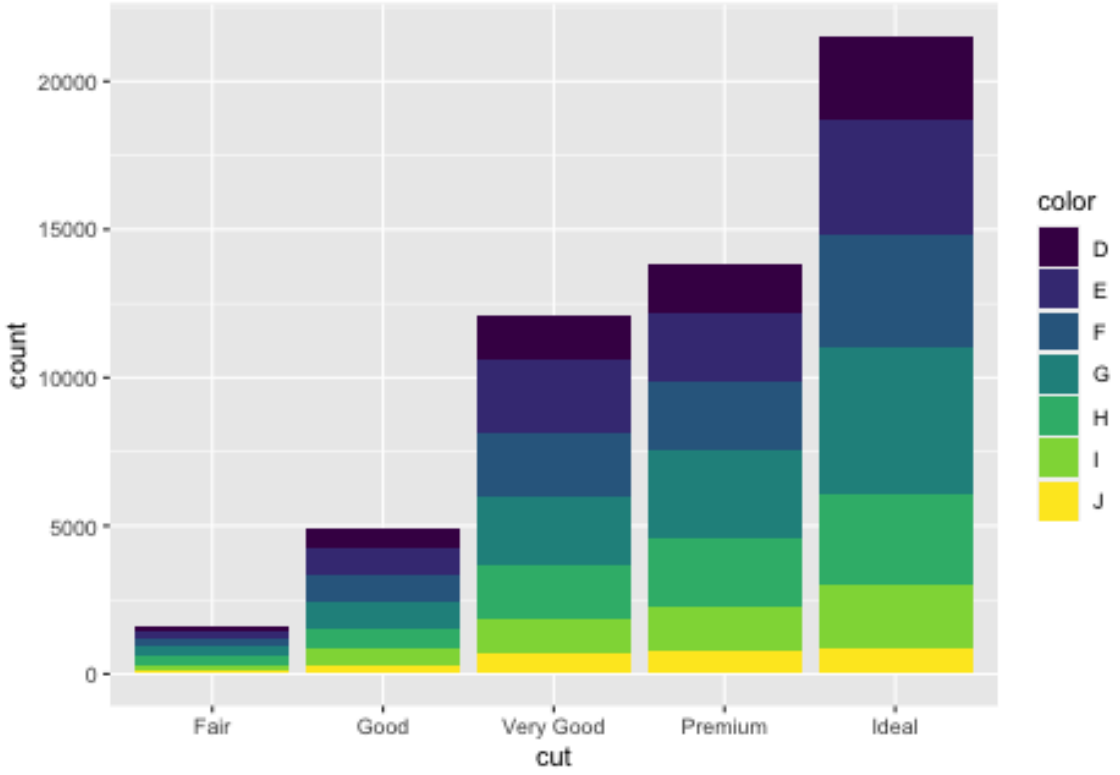
Schließlich lassen sich noch viele weitere optische Aspekte anpassen, z.B. Achsen, Farben, etc.:

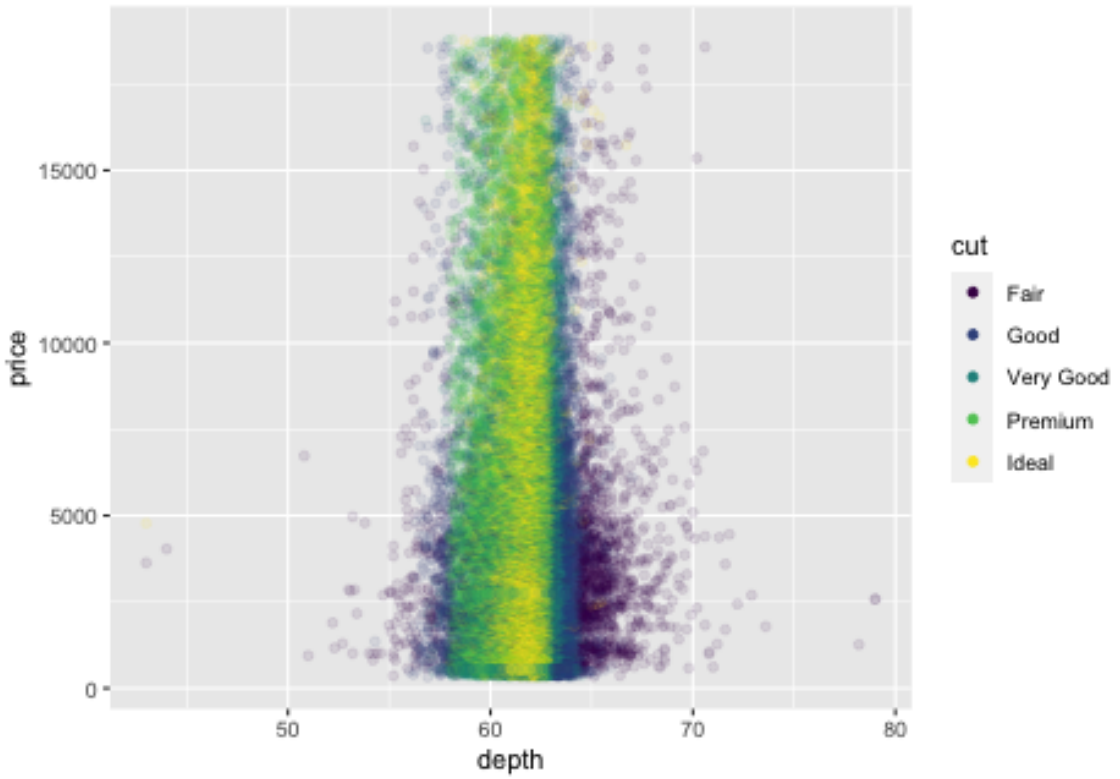
```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(aes(color = clarity), size = 0.5) +
  geom_smooth(color = "red") +
  scale_x_continuous("Karatzahl", breaks = seq(0, 5, 0.5)) +
  scale_y_continuous("Preis") +
  scale_color_brewer("Klarheit") +
  theme_dark()
```

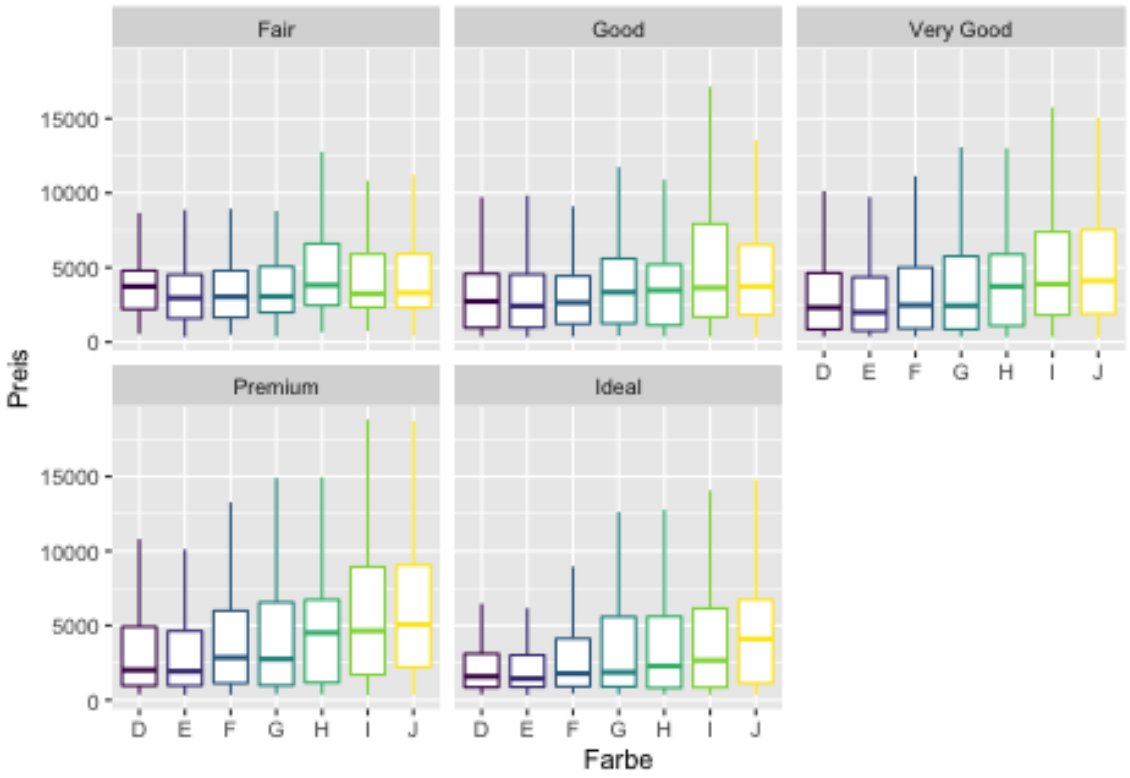


2.6 Aufgaben

1. Versuchen Sie, die folgenden Visualisierungen des Datensatzes `diamonds` auszugeben:


20000
15000
10000
5000
0
count
Fair
Good
Very Good
Premium
Ideal
cut
color
D
E
F
G
H
I
J


15000
10000
5000
0
price
50
60
70
80
depth
cut
Fair
Good
Very Good
Premium
Ideal



2. Schauen Sie sich die Publikation R for Data Science an.
3. Was ist das für ein Buch? Wer ist das Zielpublikum?
4. Lesen Sie das Kapitel “3: Data Visualization” und vollziehen Sie die Visualisierungen nach.
5. Bearbeiten Sie die Aufgaben.
6. Bearbeiten Sie die RStudio Primers zu Datenvisualisierung.

3 Karten erstellen (FTR)

3.1 Lernziele dieser Sitzung

Sie können…

- Pipes benutzen
- einfache `dplyr`-Befehle ausführen
- Koordinaten visualisieren

3.2 Voraussetzungen

Wir laden erstmal `tidyverse`:

```
library(tidyverse)
```

3.3 Exkurs: Pipes

Teil vom `tidyverse` ist auch das Paket `magrittr`, das einen besonderen Operator enthält: %>%

Der Operator %>% heißt "Pipe" und setzt das Ergebnis der vorherigen Funktion als ersten Parameter in die nächste Funktion ein. Zur Veranschaulichung:

```
anzahl_buchstaben <- length(letters)
sqrt(anzahl_buchstaben)
```

…ist das gleiche wie…

```
sqrt(length(letters))
```

…ist das gleiche wie…

```
length(letters) %>%
  sqrt()
```

…ist das gleiche wie…

```
letters %>%
  length %>%
  sqrt()
```

So können beliebig viele Funktionen aneinandergereiht werden. Und mit -> kann eine Variable „in die andere Richtung" zugewiesen werden

```
letters %>%
  length() %>%
  sqrt() %>%
  round() %>%
  as.character() ->
  my_var
```

Gerade bei komplizierteren Zusammenhängen wird der Code so oft lesbarer, weil die Logik von links nach rechts, bzw. von oben nach unten gelesen werden kann.

3.4 Daten importieren

Beim Open-Data-Portal der Stadt Frankfurt steht ein Baumkataster zur Verfügung.

Die Datei im CSV-Format (comma separated values) kann entweder heruntergeladen und durch klicken importiert werden, oder direkt über den Befehl:

```
baumkataster <- read_csv2("http://offenedaten.frankfurt.de/dataset/73c5a6b3-c033-4dad-bb7d-87
```

3.5 Überblick verschaffen

Mit `summary()` lässt sich eine Zusammenfassung der Werte generieren:

```
summary(baumkataster)
##  Gattung/Art/Deutscher Name   Baumnummer         Objekt             Pflanzjahr
##  Length:118403              Min.   :    1.0   Length:118403      Min.   :1645
##  Class :character           1st Qu.:   24.0   Class :character   1st Qu.:1970
##  Mode  :character           Median :   82.0   Mode  :character   Median :1982
##                             Mean   :  232.7                      Mean   :1979
##                             3rd Qu.:  270.0                      3rd Qu.:1995
```

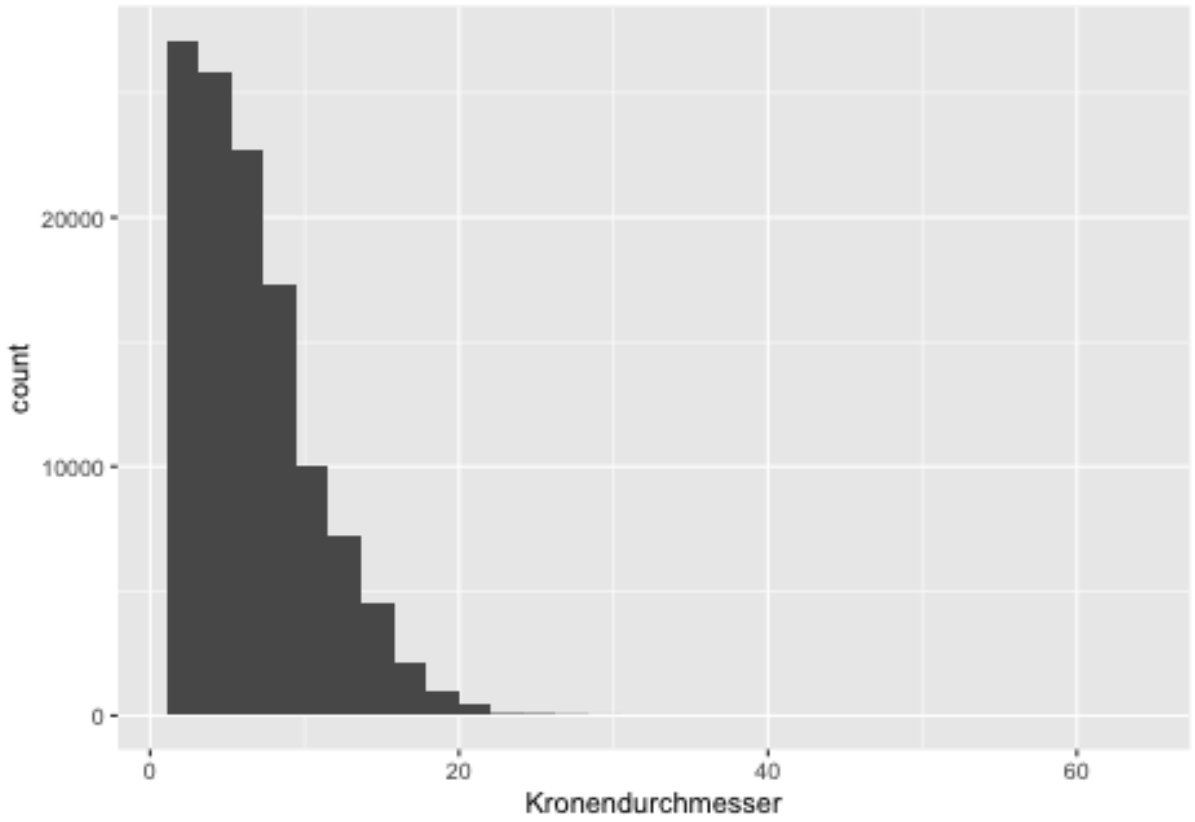
```
##                               Max.   :20158.0                       Max.   :2017
##                               NA's   :1853
##  Kronendurchmesser    HOCHWERT          RECHTSWERT
##  Min.   : 2.000    Min.   :5545117   Min.   :463163
##  1st Qu.: 4.000    1st Qu.:5550428   1st Qu.:472715
##  Median : 6.000    Median :5552601   Median :475219
##  Mean   : 6.688    Mean   :5552953   Mean   :475244
##  3rd Qu.: 9.000    3rd Qu.:5555165   3rd Qu.:478201
##  Max.   :63.000    Max.   :5563639   Max.   :485361
##
```

Genauere Infos über diese Merkmale gibt es auf dem Datenportal.

3.6 Visualisieren

Wie in der letzten Lektion besprochen, lässt sich der Datensatz mit `ggplot()` visualisieren, z. B.:

```
ggplot(baumkataster, aes(x = Kronendurchmesser)) +
  geom_histogram()
```



Eine neue Messreihe lässt sich z. B. so errechnen:

```
alter <- 2020 - baumkataster$Pflanzjahr
head(alter)
## [1] 100 100 100 100 100 100
```

Der Befehl `mutate()` funktioniert sehr ähnlich, gibt aber den veränderten Datensatz zurück:

```
mutate(baumkataster, alter = 2020 - Pflanzjahr)
## # A tibble: 118,403 x 8
##    `Gattung/Art/Deutsch~ Baumnummer Objekt  Pflanzjahr Kronendurchmess~ HOCHWERT
##    <chr>                       <dbl> <chr>        <dbl>            <dbl>    <dbl>
##  1 Platanus x hispanica~           1 Ackerm~       1920                8 5549511.
##  2 Platanus x hispanica~           2 Ackerm~       1920                8 5549517.
##  3 Platanus x hispanica~           3 Ackerm~       1920                8 5549524.
##  4 Platanus x hispanica~           4 Ackerm~       1920                8 5549531
##  5 Platanus x hispanica~           5 Ackerm~       1920                8 5549538.
##  6 Platanus x hispanica~           6 Ackerm~       1920                8 5549544.
##  7 Platanus x hispanica~           7 Ackerm~       1920                8 5549551.
##  8 Platanus x hispanica~           8 Ackerm~       1920                8 5549557.
##  9 Platanus x hispanica~           9 Ackerm~       1920                8 5549564.
## 10 Platanus x hispanica~          10 Ackerm~       1920                8 5549571.
## # ... with 118,393 more rows, and 2 more variables: RECHTSWERT <dbl>,
## #   alter <dbl>
```

Derselbe Befehl mit dem Pipe-Operator:

```
baumkataster %>%
  mutate(alter = 2020 - Pflanzjahr)
## # A tibble: 118,403 x 8
##    `Gattung/Art/Deutsch~ Baumnummer Objekt  Pflanzjahr Kronendurchmess~ HOCHWERT
##    <chr>                       <dbl> <chr>        <dbl>            <dbl>    <dbl>
##  1 Platanus x hispanica~           1 Ackerm~       1920                8 5549511.
##  2 Platanus x hispanica~           2 Ackerm~       1920                8 5549517.
##  3 Platanus x hispanica~           3 Ackerm~       1920                8 5549524.
##  4 Platanus x hispanica~           4 Ackerm~       1920                8 5549531
##  5 Platanus x hispanica~           5 Ackerm~       1920                8 5549538.
##  6 Platanus x hispanica~           6 Ackerm~       1920                8 5549544.
##  7 Platanus x hispanica~           7 Ackerm~       1920                8 5549551.
##  8 Platanus x hispanica~           8 Ackerm~       1920                8 5549557.
##  9 Platanus x hispanica~           9 Ackerm~       1920                8 5549564.
## 10 Platanus x hispanica~          10 Ackerm~       1920                8 5549571.
## # ... with 118,393 more rows, and 2 more variables: RECHTSWERT <dbl>,
## #   alter <dbl>
```

So lassen sich auch hier verschiedene Befehle verknüpfen. `filter()` beschränkt den Datensatz auf Merkmalsträger, die den Kriterien entsprechen:

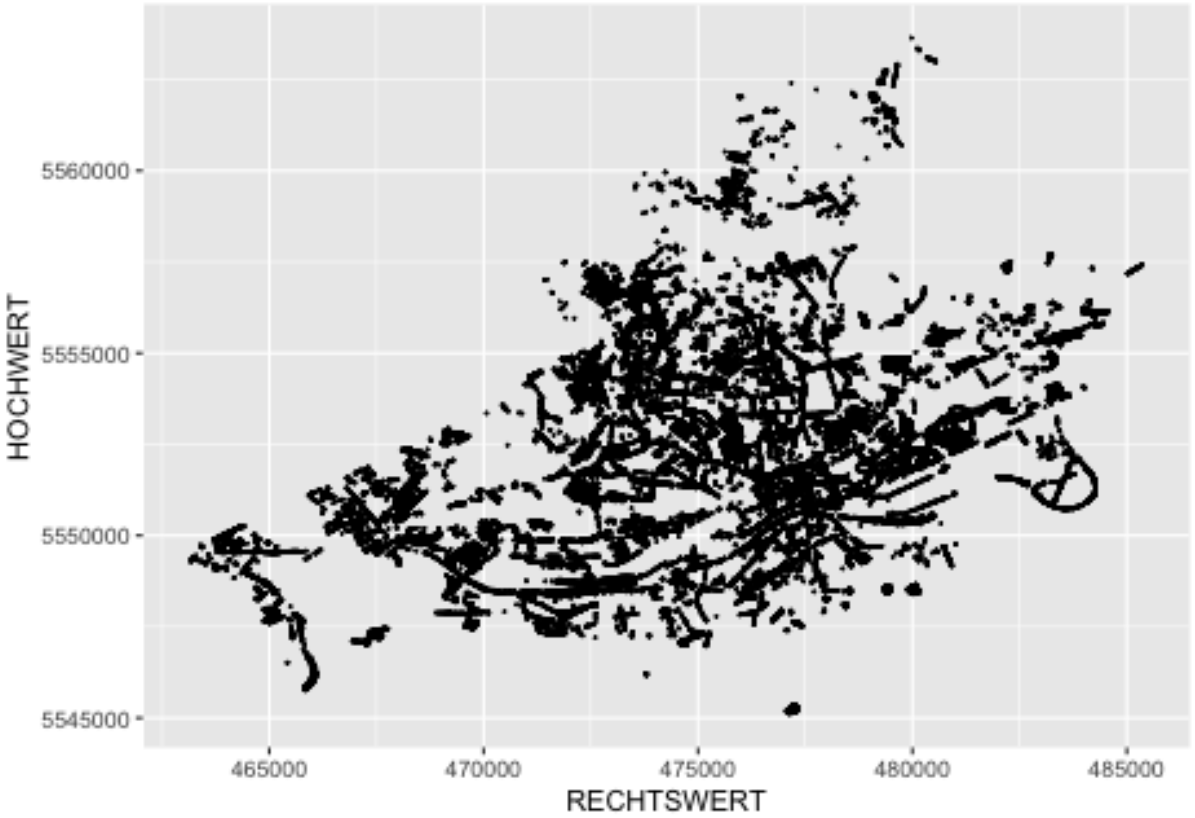
```
baumkataster %>%
  mutate(alter = 2020 - Pflanzjahr) %>%
  filter(alter > 30) ->
  alte_baeume

summary(alte_baeume)
##  Gattung/Art/Deutscher Name   Baumnummer         Objekt            Pflanzjahr
##  Length:73859               Min.   :    1.0   Length:73859       Min.   :1645
##  Class :character           1st Qu.:   29.0   Class :character   1st Qu.:1960
```

```
##  Mode  :character           Median :   97.0   Mode  :character   Median :1974
##                             Mean   :  263.2                     Mean   :1966
##                             3rd Qu.:  314.0                     3rd Qu.:1980
##                             Max.   :10489.0                     Max.   :1989
##                             NA's   :684
##  Kronendurchmesser    HOCHWERT         RECHTSWERT         alter
##  Min.   : 2.000    Min.   :5545117   Min.   :463163   Min.   : 31.00
##  1st Qu.: 6.000    1st Qu.:5550415   1st Qu.:472667   1st Qu.: 40.00
##  Median : 8.000    Median :5552480   Median :475708   Median : 46.00
##  Mean   : 8.503    Mean   :5552593   Mean   :475402   Mean   : 53.54
##  3rd Qu.:10.000    3rd Qu.:5554589   3rd Qu.:478539   3rd Qu.: 60.00
##  Max.   :35.000    Max.   :5563639   Max.   :485360   Max.   :375.00
##
```

Schließlich ergibt das Streudiagramm von Koordinaten so eine art Karte:

```
ggplot(alte_baeume) +
    geom_point(size = 0.1, aes(x = RECHTSWERT, y = HOCHWERT))
```



Diesen Ansatz werden wir in der nächsten Lektion vertiefen.

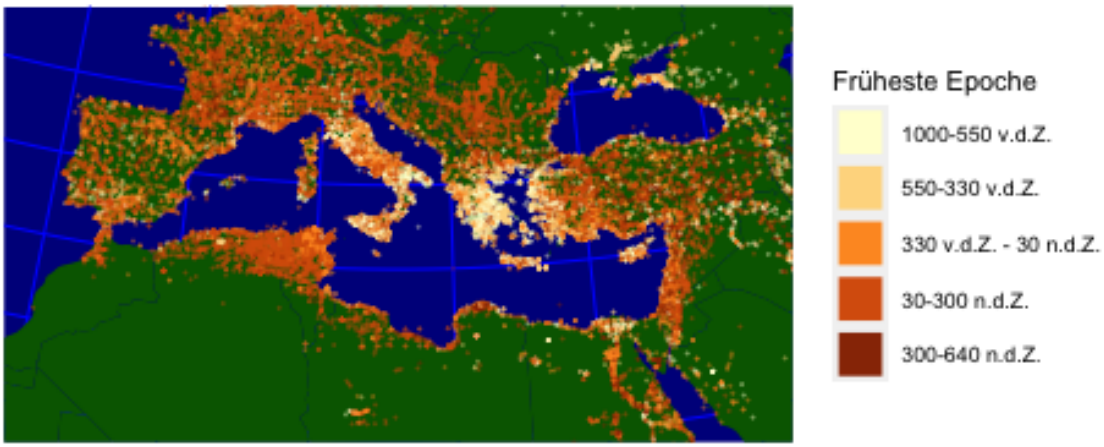
4 Karten erstellen (HOS)

4.1 Aufgaben

1. Besuchen Sie https://pleiades.stoa.org/ - worum geht es hier?
2. Finden Sie den kompletten aktuellen Datensatz für „locations“ als CSV-Datei.
3. Importieren Sie ihn in R und weisen Sie dem Datensatz den Namen `pleiades` zu.
4. Finden Sie geeignete Werte für (einzelne) Längen- und Breitengrade im Datensatz.
5. Plotten Sie die Koordinaten auf x- und y-Achse mit `ggplot()`. Was erkennen Sie?
6. Halbieren Sie die Größe und setzen Sie den Alpha-Wert der Punkte auf 0,2.
7. Bringen Sie die Grafik in die Mercator-Projektion.
8. Schauen Sie sich diesen Befehl an:

```
map_data("world") %>%
  ggplot() +
    geom_polygon(mapping = aes(x = long,
                               y = lat,
                               group = group)) +
    coord_quickmap(xlim = c(-8, 40),
                   ylim = c(26, 48))
```

9. Versuchen Sie, jede einzelne Zeile nachzuvollziehen, indem Sie die entsprechenden Funktionen recherchieren.
10. Führen Sie den Befehl aus.
11. Ändern Sie die Farbe der Flächen in hellgrau.
12. Wählen Sie einen Kartenausschnitt, auf dem Portugal, Ägypten, Irak und Frankreich komplett zu sehen sind.
13. Plotten Sie auf diesem Hintergrund den Datensatz `pleiades`. Passen Sie dabei die Parameter so an, dass es Ihnen optisch zusagt.
14. Wählen Sie für die Karte die Bonnesche Projektion mit Standardparallele bei 40°N.
15. Entfernen Sie alle Achsenbeschriftungen.
16. (Achtung: knifflig!) Bilden Sie diese Grafik nach, die die Orte geordnet nach ältestem Fund darstellt:


Früheste Epoche
1000-550 v.d.Z.
550-330 v.d.Z.
330 v.d.Z. - 30 n.d.Z.
30-300 n.d.Z.
300-640 n.d.Z.