Einführung in R

Clemens Brunner

18.-19.10.2019

Deskriptive Statistiken

Wenn man die Daten im gewünschten Format in R geladen hat (d.h. wenn diese in einem Data Frame oder Tibble vorhanden sind, im gewünschten Wide- oder Long-Format vorliegen, und alle Spalten den passenden Datentyp haben), kann man mit der statistischen Analyse beginnen. Der erste Schritt ist meist, sich mittels desktriptiven Statistiken einen Überblick über die Daten zu verschaffen.

Als Beispiel lesen wir wieder die Daten aus der letzten Übung in ein Data Frame bzw. Tibble ein.

```
library(readr)
df <- read_tsv("lecturer.dat")
df
```

```
# A tibble: 10 x 7
   name   birth_date   job friends alcohol income neurotic
   <chr>  <chr>      <dbl>   <dbl>   <dbl>  <dbl>    <dbl>
 1 Ben    7/3/1977       1       5      10  20000       10
 2 Martin 5/24/1969      1       2      15  40000       17
 3 Andy   6/21/1973      1       0      20  35000       14
 4 Paul   7/16/1970      1       4       5  22000       13
 5 Graham 10/10/1949     1       1      30  50000       21
 6 Carina 11/5/1983      2      10      25   5000        7
 7 Karina 10/8/1987      2      12      20    100       13
 8 Doug   1/23/1989      2      15      16   3000        9
 9 Mark   5/20/1973      2      12      17  10000       14
10 Zoe    11/12/1984     2      17      18     10       13
```

Danach führen wir wieder die Konvertierungen für die Spalten `birth_date` und `job` durch.

```
df$birth_date <- as.Date(df$birth_date, format="%m/%d/%Y")
df$job <- factor(df$job, levels=c(1, 2), labels=c("Lecturer", "Student"))
df
```

```
# A tibble: 10 x 7
   name   birth_date job      friends alcohol income neurotic
   <chr>  <date>     <fct>      <dbl>   <dbl>  <dbl>    <dbl>
 1 Ben    1977-07-03 Lecturer       5      10  20000       10
 2 Martin 1969-05-24 Lecturer       2      15  40000       17
 3 Andy   1973-06-21 Lecturer       0      20  35000       14
 4 Paul   1970-07-16 Lecturer       4       5  22000       13
 5 Graham 1949-10-10 Lecturer       1      30  50000       21
 6 Carina 1983-11-05 Student       10      25   5000        7
 7 Karina 1987-10-08 Student       12      20    100       13
 8 Doug   1989-01-23 Student       15      16   3000        9
 9 Mark   1973-05-20 Student       12      17  10000       14
10 Zoe    1984-11-12 Student       17      18     10       13
```

Die `name`-Spalte brauchen wir für unsere nachfolgenden Betrachtungen nicht mehr, deswegen entfernen wir sie:

```
df$name <- NULL
```

Zusammenfassende Beschreibungen

Es gibt in R eine Reihe an Funktionen, welche zusammenfassende Statistiken eines Vektors berechnen. Nützliche Funktionen sind z.B. `mean`, `sd`, `var`, `min`, `max`, `median`, `range` und `quantile`. Den Mittelwert einer Spalte von `df` kann man also wie folgt berechnen:

```
mean(df$friends)
```

```
[1] 7.8
```

Dies müsste man nun für jede interessierende Spalte wiederholen, was relativ mühsam wäre. Deswegen gibt es in R die Funktion `sapply`, welche eine Funktion auf jede Spalte eines Data Frames einzeln anwendet. Möchte man also den Mittelwert für jede numerische Spalte von `df` (also alle Spalten bis auf die ersten beiden) berechnen, kann man dies so tun:

```
sapply(df[, -c(1, 2)], mean)
```

```
 friends  alcohol   income neurotic
     7.8     17.6  18511.0     13.1
```

So kann man jede beliebige Funktion auf mehrere Spalten gleichzeitig anwenden.

Es gibt aber auch spezielle Funktionen, welche mehrere statistische Kenngrößen für alle Spalten eines Data Frames berechnen. Im Folgenden gehen wir näher auf drei dieser Funktionen ein, nämlich `summary`, `describe` und `stat.desc`.

Die Funktion `summary`

Die Funktion `summary` liefert eine geeignete Zusammenfassung für jede Spalte eines Data Frames (Tibbles). Numerische Spalten sowie Datumsspalten werden mit sechs Werten beschrieben: Minimum, 1. Quartil, Median, 3. Quartil, Maximum sowie Mittelwert. Für Faktoren werden die Stufen sowie die Anzahl an Fällen pro Stufe aufgelistet.

```
summary(df)
```

```
   birth_date             job       friends        alcohol          income         neurotic    
 Min.   :1949-10-10   Lecturer:5   Min.   : 0.0   Min.   : 5.00   Min.   :   10   Min.   : 7.00  
 1st Qu.:1971-04-01   Student :5   1st Qu.: 2.5   1st Qu.:15.25   1st Qu.: 3500   1st Qu.:10.75  
 Median :1975-06-27                Median : 7.5   Median :17.50   Median :15000   Median :13.00  
 Mean   :1975-12-17                Mean   : 7.8   Mean   :17.60   Mean   :18511   Mean   :13.10  
 3rd Qu.:1984-08-10                3rd Qu.:12.0   3rd Qu.:20.00   3rd Qu.:31750   3rd Qu.:14.00  
 Max.   :1989-01-23                Max.   :17.0   Max.   :30.00   Max.   :50000   Max.   :21.00  
```

Die Funktion `describe`

Eine weitere Möglichkeit noch mehr statistische Kenngrößen für numerische Spalten auszugeben bietet die Funktion `describe` aus dem `psych`-Paket. Nicht-numerische Spalten werden hier nicht vernünftig zusammengefasst, deshalb sollte man der Funktion nur numerische Spalten übergeben.

```
library(psych)
describe(subset(df, select=c("friends", "alcohol", "income", "neurotic")))
```

```
          vars  n    mean       sd  median  trimmed      mad min   max range  skew kurtosis      se
friends      1 10     7.8     6.14     7.5     7.62     7.41   0    17    17  0.11    -1.75    1.94
alcohol      2 10    17.6     7.04    17.5    17.62     3.71   5    30    25 -0.03    -0.75    2.23
income       3 10 18511.0 18001.35 15000.0 16887.50 19940.97  10 50000 49990  0.45    -1.48 5692.53
neurotic     4 10    13.1     3.98    13.0    12.88     2.97   7    21    14  0.36    -0.66    1.26
```

Man kann diese Funktion auch auf einzelne Gruppen separat anwenden. Im Beispiel könnte man dies getrennt für alle Levels von `df$job` tun. Dazu verwendet man die Funktion `describeBy`.

```
describeBy(df[, c("friends", "alcohol", "income", "neurotic")], df$job)
```

```
 Descriptive statistics by group
group: Lecturer
         vars n    mean       sd median trimmed      mad   min   max range skew kurtosis      se
friends     1 5     2.4     2.07      2     2.4     2.97     0     5     5 0.11    -2.03    0.93
alcohol     2 5    16.0     9.62     15    16.0     7.41     5    30    25 0.28    -1.72    4.30
income      3 5 33400.0 12561.85  35000 33400.0 19273.80 20000 50000 30000 0.10    -1.98 5617.83
neurotic    4 5    15.0     4.18     14    15.0     4.45    10    21    11 0.25    -1.72    1.87
------------------------------------------------------------------------------
group: Student
         vars n   mean       sd median trimmed     mad min   max range  skew kurtosis      se
friends     1 5   13.2    2.77     12    13.2    2.97  10    17     7  0.23    -1.89    1.24
alcohol     2 5   19.2    3.56     18    19.2    2.97  16    25     9  0.66    -1.43    1.59
income      3 5 3622.0 4135.69   3000  3622.0 4299.54  10 10000  9990  0.48    -1.64 1849.54
neurotic    4 5   11.2    3.03     13    11.2    1.48   7    14     7 -0.37    -2.01    1.36
```

Das erste Argument ist das zu beschreibende Data Frame, und das zweite Argument ist die Spalte, nach der gruppiert werden soll.

Die Funktion `stat.desc`

Das Paket `pastecs` beinhaltet die Funktion `stat.desc` zur Beschreibung von Daten. Mit der Funktion `round` sollte man einstellen, wie viele Kommastellen ausgegeben werden sollen, da die Ausgabe der Funktion sonst relativ unübersichtlich ist. Wenn das Argument `norm` auf `TRUE` gesetzt wird, werden für alle Spalten Tests auf Normalverteilung durchgeführt.

```
library(pastecs)
round(stat.desc(df[, c("friends", "alcohol", "income", "neurotic")], norm=TRUE), 2)
```

```
             friends alcohol       income neurotic
nbr.val        10.00   10.00        10.00    10.00
nbr.null        1.00    0.00         0.00     0.00
nbr.na          0.00    0.00         0.00     0.00
min             0.00    5.00        10.00     7.00
max            17.00   30.00     50000.00    21.00
range          17.00   25.00     49990.00    14.00
sum            78.00  176.00    185110.00   131.00
median          7.50   17.50     15000.00    13.00
mean            7.80   17.60     18511.00    13.10
SE.mean         1.94    2.23      5692.53     1.26
CI.mean.0.95    4.39    5.04     12877.39     2.85
var            37.73   49.60 324048765.56    15.88
std.dev         6.14    7.04     18001.35     3.98
coef.var        0.79    0.40         0.97     0.30
skewness        0.11   -0.03         0.45     0.36
skew.2SE        0.08   -0.03         0.33     0.26
kurtosis       -1.75   -0.75        -1.48    -0.66
kurt.2SE       -0.66   -0.28        -0.55    -0.25
normtest.W      0.92    0.98         0.90     0.95
normtest.p      0.37    0.94         0.20     0.65
```

Gruppieren mit `by`

Für die Funktion `stat.desc` gibt es keine direkte Variante für gruppierte Daten. Es gibt aber in R die Funktion `by`, welche beliebige Funktionen auf gruppierte Daten anwendet. Das erste Argument ist hier wie üblich der Datensatz, das zweite Argument ist die Gruppierungsspalte, und das dritte Argument ist die Funktion, die auf die gruppierten Daten angewendet werden soll.

```
by(df[, c("friends", "alcohol", "income", "neurotic")], df$job, describe)
```

```
df$job: Lecturer
         vars n    mean       sd median trimmed      mad   min   max range skew kurtosis      se
friends     1 5     2.4     2.07      2     2.4     2.97     0     5     5 0.11    -2.03    0.93
alcohol     2 5    16.0     9.62     15    16.0     7.41     5    30    25 0.28    -1.72    4.30
income      3 5 33400.0 12561.85  35000 33400.0 19273.80 20000 50000 30000 0.10    -1.98 5617.83
neurotic    4 5    15.0     4.18     14    15.0     4.45    10    21    11 0.25    -1.72    1.87
------------------------------------------------------------------------------
df$job: Student
         vars n   mean       sd median trimmed     mad min   max range  skew kurtosis      se
friends     1 5   13.2    2.77     12    13.2    2.97  10    17     7  0.23    -1.89    1.24
alcohol     2 5   19.2    3.56     18    19.2    2.97  16    25     9  0.66    -1.43    1.59
income      3 5 3622.0 4135.69   3000  3622.0 4299.54  10 10000  9990  0.48    -1.64 1849.54
neurotic    4 5   11.2    3.03     13    11.2    1.48   7    14     7 -0.37    -2.01    1.36
```

Möchte man der Funktion im dritten Argument (im folgenden Beispiel `stat.desc`) selbst Argumente übergeben (z.B. `norm=TRUE`), kann man dies mit weiteren Argumenten ganz am Ende tun:

```
by(df[, c("friends", "alcohol", "income", "neurotic")], df$job, stat.desc, norm=TRUE)
```

```
df$job: Lecturer
                friends    alcohol        income   neurotic
nbr.val      5.00000000  5.0000000  5.000000e+00  5.0000000
nbr.null     1.00000000  0.0000000  0.000000e+00  0.0000000
nbr.na       0.00000000  0.0000000  0.000000e+00  0.0000000
min          0.00000000  5.0000000  2.000000e+04 10.0000000
max          5.00000000 30.0000000  5.000000e+04 21.0000000
range        5.00000000 25.0000000  3.000000e+04 11.0000000
sum         12.00000000 80.0000000  1.670000e+05 75.0000000
median       2.00000000 15.0000000  3.500000e+04 14.0000000
mean         2.40000000 16.0000000  3.340000e+04 15.0000000
SE.mean      0.92736185  4.3011626  5.617829e+03  1.8708287
CI.mean.0.95 2.57476927 11.9419419  1.559759e+04  5.1942532
var          4.30000000 92.5000000  1.578000e+08 17.5000000
std.dev      2.07364414  9.6176920  1.256185e+04  4.1833001
coef.var     0.86401839  0.6011058  3.761032e-01  0.2788867
skewness     0.11304669  0.2832618  9.869949e-02  0.2458756
skew.2SE     0.06191822  0.1551489  5.405994e-02  0.1346716
kurtosis    -2.03411574 -1.7235062 -1.980205e+00 -1.7239184
kurt.2SE    -0.50852893 -0.4308766 -4.950513e-01 -0.4309796
normtest.W   0.95235149  0.9787162  9.342460e-01  0.9785712
normtest.p   0.75397300  0.9276364  6.255965e-01  0.9268345
------------------------------------------------------------------------------
df$job: Student
               friends    alcohol        income   neurotic
nbr.val      5.0000000  5.0000000  5.000000e+00  5.0000000
nbr.null     0.0000000  0.0000000  0.000000e+00  0.0000000
nbr.na       0.0000000  0.0000000  0.000000e+00  0.0000000
```

```
min          10.0000000 16.0000000  1.000000e+01  7.0000000
max          17.0000000 25.0000000  1.000000e+04 14.0000000
range         7.0000000  9.0000000  9.990000e+03  7.0000000
sum          66.0000000 96.0000000  1.811000e+04 56.0000000
median       12.0000000 18.0000000  3.000000e+03 13.0000000
mean         13.2000000 19.2000000  3.622000e+03 11.2000000
SE.mean       1.2409674  1.5937377  1.849536e+03  1.3564660
CI.mean.0.95  3.4454778  4.4249254  5.135136e+03  3.7661534
var           7.7000000 12.7000000  1.710392e+07  9.2000000
std.dev       2.7748874  3.5637059  4.135689e+03  3.0331502
coef.var      0.2102187  0.1856097  1.141825e+00  0.2708170
skewness      0.2291423  0.6649718  4.835220e-01 -0.3663862
skew.2SE      0.1255064  0.3642200  2.648359e-01 -0.2006780
kurtosis     -1.8935200 -1.4346010 -1.644573e+00 -2.0145180
kurt.2SE     -0.4733800 -0.3586503 -4.111433e-01 -0.5036295
normtest.W    0.9385501  0.8852008  8.943871e-01  0.8576350
normtest.p    0.6557061  0.3335463  3.796449e-01  0.2198809
```

```r
by(df[, -c(1, 2)], df$job, summary)
```

```
df$job: Lecturer
    friends       alcohol       income         neurotic
 Min.   :0.0   Min.   : 5   Min.   :20000   Min.   :10
 1st Qu.:1.0   1st Qu.:10   1st Qu.:22000   1st Qu.:13
 Median :2.0   Median :15   Median :35000   Median :14
 Mean   :2.4   Mean   :16   Mean   :33400   Mean   :15
 3rd Qu.:4.0   3rd Qu.:20   3rd Qu.:40000   3rd Qu.:17
 Max.   :5.0   Max.   :30   Max.   :50000   Max.   :21
------------------------------------------------------------------------
df$job: Student
    friends        alcohol         income         neurotic
 Min.   :10.0   Min.   :16.0   Min.   :   10   Min.   : 7.0
 1st Qu.:12.0   1st Qu.:17.0   1st Qu.:  100   1st Qu.: 9.0
 Median :12.0   Median :18.0   Median : 3000   Median :13.0
 Mean   :13.2   Mean   :19.2   Mean   : 3622   Mean   :11.2
 3rd Qu.:15.0   3rd Qu.:20.0   3rd Qu.: 5000   3rd Qu.:13.0
 Max.   :17.0   Max.   :25.0   Max.   :10000   Max.   :14.0
```

```r
by(df$friends, df$job, mean)
```

```
df$job: Lecturer
[1] 2.4
------------------------------------------------------------------------
df$job: Student
[1] 13.2
```

Test auf Normalverteilung

Die Funktion `stat.desc` liefert bereits das Ergebnis des Shapiro-Wilk-Tests auf Normalverteilung (die Einträge `normtest.W` und `normtest.p` enthalten den Wert der Teststatistik bzw. die Signifikanz). Wenn `normtest.p` signifikant ist (z.B. kleiner als 0.05), dann kann man die Nullhypothese der Normalverteilung verwerfen. Man kann den Shapiro-Wilk-Test auch direkt mit der Funktion `shapiro.test` aufrufen.

```r
shapiro.test(df$income)
```

```
    Shapiro-Wilk normality test

data:  df$income
W = 0.89721, p-value = 0.2041
```

Mit der by-Funktion kann man den Test auch getrennt auf verschiedene Gruppen anwenden.

```
by(df$income, df$job, shapiro.test)
```

```
df$job: Lecturer

    Shapiro-Wilk normality test

data:  dd[x, ]
W = 0.93425, p-value = 0.6256

------------------------------------------------------------------------
df$job: Student

    Shapiro-Wilk normality test

data:  dd[x, ]
W = 0.89439, p-value = 0.3796
```

Der Kolmogorov-Smirnov-Test kann gegebene Daten auf beliebige Verteilungen testen, d.h. natürlich auch auf Normalverteilung. Im Falle der Normalverteilung ist aber der Shapiro-Wilk-Test vorzuziehen, da dieser speziell auf die Normalverteilung zugeschnitten ist und daher mehr statistische Power besitzt.

```
ks.test(df$income, "pnorm", mean(df$income), sd(df$income))
```

```
    One-sample Kolmogorov-Smirnov test

data:  df$income
D = 0.18182, p-value = 0.8389
alternative hypothesis: two-sided
```

Da die Stichprobengröße in unserem Beispiel nur sehr klein ist, lassen sich aber ohnehin keine vernünftigen Aussagen über die Verteilung der Daten treffen.

Test auf Varianzhomogenität

Der Levene-Test prüft auf Gleichheit der Varianzen (Homoskedastizität) von zwei oder mehr Gruppen. Die Nullhypothese ist, dass die Varianzen in allen Gruppen gleich sind. In R führt man den Test mit der Funktion `leveneTest` aus dem Paket `car` durch. Dazu sehen wir uns die Beispieldaten `Moore` an, welche mit dem Paket `car` automatisch geladen werden.

```
library(car)
?Moore
head(Moore, 4)
```

```
  partner.status conformity fcategory fscore
1            low          8       low     37
2            low          4      high     57
3            low          8      high     65
4            low          7       low     20
```

```r
tail(Moore, 4)
```

```
   partner.status conformity fcategory fscore
42           high         13      high     57
43           high         16       low     35
44           high         10      high     52
45           high         15    medium     44
```

Der Levene-Test für die Spalte `conformity` gruppiert nach der Spalte `fcategory` wird wie folgt aufgerufen:

```r
leveneTest(Moore$conformity, Moore$fcategory)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2   0.046 0.9551
      42
```

In diesem Beispiel kann die Nullhypothese der Varianzgleichheit der Spalte `conformity` in den Gruppen `fcategory` also nicht verworfen werden.

Übungen

Übung 1

Berechnen Sie statistische Kenngrößen wie Mittelwert, Median, Minimum und Maximum für die vier numerischen Spalten `Global_active_power`, `Global_reactive_power`, `Voltage` und `Global_intensity` aus dem Data Frame aus Übung 3 der vorigen Einheit (Individual Household Electric Power Consumption).

- Berechnen Sie die Kenngrößen mit der Funktion `sapply`.
- Berechnen Sie die obigen Kenngrößen mit der Funktion `summary`.
- Wie groß ist die gemessene mittlere Spannung `Voltage`?
- Wie groß ist der Median der globalen Wirkleistung `Global_active_power`?
- Sehen Sie sich die Ausgabe von `describe` aus dem Paket `psych` an.
- Wenden Sie die Funktion `stat.desc` aus dem Paket `pastecs` auf die Daten an (runden Sie die Ergebnisse auf eine Nachkommastelle).

Übung 2

Einer der bekanntesten Datensätze im Bereich Machine Learning ist der Iris-Datensatz von R. A. Fisher. Er beschreibt drei verschiedene Spezies einer Iris-Pflanze. Die Daten sind standardmäßig in R im Data Frame `iris` verfügbar.

- Wie viele Pflanzen gibt es in dem Datensatz?
- Wie viele Merkmale wurden pro Pflanze erhoben?
- Berechnen Sie deskriptive Statistiken aller Spalten und vergleichen Sie Ihre Ergebnisse vom Mittelwert, Standardabweichung, Minimum und Maximum mit der Beschreibung auf der UCI Machine Learning Datenbank. Gibt es Unterschiede?
- Berechnen Sie die Mittelwerte der vier Merkmale getrennt für jede der drei Iris-Arten.

Diese Unterlagen sind lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.