Statistik mit Octave und MATLAB

Alle im folgenden aufgeführten Befehle sollten unter dem freien Programm Octave und dem kommerziellen Programm MATLAB der Firma MathWorks in ähnlicher Weise funktionieren, soweit hier nicht anders beschrieben. Leider scheinen sich die wenigen Unterschiede im Bereich der statistischen Funktionen etwas zu häufen...

1. Direkte Dateneingabe

Die statistischen Funktionen operieren zumeist auf Spaltenvektoren oder Matrizen. Einen Spaltenvektor kann man in der Form

```
x = [1.1; 2; 1.2; 1.8]
```

direkt eingeben. Als schreibfauler Physiker wird man aber vermutlich die Eingabe eines Zeilenvektors mit nachfolgendem Transponieren (mittels ') bevorzugen, um die Semikola einzusparen:

```
x = [1.1 2 1.2 1.8]'
```

Beides führt zum Resultat:

```
x =
   1.1000
   2.0000
   1.2000
   1.8000
```

Eine Matrix wird zeilenweise eingegeben, wobei die Zeilen jeweils durch ein Semikolon getrennt werden.

```
A = [1 2 3 4; 2 3 4 1; 0 0 0 5]
```

Alternativ kann man die Einträge in einer Matrixzeile auch durch Kommata trennen:

```
A = [1,2,3,4; 2,3,4,1; 0, 0, 0, 5]
```

Beides liefert die 3x4 Matrix

```
A =
   1   2   3   4
   2   3   4   1
   0   0   0   5
```

2. Einlesen der Daten aus einer Datei

Üblicherweise liegen die zu untersuchenden Daten in einer Datei (z.B. `Dat.dat`) vor, wobei jeder Wert in einer Zeile steht. Die Daten werden im einfachsten Fall mit

```
load Dat.dat
```

eingelesen. Gelegentlich wird auch

```
load -ascii Dat.dat
```

empfohlen, was aber in allen mir zugänglichen Versionen nicht erforderlich ist. Hiernach befinden sich die Daten im Spaltenvektor `Dat`. Mit dem Kommando

```
whos
```

können Sie feststellen, wie groß der Datensatz ist. Durch Eingeben des Namens wird der Inhalt angezeigt. Bei richtig großen Datensätzen, wie den Wetterdaten von Helgoland, sollte man sich z.B. mit

```
Dat(1:10)
```

auf eine Inspektion der ersten Elemente beschränken.

Falls Sie einen etwas länglichen Dateinamen verwendet haben, empfiehlt sich vor der eigentlichen Rechnung ein Umbenennen, z.B.

```
load 'Datei mit langemNamen.dat'
x = Datei_mit_langemNamen;
```

wobei Octave die Leerzeichen durch _ ersetzt hat, um einen gültigen Namen zu erzeugen.

Es sind natürlich auch Dateien mit mehreren Spalten möglich, wobei jeder Spalte ein Merkmal entspricht, z.B. Alter, Größe und Gewicht. Beim Einlesen mit

```
load XYZ.dat
```

würde man in diesem Fall eine nx3 Matrix erhalten. Mit den Anweisungen

```
X = XYZ(:,1);
Y = XYZ(:,2);
Z = XYZ(:,3);
```

kann man daraus 3 einzelne Spaltenvektoren erzeugen.

2. Ausgabe der Daten in eine Datei

Häufig möchte man einen Vektor oder eine Matrix, die man mühsam oder trickreich erzeugt hat, für die Weiterverarbeitung (z.B. mit gnuplot oder einer Textverarbeitung) in einer Datei speichern. (Im richtigen Leben sind die Datensätze oft so groß, dass *Drag&Drop* keine Option mehr ist...)

Zum Speichern des Vektors (oder der Matrix) X in der Datei `Dat.dat` kann man den Befehl

```
save -ascii Dat.dat X
```

benutzen. Hat man mehrere zusammengehörige Spalten, z.B, x, y und dy, so kann man Sie wie folgt als Matrix mit 3 Spalten speichern:

```
M = [x y dy];  save -ascii Dat.dat M
```

3 . Deskriptive Statistik

Im folgenden wird angenommen, dass x,y,z Spaltenvektoren sind. Viele Befehle können auch 'spaltenweise' auf Matrizen operieren. Wenn man diese und andere fortgeschrittene Techniken benutzen möchte, sollte man sich die zugehörigen Informationen mit dem eingebauten Befehl `help` ansehen.

3.1 Mittelwerte

Zweifellos die wichtigste Operation ist die Bestimmung des **arithmetischen Mittels**:

```
mean(x)
```

Im Programm Octave können zusätzlich auch das geometrische und das harmonische Mittel berechnet werden:

```
mean(x,"g")
mean(x,"h")
```

Mit der Funktion `meansq()` (nur in Octave) bestimmt man den Mittelwert der Quadrate, d.h. $\frac{1}{n}\sum_i x_i^2$, so dass man das **quadratische Mittel** $\bar{x}_{\mathrm{rms}}$ mit

```
x_rms = sqrt(meansq(x))
```

erhält. Das kann man natürlich auch ebensogut mit

```
x_rms = sqrt(mean(x.*x))
```

ausrechnen.

Auch für den in der Physik weniger gebräuchlichen **Median** und den **Modus** gibt es die Funktionen `median(x)` und `mode(x)` . Die Bestimmung des Median ist im Übrigen nicht so einfach (`help median`); es gibt hier auch leichte Variationen in der Definition bei anderen Programmpaketen.

Beispiel:

```
x = [1 2 4 1 2 4 2]';
mean(x)                 -> ans = 2.2857
mean(x,"g")             -> ans = 2      (nur Octave)
mean(x,"h")             -> ans = 1.7500 (nur Octave)
meansq(x)               -> ans = 6.5714 (nur Octave)
sqrt(meansq(x))         -> ans = 2.5635 (nur Octave)
sqrt(mean(x.*x))        -> ans = 2.5635
median(x)               -> ans = 2
mode(x)                 -> ans = 2
```

3.2 Varianz und Standardabweichung

Bei Varianz und Standardabweichung muss man zwischen den Größen für die Grundgesamtheit (alle möglichen Daten)

$$V(x) = \sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

und der erwartungstreuen Schätzung der Varianz aus einer Stichprobe

$$s^2 = \sigma_{n-1}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

unterscheiden. Die in der Praxis häufiger benötigten Schätzgrößen aus der **Stichprobe** erhält man mit
`var(x)` bzw. `std(x)` .
Die Normierung auf n erhält man mit
`var(x,1)` bzw. `std(x,1)` .

Es gibt auch Funktionen für die Schiefe (skewness) und Kurtosis (kurtosis) (bei MATLAB sogar mehrere Varianten), die sich in den Definitionen jedoch unterscheiden. Wenn man tatsächlich darauf zurückgreifen muss oder will, ist ein genaues Studium der jeweiligen Definition unumgänglich. Hier ist es eventuell besser, eine selbstdefinierte Funktion zu verwenden, damit man beim Wechsel zwischen den Programmen und Programmversionen keine Überraschungen erlebt. Glücklicherweise kommen diese beiden Funktionen in der Physik etwa so häufig vor wie der Große Panda.

Nützlich sind auch die Funktionen `min(x)`, `max(x)` und `range(x)`, die den kleinsten Wert, den größten Wert, sowie die Spannweite (Maximum - Minimum) liefern.

Beispiel:

```
x = [1 2 4 1 2 4 2]';
var(x)                  -> ans = 1.5714
std(x)                  -> ans = 1.2536
var(x,1)                -> ans = 1.3469
std(x,1)                -> ans = 1.1606

min(x)                  -> ans = 1
max(x)                  -> ans = 4
range(x)                -> ans = 3
```

3.3 Paare von Daten: Kovarianz und Korrelation

Betrachtet werden Paare von Daten (x_i, y_i) die in den Spaltenvektoren x und y angeordnet sind. Die **Kovarianzmatrix** kann dann mit `cov([x y])` bestimmt werden, wobei definitionsgemäß die Varianzen der einzelnen Variablen auf der Diagonale stehen, während die Nicht-Diagonalelemente die Kovarianzen zwischen den jeweiligen Variablen sind.
Ähnlich wie bei der Varianz gibt es eine Funktion für die Schätzung aus Stichproben (erwartungstreu, normiert auf $\frac{1}{n-1}$) und eine Variante `cov([x y],1)`, die auf $\frac{1}{n}$ normiert ist.
Gibt man bei Oktave keine Matrix sondern nur die Vektoren an, d.h. `cov(x,y)` anstelle von `cov([x y])` so wird lediglich die Kovarianz als Zahl geliefert.
Die Funktion `corr([x y])` liefert die Matrix der Korrelationskoeffizienten.
Beide Funktionen können natürlich mit mehr als zwei Variablen (Spalten) arbeiten.

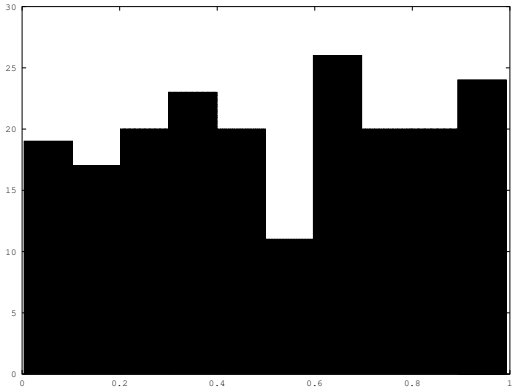
Beispiele:

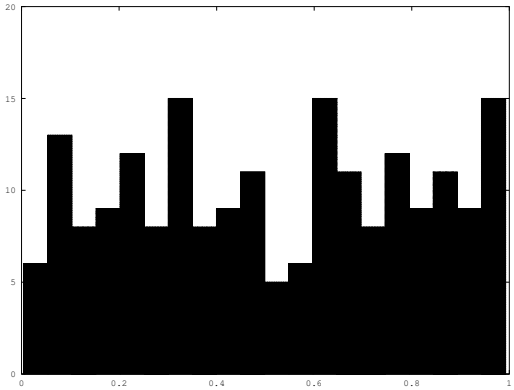
```
x = [1 2 3 4 5]'
y = [2.1 2.9 3.2 4.1 4.8]'
cov([x y])       -> ans =
                      2.5000   1.6500
                      1.6500   1.1070

cov([x y],1)     -> ans =
                      2.00000   1.32000
                      1.32000   0.88560

cov(x,y)         -> ans = 1.6500    (nur Octave)

corr([x y])      -> ans =
                      1.00000   0.99184
                      0.99184   1.00000

z = x+y;
corr([x y z])    -> ans =
                       1.00000   0.99184   0.99870
                       0.99184   1.00000   0.99705
                       0.99870   0.99705   1.00000
```

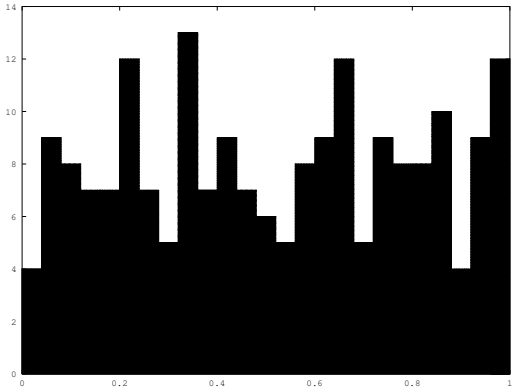
3.4 Histogramme

Mit dem Befehl `hist(X)` kann auf einfache Weise ein Histogramm aus einem Vektor X von Messwerten erzeugt werden. In dieser einfachsten Form wird die Klassenbreite automatisch gewählt.
In der Form `hist(X,n)` wird die Klassenbreite so gewählt, dass insgesamt n Klassen entstehen. Dies ist die geeignete Methode für einen schnellen Überblick und das Experimentieren mit verschiedenen Breiten. Leider ergeben sich dabei nur selten "schöne Werte" für die Klassengrenzen und -breiten.
Die beste Kontrolle erhält man, wenn man als zweites Argument einen Vektor mit den gewünschten Klassenmitten angibt. Die Klassenbreiten ergeben sich dabei aus den Abständen dieser Werte.
Zum Testen wird hier mit dem Befehl rand(200,1) eine 200x1 Matrix, also ein Spaltenvektor mit Zufallszahlen erzeugt.

```
X = rand(200,1);
hist(X)
hist(X,20)
hist(X,[0.02:0.04:1])
```







Die erhaltenen Bilder wurden jeweils mit dem Befehl
`print -deps dateiname`
abgespeichert und später in diesen Text integriert. Beim Verkleinern ist die Schrift unzumutbar klein geworden, was man durch entsprechende Manipulationen vor dem `print`-Befehl verbessern könnte. Das ist aber ein anderes Thema...

Wenn man die Histogramm-Daten weiterverarbeiten möchte, empfiehlt sich eine der folgenden Varianten:

```
N = hist(X); [N XM] = hist(X);
```

Hierbei werden die Anzahlen der Ereignisse in den einzelnen Klassen im Zeilen(!)vektor N gespeichert. Der Zeilenvektor XM enthält die Klassenmitten der zugehörigen Klasseneinteilung. Will man das Histogramm mit anderen Programmen weiterverarbeiten, so kann man es durch

```
M = [N' XM'];
save -ascii histo.dat M
```

in geeigneter Form abspeichern.

Die Möglichkeit, Histogramme aus der Liste der Beobachtungen zu erstellen, ist eine nützliche Eigenschaft, da viele Auswerte- und Grafikprogramme (auch gnuplot) davon ausgehen, dass das Histogramm bereits in dieser Tabellenform vorliegt.