Statistik und Wahrscheinlichkeitsrechnung S06 - Siegel Hannah - 5AHITT

0.1 Bloecke

```
(%i1)  kill(all);
       load(descriptive)$
       load(draw)$
```

(%o0) *done*

Der Block rand_col returnt eine schoene farbe nach zufallsprinzip. intnum ... 1 = pink - ton 2 = blau - ton 3 = grau - ton

```
(%i3)  rand_col(intnum):=block(
           colors[1]:[magenta,dark-magenta, violet, dark-violet, plum, purple, light-pin
           colors[2]:[cyan,dark-blue,skyblue,navy,light-blue,blue,dark-cyan],
           colors[3]:[gray10,gray20,gray30,gray40,gray50,gray60,gray70,gray80,gray90],
           colors[intnum][random(length(colors[intnum])-1)+1]
       )$
```

Der Block prepare_list nimmt die basis liste der werte (urliste in dem Format [[Fehler1, Fehler-Bezeichnung1],[Fehler2, FehlerBezeichnung2],..]) und formt sie auf [[1,2,3...],[FehlerBezeichnung1 ,FehlerBezeichnung5, FehlerBezeichnung3,...],[Fehler1, Fehler5, Fehler3, ...]] um. Weiters wird die Liste nach der groesse der Fehler sortiert. Prinzipiell wird dieser Block eigenstaendig von draw_pareto und draw_piechart aufgerufen.

```
(%i4)  prepare_list(list_angabe):=block(
           list_angabe:sort(list_angabe,ordergreatp),
           angabe_beschriftung:[],
        angabe_wert:[],
           angabe_inkr:[],
           for i:1 while i<= length(list_angabe) do [
               angabe_beschriftung: append(angabe_beschriftung,[list_angabe[i][2]]),
               angabe_wert: append(angabe_wert,[list_angabe[i][1]]),
               angabe_inkr: append(angabe_inkr,[i])
           ],
           list_angabe:[angabe_inkr,angabe_beschriftung,angabe_wert]
       )$
```

Der Block draw_pareto nimmt die basis liste der werte (urliste in dem Format [[Fehler1, Fehler-Bezeichnung1],[Fehler2, FehlerBezeichnung2],..]) und zeichnet ein Paretto Diagramm

```
(%i5)  draw_pareto(list_angabe):=block(
         list_angabe:prepare_list(list_angabe),
         wxdraw2d(
               xrange=[0,length(list_angabe[3])+1],
               yrange=[lmin(list_angabe[3])-1,lmax(list_angabe[3])+1],
               xaxis=true,
               yaxis=true,
               grid=true,
               xtics=1,
               ytics=5,
               points_joined = impulses,
               color=rand_col(1),
               line_width=10,
               points(list_angabe[1],list_angabe[3]),
               xlabel="Fehler", ylabel="Anzahl"
          )
       )$
```

Der Block draw_piechart nimmt die basis liste der werte (urliste in dem Format [[Fehler1, FehlerBezeichnung1],[Fehler2, FehlerBezeichnung2],..]) und zeichnet ein Torten Diagramm

```
(%i6)  draw_piechart(list_angabe):=block(
         list_angabe:prepare_list(list_angabe),
         list:[],
         for i:1 while i <= length(list_angabe[1]) do [
           list: append(list, makelist(list_angabe[2][i],j,1,list_angabe[3][i]))
         ],
         wxpiechart(
               list,
               proportional_axes=xy,
               xrange=[-1.03,3],
               yrange=[-1.03,1.03],
               xtics=false,
               ytics=false,
               sector_colors = [magenta,dark-magenta, violet, dark-violet, plum, purple,
               title="Fehler"
           )
       )$
```

Der Block klassenbildung hat folgende eingangsparameter: messwerte: messwerte steps: Klassenbreite W anzahl_klassen: anzahl der zu bildenden Klassen Folgende ausgangsparameter: klassen: liste der gebildeten Klassen end_list: neue liste im Format: [1,2,6,2,8,3,5,1,5,7,8,3,7,...]

```
(%i7)  klassenbildung(messwerte,steps, anzahl_klassen):=block(
         xmin: lmin(messwerte),
         xmax: lmax(messwerte),
         klassen:[[0,xmin+2*steps]],
           xmin:xmin+2*steps,
         for j:1 while j<anzahl_klassen do [
           neuer_kleinstwert:xmin+steps,
           klassen:append(klassen, [[xmin,neuer_kleinstwert]]),
           xmin:neuer_kleinstwert
         ],
         klassen[anzahl_klassen][2]:100000,
         for n:1 while n<=length(klassen) do [
           class[n]:[]
         ],
         end_list:[],
         for i:1 while i<=length(messwerte) do [
           for j:1 while j<= length(klassen) do [
               if (messwerte[i] > klassen[j][1] ) then if( messwerte[i] <= klassen[j][2]
           ]
         ]
       )$
```

Der Block get_labels stueckelt die Klassen in ein labels object, welches nachher dann in die Grafik eingebunden werden kann.

klassen ... Klassen welche nach dem Aufruf des klassenbildungs-blockes verfuegbar sind delta_minus ... Platz zum Zeichnen der Labels in die negative Richtung

```
(%i8)  get_labels(klassen,delta_minus):=block(
         a:label(),
         int_place1:-(delta_minus*0.25),
         int_place2:-(delta_minus*0.75),
         for i:1 while i<= length(klassen) do [
               num:concat("[",klassen[i][1],",",klassen[i][2],"]"),
               if (evenp(i)=true) then a: append(a,label([num,i,int_place1])),
               if (evenp(i)=false) then a: append(a,label([num,i,int_place2]))
         ],
         label_arr:a
       )$
```

draw_normal_generalized — draw_normal_g

Der draw_normal_generalized Block, zeichnet die normalen Grafiken der Statistik.

Er wird von den anderen draw_ Bloecken aufgerufen und sollte fuer den User niemals von Bedeutung sein.

draw_normal_generalized(ytics_value,title_grafik,ylabel_grafik,balken,list_discrete)

title_grafik ... Titel der Grafik ylabel_grafik ... Beschriftung der y_achse (ob Prozente oder Anzahl) balken ... bars() object ytics_value ... Abstand der Werte auf der y-Achse (Default:5) list_discrete ... die discrete liste, zum berechnen der ranges benoetigt

```
(%i9)  draw_normal_generalized(ytics_value,title_grafik,ylabel_grafik,balken,list_discre
            wxdraw2d(
               xrange=[lmin(list_discrete[1])-1,lmax(list_discrete[1])+1],
               yrange=[0,ceiling (lmax(list_discrete[2])+(lmax(list_discrete[2])/10))],
               xaxis=true, yaxis=true,
               grid=true,
               xtics=1,
               ytics=ytics_value,
               fill_color=rand_col(1),
               balken,
               title=title_grafik,
               xlabel="Messwert", ylabel=ylabel_grafik
            )
       )$
```

——————————————— draw_klassen_generalized ——————————————— draw_klassen_g

Der draw_klassen_generalized Block, zeichnet die normalen Grafiken der Statistik.

Er wird von den anderen draw_ Bloecken aufgerufen und sollte fuer den User niemals von Bedeutung sein. draw_normal_generalized(xdimension,ydimension,ylabel_grafik,title_grafik,list_discrete,labels_list title_grafik ... Titel der Grafik ylabel_grafik ... Beschriftung der y_achse (ob Prozente oder Anzahl) balken ... bars() object ytics_value ... Abstand der Werte auf der y-Achse list_discrete ... die discrete liste, zum berechnen der ranges benoetigt xdimension ... Groesse der Zeichnung nach x (Default:500) ydimension ... Groesse der Zeichnung nach y (Default:300) labels_list ... labels() object delta_minus ... POSITIVER Wert, range fuer das anzeigen der Labels (sollte meist zwischen ca 5 und 20 liegen)

```
(%i10) draw_klassen_generalized(xdimension,ydimension,ylabel_grafik,title_grafik,list_di
           wxdraw2d(
              xrange=[lmin(list_discrete[1])-1,lmax(list_discrete[1])+1],
              yrange=[-delta_minus,ceiling (lmax(list_discrete[2])+(lmax(list_discrete[
              xaxis=false,
              yaxis=false,
              grid=true,
              xtics=false,
              ytics=ytics_value,
              color=black,
              line_type=solid,
              font_size=11,
              font="Arial",
              dimensions = [xdimension, ydimension],
              labels_list,
              font_size=10,
              fill_color=rand_col(1),
              balken,
              xlabel="Klassen",
              ylabel=ylabel_grafik,
              title=title_grafik
           )
       )$
```

draw_<TYPE>_klassen

draw_TYPE_klassen Jeder draw_TYPE_klassen Block, macht die Klasseneinteilung sowie das Zeichnen mit den Labels automatisch. Die Bloecke dienen nur zur abstraktion sowie hilfe fuer den User, um die verwendung des tatsaechlichens draw-Blocks zu vermeiden

draw_TYPE_klassen(messwerte,steps,anzahl_klasse,ytics,delta_minus,xdimension,ydimension)

messwerte ... Eine Liste der messwerte im Format [Messwert1, Messwert2, ...] steps ... Die Klassenbreite W anzahl_klassen ... Anzahl der Klassen ytics ... Abstand der Werte auf der y-Achse delta_minus ... POSITIVER Wert, range fuer das anzeigen der Labels (sollte meist zwischen ca 5 und 20 liegen) xdimension ... Groesse der Zeichnung nach x (Default:500) ydimension ... Groesse der Zeichnung nach y (Default:300)

```
(%i11) draw_KUM_ABS_HFG_klassen(messwerte,steps,anzahl_klassen,ytics_value,delta_minus,g
         klassenbildung(messwerte,steps, anzahl_klassen),
         if (delta_minus = 0) then delta_minus:15,
         get_labels(klassen,delta_minus),
         draw_KUM_ABS_HFG(end_list,ytics_value,false,label_arr,delta_minus,groesse1,groe
       )$
```

```
(%i12) draw_RELATIVE_HFG_klassen(messwerte,steps,anzahl_klassen,ytics_value,delta_minus,
         klassenbildung(messwerte,steps, anzahl_klassen),
         if (delta_minus = 0) then delta_minus:5,
         get_labels(klassen,delta_minus),
         draw_RELATIVE_HFG(end_list,ytics_value,false,label_arr,delta_minus,groesse1,gro
       )$

(%i13) draw_KUM_REL_HFG_klassen(messwerte,steps,anzahl_klassen,ytics_value,delta_minus,g
         klassenbildung(messwerte,steps, anzahl_klassen),
         if (delta_minus = 0) then delta_minus:20,
         get_labels(klassen,delta_minus),
         draw_KUM_REL_HFG(end_list,ytics_value,false,label_arr,delta_minus,groesse1,groe
       )$

(%i14) draw_ABSOLUTE_HFG_klassen(messwerte,steps,anzahl_klassen,ytics_value,delta_minus,
         klassenbildung(messwerte,steps, anzahl_klassen),
         if (delta_minus = 0) then delta_minus:5,
         get_labels(klassen,delta_minus),
         draw_ABSOLUTE_HFG(end_list,ytics_value,false,label_arr,delta_minus,groesse1,gro
       )$
```

draw_<TYPE>_normal

draw_TYPE_normal Jeder draw_TYPE_normal Block, ruft den komplitzierten draw Block auf, und uebergibt ihm die richtigen Parameter

draw_TYPE_normal(list_angabe,ytics_value)

list_angabe ... Eine Liste der messwerte im Format [Messwert1, Messwert2, ...] ytics_value ... Abstand der Werte auf der y-Achse (Default:5)

```
(%i15) draw_KUM_ABS_HFG_normal(list_angabe,ytics_value):=block(
         draw_KUM_ABS_HFG(list_angabe,ytics_value,true,0,0,0,0)
       )$

(%i16) draw_KUM_REL_HFG_normal(list_angabe,ytics_value):=block(
         draw_KUM_REL_HFG(list_angabe,ytics_value,true,0,0,0,0)
       )$

(%i17) draw_ABSOLUTE_HFG_normal(list_angabe,ytics_value):=block(
         draw_ABSOLUTE_HFG(list_angabe,ytics_value,true,0,0,0,0)
       )$

(%i18) draw_RELATIVE_HFG_normal(list_angabe,ytics_value):=block(
         draw_RELATIVE_HFG(list_angabe,ytics_value,true,0,0,0,0)
       )$
```

draw_TYPE Jeder draw_TYPE Block, berechnet und zeichnet die jeweiligen Darstellungen. Jeder draw_TYPE Block kann sowohl die normale als auch die komplizierte Klassen darstellung zeichnen.

draw_TYPE_klassen(messwerte,ytics,normal_bool,labels,delta_minus,xdimension,ydimension)

messwerte ... Eine Liste der messwerte im Format [Messwert1, Messwert2, ...] normal_bool ... true = normal ; false = mit labels und allem dazugehoerigen berechnungen labels_list ... Liste der Labels welche nach dem Aufruf get_labels vorhanden ist. ytics ... Abstand der Werte auf der y-Achse delta_minus ... POSITIVER Wert, range fuer das anzeigen der Labels (sollte meistens zwischen ca 5 und 20 liegen) xdimension ... Groesse der Zeichnung nach x (Default:500) ydimension ... Groesse der Zeichnung nach y (Default:300)

```
(%i19) draw_ABSOLUTE_HFG(list_angabe,ytics_value,normal_bool,labels_list,delta_minus,xdi
           list_discrete:discrete_freq(list_angabe),
           r:bars(),
           if (xdimension = 0) then xdimension:500,
           if (ydimension = 0) then ydimension:300,
           if (ytics_value = 0) then ytics_value:5,
           for i:1 while i<=length(list_discrete[1]) do [
               r: append(r, bars([list_discrete[1][i],list_discrete[2][i],0.5]))
           ],
           if (normal_bool=true) then
               draw_normal_generalized(ytics_value,"Absolute Haeufigkeit","Anzahl",r,lis
           else
           if (normal_bool=false) then
               draw_klassen_generalized(xdimension,ydimension,"Anzahl","Absolute Haeufig
       )$
```

```
(%i20) draw_KUM_ABS_HFG(list_angabe,ytics_value,normal_bool,labels_list,delta_minus,xdim
           list_discrete:discrete_freq(list_angabe),
           if (xdimension = 0) then xdimension:500,
           if (ydimension = 0) then ydimension:300,
           if (ytics_value = 0) then ytics_value:10,
           neu1:[],
           neu2:[],
           found:1,
           sum:0,
           neue_werte:[],
           for n:1 while n<=length(list_discrete[2]) do [
               neuer_wert: list_discrete[2][n]+sum,
               sum : sum + list_discrete[2][n],
               neue_werte:append(neue_werte,[neuer_wert])
           ],
           abs_hf:[list_discrete[1],neue_werte],
           for k:lmin(abs_hf[1]) while k<=lmax(abs_hf[1]) do [
               neu1:append(neu1,[k]),
               if ((member(k,abs_hf[1]))) then neu2:append(neu2,[abs_hf[2][found]]),
               if ((member(k,abs_hf[1]))) then found:found+1,
               if (not(member(k,abs_hf[1]))) then neu2:append(neu2,[neu2[k-lmin(abs_hf[1
           ],
           list_discrete:[neu1,neu2],
           r:bars(),
           for i:1 while i<=length(list_discrete[1]) do [
               r: append(r, bars([list_discrete[1][i],list_discrete[2][i],0.5]))
           ],
           if (normal_bool = true) then
               draw_normal_generalized(ytics_value,"Kummulative Absolute Haeufigkeit","A
           else
           if (normal_bool = false) then
             draw_klassen_generalized(xdimension,ydimension,"Anzahl","Kummulative Absolu
       )$
```

```
(%i21) draw_KUM_REL_HFG(list_angabe,ytics_value,normal_bool,labels_list,delta_minus,xdim
          list_discrete:discrete_freq(list_angabe),
           if (xdimension = 0) then xdimension:500,
           if (ydimension = 0) then ydimension:300,
           if (ytics_value = 0) then ytics_value:10,
           prozente:[],
           for n:1 while n<=length(list_discrete[2]) do [
               prozent: (100/length(list_angabe)) * list_discrete[2][n],
               prozent: round(prozent),
               prozente:append(prozente,[prozent])
           ],
           list_discrete:[list_discrete[1],prozente],
           neu1:[],
           neu2:[],
           found:1,
           sum:0,
           neue_werte:[],
           for n:1 while n<=length(list_discrete[2]) do [
               neuer_wert: list_discrete[2][n]+sum,
               sum : sum + list_discrete[2][n],
               neue_werte:append(neue_werte,[neuer_wert])
           ],
           abs_hf:[list_discrete[1],neue_werte],
           for k:lmin(abs_hf[1]) while k<=lmax(abs_hf[1]) do [
               neu1:append(neu1,[k]),
               if ((member(k,abs_hf[1]))) then neu2:append(neu2,[abs_hf[2][found]]),
       if ((member(k,abs_hf[1]))) then found:found+1,
               if (not(member(k,abs_hf[1]))) then neu2:append(neu2,[neu2[k-lmin(abs_hf[1
         ],
           list_discrete:[neu1,neu2],
           r:bars(),
           for i:1 while i<=length(list_discrete[1]) do [
               r: append(r, bars([list_discrete[1][i],list_discrete[2][i],0.5]))
           ],
           if (normal_bool=true) then
               draw_normal_generalized(ytics_value,"Kummulative Relative Haeufigkeit","P
           else
           if (normal_bool=false) then
               draw_klassen_generalized(xdimension,ydimension,"Prozent","Kummulative Rel
       )$
```

```
(%i22) draw_RELATIVE_HFG(list_angabe,ytics_value,normal_bool,labels_list,delta_minus,xdi
           prozente:[],
           if (xdimension = 0) then xdimension:500,
           if (ydimension = 0) then ydimension:300,
           if (ytics_value = 0) then ytics_value:5,
           list_discrete:discrete_freq(list_angabe),
           for n:1 while n<=length(list_discrete[2]) do [
               prozent: (100/length(list_angabe)) * list_discrete[2][n],
               prozent: float(prozent),
               prozent: round(prozent),
               prozente:append(prozente,[prozent])
           ],
           list_discrete:[list_discrete[1],prozente],
           r:bars(),
           for i:1 while i<=length(list_discrete[1]) do [
               r: append(r, bars([list_discrete[1][i],list_discrete[2][i],0.5]))
           ],
           if (normal_bool = true) then
               draw_normal_generalized(ytics_value,"Relative Haeufigkeit","Prozent",r,li
           else
           if (normal_bool = false) then
               draw_klassen_generalized(xdimension,ydimension,"Prozent","Relative Haeufi
       )$
```

Block fuer das Zeichnen von Punkten als Punkte: list_angabe ... angabe der Punkte

```
(%i23) paint_TYPE1_points(list_angabe):=block(
           return_points:[],
           for i:1 while i<=length(list_angabe) do [
               return_points:append(return_points,[[i,list_angabe[i]]])
           ],
           wxplot2d([[discrete,return_points]],[x,0,length(return_points)+1],[y,lmin(ang
       )$
```

Block fuer das Zeichnen von Punkten als Balken: list_angabe ... angabe der Punkte

```
(%i24) paint_TYPE1_stricherl(list_angabe):=block(
           first:[],
           second:[],
           for i:1 while i<=length(angabe) do [
               first: append(first,[angabe[i]]),
               second: append(second,[i])
           ],
           return_stuff:[second,first],
           wxdraw2d(
               xrange=[lmin(return_stuff[1])-1,lmax(return_stuff[1])+1],
               yrange=[lmin(return_stuff[2])-1,lmax(return_stuff[2])+1],
               xaxis=true,
               yaxis=true,
               grid=true,
               xtics=1,
               ytics=2,
               points_joined = impulses,
               color=rand_col(1),
               line_width=10,
               points(return_stuff[1],return_stuff[2]),
               xlabel="Größe Merkmal", ylabel="Anzahl"
          )
       )$
```

1 Glossar

BESCHREIBENDE STATISTIK:
Untersucht eine STICHPROBE und davon ein MERKMAL

MERKMALE werden unterteilt man in Qualitative(eg Farbe eines Autos) und Quantitive( )

Groessere Eimheit aus welcher die Stichprobe kommt ist die GRUNDGESAMMTHEIT

STICHPROBE:
* Merkmalswerte & KENNGROESSEN
* Darstellung (sowohl mit als auch ohne Kenngroessen als graphik oder tabelle)
=> Beischreibung -> 'beschreibende Statistik'

URLISTE
Die Urliste gilt als eingangsparameter, sie sollte jedoch immer UNVERAENDERT behalten

werden, daher wird sie einfach kopiert.

ABSOLUTE HAEUFIGKEIT
Die Absolute Haeufigkeit zeichnet auf, wie oft (=Anzahl) welcher Messwert (=Groesse) vorkommt.
absolute Haeufigkeit = relative Haeufigkeit * Anzahl der Erhebungen

RELATIVE HAEUFIGKEIT
Die Relative Haeufigkeit zeigt auf, zu wie viel Prozent welcher Messwert (=Groesse)vorkommt.
relative Haeufigkeit = absolute Haeufigkeit / Anzahl der Erhebungen

KUMMULATIV
Kummulative Haeufigkeiten zeigen auf, wie oft welcher Messwert vorkommt.
Man addiert immer die vorherigen Messwerte zu dem neuen dazu.
Dadurch wird im endeffekt immer die anzahl der messwerte (absolut) oder 100% (reativ) erreicht.
Durch die Kummulative Haeufigkeit entstehen neue Moeglichkeiten, die Messwerte zu interpretieren.

PARETTODIAGRAMM
ein Pareto Diagramm ist ein Balkendiagramm
y-Achse : Absolute Haeufigkeit
x-Achse : Fehler nach Wichtigkeit geordnet

KREISDIAGRAMM
ein Kreisdiagramm Diagramm (oder auch Tortendiagramm) zeigt die verteilung der Fehler auf.
Der Winkel der flaeche wird durch die Haeufigkeit bestimmt.

KLASSENBILDUNG
Man teilt in Klassen, wenn die Daten zu unterschiedlich sind.
Durch einen VERLUST an INFORMATION gleichzeitig ein GEWINN an UEBERSICHTLICHKEIT der Darstellung

SCHRITTE ZUR KLASSENBILDUNG
(1) Die Klassengrenzen sollen moeglichst einfache Zahlen sein, welche genau sind allerdings von den Eingangsdaten abhaengig
(2) Unterste und oberste Klasse werden so gewaehlt, dass der kleinstwert und der groesstwert

drinnenliegt.
(3) Wenn m die STICHPROBEN: meistens sqrt(m) aber immer <= 20 Klassen
(4) Die Klassenbreiten sollten gleich gross gewaehlt werden (mit ausnahme der 1. und letzten Klasse) (5) Klassen breite W: wenn n <= 400 –> W = (xmax - xmin) / sqrt(n) –> Ergebniss runden wenn n > 400 –> W = (xmax - xmin) / 20 –> Ergebniss runden

Die Urliste wird durch die Klasseneinteilung (welche dokumentiert und begruendet werden muss) in eine neue liste eingetragen
Diese Liste hat das Format: [1,2,6,2,8,3,5,1,5,7,8,3,7,...] und kann somit wieder interpretiert werden.

KENNZAHLEN (Kenngeoessen): unter Verlust von Information enthaltene Kenngroessen zum Vergleich von Haeufigkeiten
beeinhaltet: Mittelwert, Standardabweichung, Varianz, Spannweite, Minimalwert, Maximalwert, Median, Quantile, Quartile

MITTELWERT
HARMONISCHER MITTELWERT:
n dividiert durch die Summe der Kehrwerte aller Elemente (= arithmetische Mittel der Kehrwerte)
ARITHMETISCHER MITTELWERT:
Summe aller Elemente / n
GEOMETRISCHER MITTELWERT:
Die n-te Wurzel aus dem Produkt aller Elemente

MEDIAN
Genaue Haelfte der Liste
Wenn n gerade: Mittelwert der beiden mittleren Elemente
Wenn n ungerade: Mittelstes Element

SPANNWEITE
Die Distanz zwischen dem kleinsten und dem Groesstem Wert

QUANTIL
Ein Teil einer Menge, ein Lagemass, ein Schwellenwert.
Spezielle Quantile sind der Median, die Quartile, die Quintile, die Dezile und die Perzentile.

QUARTIL (=Viertelwerte)

Ein Quartil ist ein konkreter Wert bei
0.25-Quantil = unteres Quartil
0.50-Quantil = Median
0.75-Quantil = oberes Quartil

VARIANZ
Bei der Berechnung der Varianz wird von jedem Wert aus der Urliste das Arithmetische Mittel abgezogen und die daraus folgende Summe wird quadriert und mit den entsprechenden anderen Werte addiert.
Das ganze wird dann nochmal durch n dividiert.

STANDARDABWEICHUNG
Die Standardabweichung wird berechnet, indem aus der Varianz die (quadtratische) Wurzel gezogen wird.

BOXPLOT
Ein Boxplot dient dazu, einen schnellen Ueberblick ueber die Daten zu gewinnen. Alle oben genannten Punkte werden Dargestellt.
Von links nach rechts ist die Bedeutung der Striche folgende:
1 ... Kleinster Wert
2 ... 1. Quartil
3 ... Median
4 ... 3. Quartil
5 ... Groesster Wert
Andere Groessen, so wie zb. die Spannweite koennen daher nun auch aus dieser Grafik abgelesen werden.

2 Einfache Darstellung

```
(%i25) kill(values);
```

(%o25) *done*

2.1 Angabe

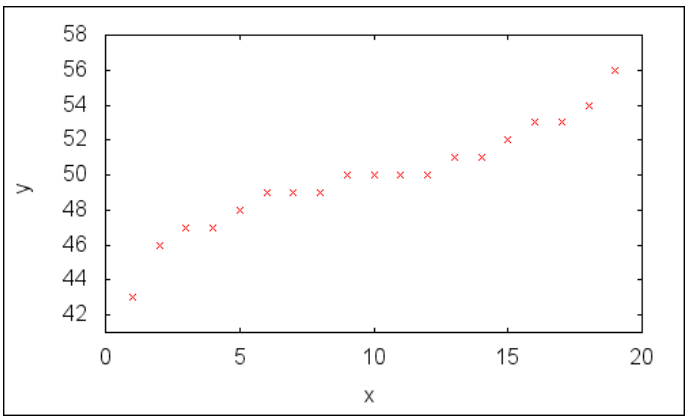
```
(%i26) urliste:[53,47,52,49,50,50,47,43,48,50,51,53,54,56,46,49,51,50,49];
```

(%o26) $[53, 47, 52, 49, 50, 50, 47, 43, 48, 50, 51, 53, 54, 56, 46, 49, 51, 50, 49]$

```
(%i27) angabe:sort(urliste)$
```

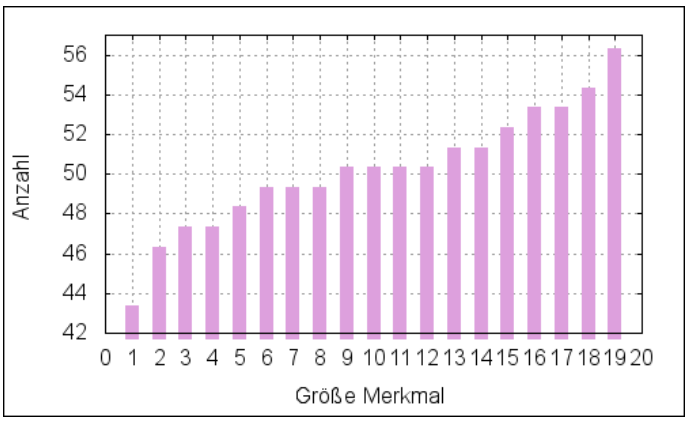
Einfache Darstellungen der Angabe

```
(%i28) paint_TYPE1_points(sort(angabe));
```



(%t28)

```
(%i29) paint_TYPE1_stricherl(angabe);
```



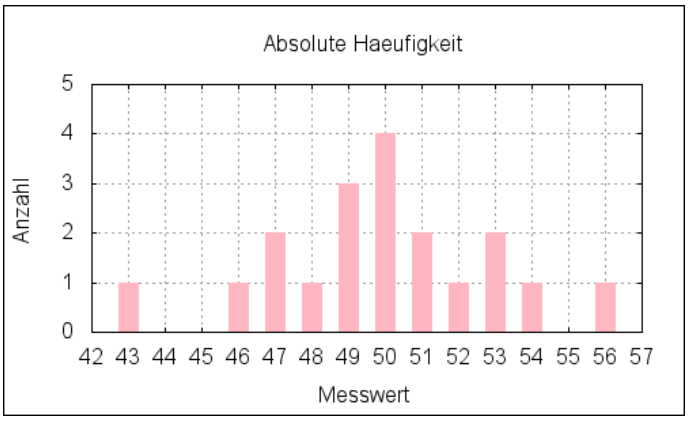
(%t29)

2.2 Absolute Haeufigkeit

Zeichnen der absoluten Haeufigkeit fuer Kondensatoren

```
(%i30) draw_ABSOLUTE_HFG_normal(angabe,1);
```

(%t30)

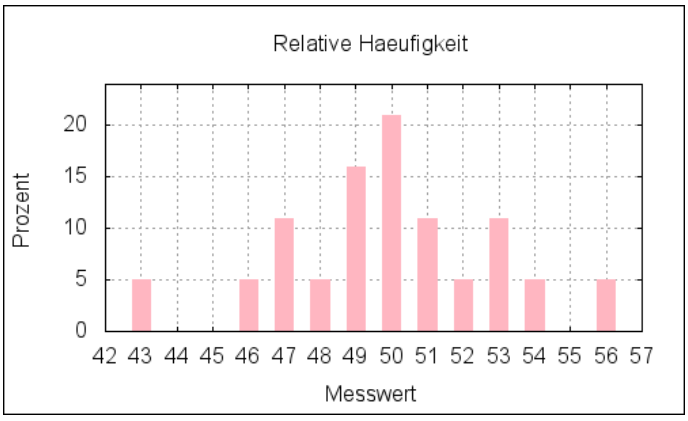


2.3 Relative Haeufigkeit

Zeichnen der relativen Haeufigkeit fuer Kondensatoren

```
(%i31) draw_RELATIVE_HFG_normal(angabe,5);
```

(%t31)

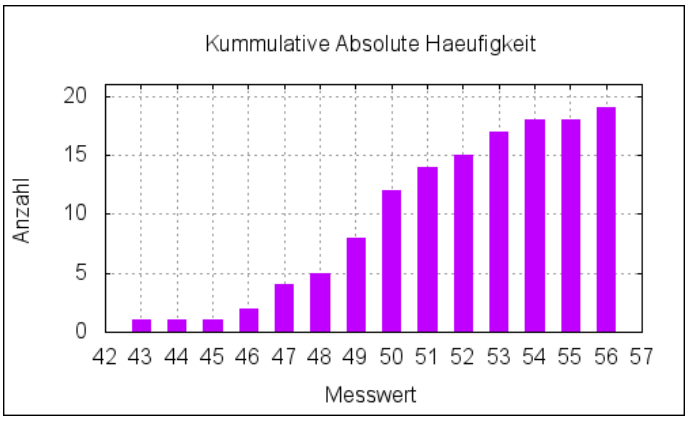


2.4 Kommulative absolute Haeufigkeit

Zeichnen der Kommulativen absoluten Haeufigkeit fuer Kondensatoren

```
(%i32) draw_KUM_ABS_HFG_normal(angabe,5);
```

(%t32)

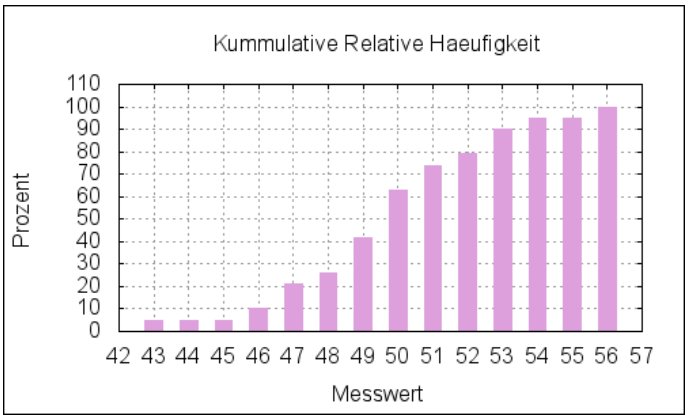


2.5 Kommulative relative Haeufigkeit

Zeichnen der Kommulativen relativen Haeufigkeit fuer Kondensatoren

```
(%i33) draw_KUM_REL_HFG_normal(angabe,10);
```

(%t33)



3 Paretto und Kreisdiagramm

```
(%i34) kill(values);
```

3.1 Angabe

Bei der Produktion von Blechteilen sind bei der Qualitätskontrolle folgende Fehler aufgetreten:
Fehler Fehlerart Häufigkeit
1 Delle 8
2 Kratzer 26
3 Korrosion 4
4 Lackfehler 38

5 Verbogen 4
6 Gerissen 2
7 Maßfehler 2
8 Bohrfehler 3
9 Sonstige 3

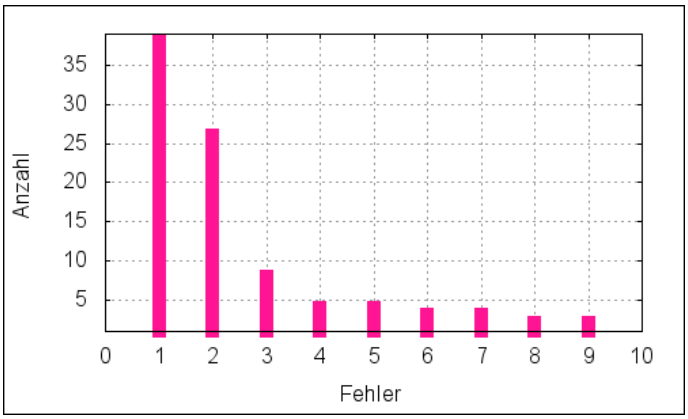
```
(%i35) urliste:[[3,Bohrfehler],[3,Sonstige],[8,Delle],[26,Kratzer],[4,Korrosion],[38,Lac

(%i36) angabe:urliste$
```

3.2 Pareto Diagram

```
(%i37) draw_pareto(angabe);
```

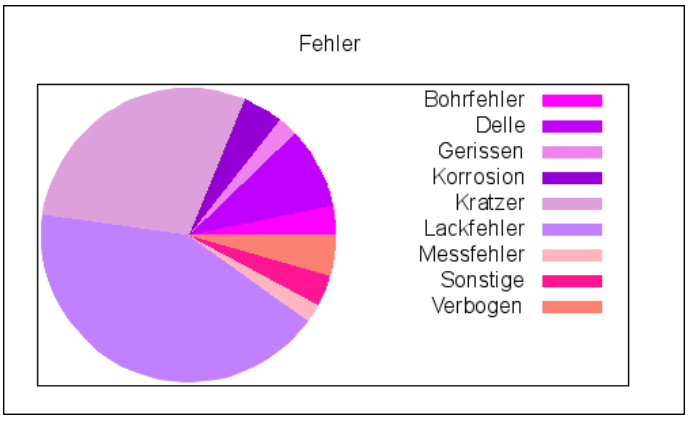
(%t37)



3.3 Kreisdiagramm

```
(%i38) draw_piechart(angabe);
```

(%t38)



4 Klassenbildung

```
(%i39) kill(values);
```

4.1 Angabe

Eine Messung des Füllgewichtes von 80 automatisch abgefüllten Marmeladegläsern ergab folgende Werte (in g):
93.0 88.2 92.1 91.0 86.9 93.8 91.0 92.0 87.4 92.1
92.7 93.3 88.8 91.5 90.4 88.1 89.7 88.9 92.7 91.4
89.6 89.4 87.3 86.5 90.8 90.3 86.9 91.2 89.3 90.4
94.2 90.4 90.8 91.9 90.8 92.4 88.3 92.0 87.3 93.8
90.6 88.0 94.0 90.9 88.0 93.1 91.7 89.7 92.7 89.5
90.7 89.3 89.3 86.5 88.9 87.5 88.2 88.7 90.2 86.3
93.9 86.4 90.6 87.9 85.0 89.1 91.8 92.3 87.9 95.4

```
(%i40) messwerte:[93.9 , 86.4 , 90.6 , 87.9 , 85.0 , 89.1 , 91.8 , 92.3 , 87.9 , 95.4,90
```

4.2 Theorie

Es herrscht bei dieser Angabe folgendes Problem: 80 Balken und alle sind ca geich gross aber doch nicht genau gleich (unterschied nach dem Komma).
Daher: teilen in Klassen (e.g. unter 85, zwische 85 und 90, zwischen 90 und 95 ... ueber 110)

Bei der Klassenbildung gilt immer dass:
durch einen VERLUST an INFORMATION gleichzeitig ein GEWINN an UEBERSICHTLICHKEIT der Darstellung

SCHRITTE ZUR KLASSENBILDUNG
(1) Die Klassengrenzen sollen moeglichst einfache Zahlen sein, welche genau sind allerdings von den Eingangsdaten abhaengig
(2) Unterste und oberste Klasse werden so gewaehlt, dass der kleinstwert und der groesstwert drinnenliegt.
(3) Wenn m die STICHPROBEN: meistens sqrt(m) aber immer <= 20 Klassen
(4) Die Klassenbreiten sollten gleich gross gewaehlt werden (mit ausnahme der 1. und letzten Klasse)
(5) Klassen breite W: wenn n <= 400 –> W = (xmax - xmin) / sqrt(n) –> Ergebniss runden
wenn n > 400 –> W = (xmax - xmin) / 20 –> Ergebniss runden

Die Urliste wird durch die Klasseneinteilung (welche dokumentiert und begruendet werden muss) in eine neue liste eingetragen
Diese Liste hat das Format: [1,2,6,2,8,3,5,1,5,7,8,3,7,...] und kann somit wieder interpretiert werden.

4.3 Klassenbildung

(2)
Kleinstwert(xmin) und Groesstwert (xmax)

```
(%i41) kleinstwert: lmin(messwerte);
       groesstwert: lmax(messwerte);
```

(%o41) 85.0

(%o42) 95.40000000000001

(3)
Anzahl der Messwerte

```
(%i43) anzahl_messwerte:length(messwerte);
```

(3)
Anzahl der Klassen

```
(%i44) anzahl_klassen:(sqrt(anzahl_messwerte)),numer;
```

(%o44) 8.366600265340756

```
(%i45) anzahl_klassen:round(anzahl_klassen);
```

(%o45) 8

(4) Klassenbreite

```
(%i46) W = ((groesstwert - kleinstwert) / anzahl_klassen),numer;
```

(%o46) $W = 1.300000000000001$

KLASSENBILDUNG
Das Ergebniss aus (3) und (4) muss nun interpretiert werden.
Da nur ganzzahlige Klassen gebildet werden koennen, sollten entweder 8 oder 9 Klassen gebildet werden.
Die Klassenbreite soll bei etwa 1.3 liegen. Da dies eine nicht sehr schoene Zahl ist, kann man entwender 1.25 oder 1.5 waehlen.
Wir waehlen nun folgende werte:
1.25 und 8

```
(%i47) steps:1.25;
       anzahl_klassen:8;
```

(%o47) 1.25

(%o48) 8

Aufruf des Blockes
Return value anzusprechen: end_wert (klasseneingeteilt) , klassen (die eingeteilt wurden)

```
(%i49) klassenbildung(messwerte,steps,anzahl_klassen);
```

(%o49) *done*

'Probe':

```
(%i50) length(end_list)=length(messwerte);
```

(%o50) 70 = 70

4.4 Grafische Darstellung der Klassen

Sollten die Werte (Labels) sich ueberschneiden, weil zu viele klassen geawehlt sind, muss man jediglich: - den letzten sowie den vorletzten Wert variieren, bis die Grafik schoen ist.
Die Default values koennen mit 0 angesprochen werden, oder einfach eingegeben werden ([500,300])

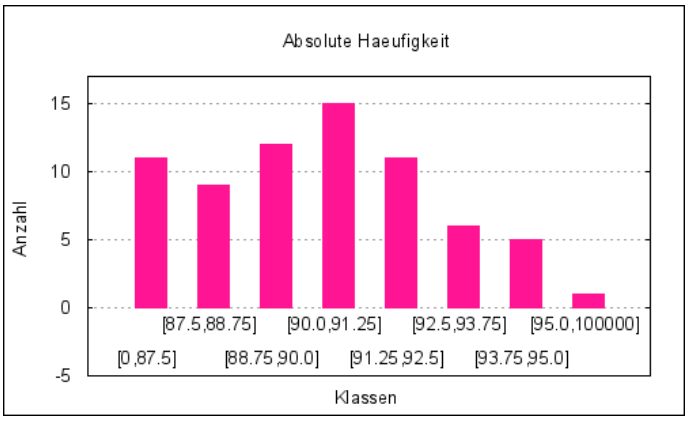
Die schon oben angefuehrte Erklaerung der draw Bloecke:
Jeder draw_TYPE_klassen Block, macht die Klasseneinteilung sowie das Zeichnen mit den Labels automatisch.

draw_TYPE_klassen(messwerte,steps,anzahl_klasse,ytics,minus_delta,xdimension,ydimension)

messwerte ... Eine Liste der messwerte im Format [Messwert1, Messwert2, ...]
steps ... Die Klassenbreite W
anzahl_klassen ... Anzahl der Klassen
ytics ... Abstand der Werte auf der y-Achse (Default:5)
minus_delta ... POSITIVER Wert, range fuer das anzeigen der Labels (sollte meist zwischen ca 5 und 20 liegen) (Default:10)
xdimension ... Groesse der Zeichnung nach x (Default:500)
ydimension ... Groesse der Zeichnung nach y (Default:300)

Zeichnen der absoluten Haeufigkeit

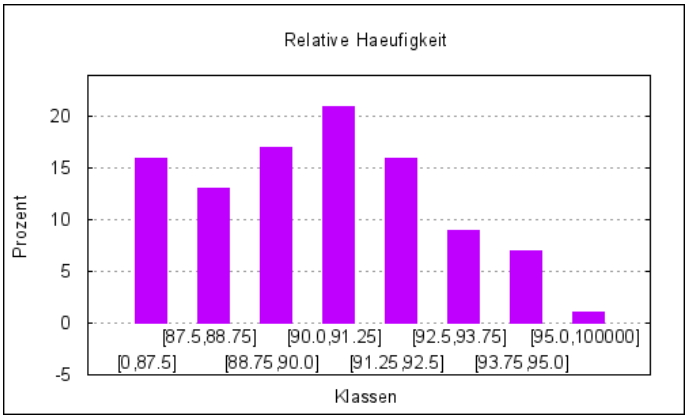
```
(%i51) draw_ABSOLUTE_HFG_klassen(messwerte,steps,anzahl_klassen,0,0,0,0);
```



(%t51)

Zeichnen der relativen Haeufigkeit

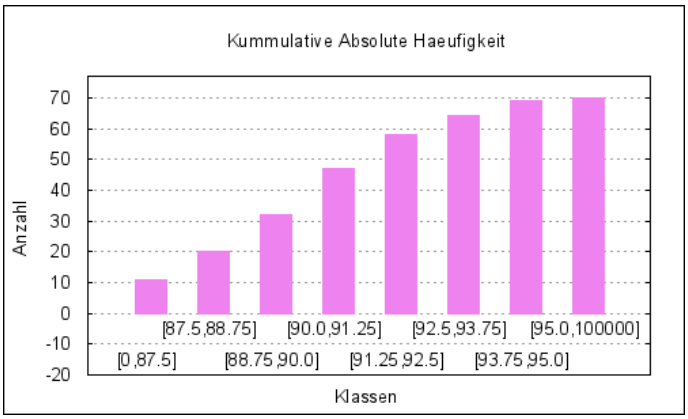
```
(%i52) draw_RELATIVE_HFG_klassen(messwerte,steps,anzahl_klassen,0,0,0,0);
```



(%t52)

Zeichnen der Kommulativen absoluten Haeufigkeit

```
(%i53) draw_KUM_ABS_HFG_klassen(messwerte,steps,anzahl_klassen,10,20,0,0);
```



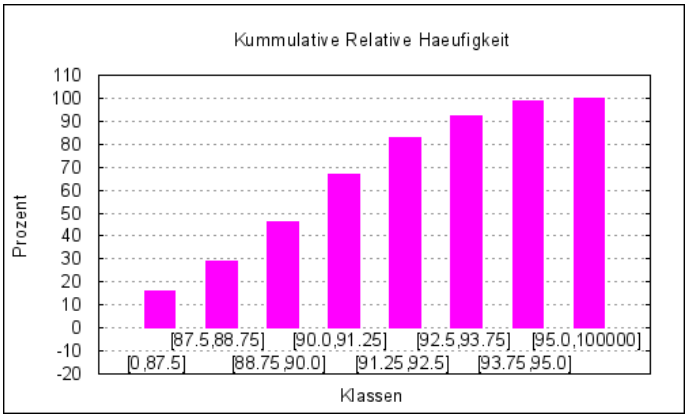
(%t53)

4.5 Kommulative relative Haeufigkeit

Zeichnen der Kommulativen relativen Haeufigkeit

```
(%i54) draw_KUM_REL_HFG_klassen(messwerte,steps,anzahl_klassen,10,20,0,0);
```

(%t54)



5 Kenngroessen

```
(%i55) kill (allbut (rand_col));
       load(descriptive)$
       load(draw)$
```

5.1 Angabe

Aus einer Lieferung von elektrischen Widerständen gleicher Art werden einige Stück herausgegriffen und gemessen.
Dabei werden folgende Werte (in Ohm) gefunden:

```
(%i3)  urliste:[98.7,  99.3,  95.8,  101.6,  100.4 , 98.0 , 96.3];
```

(%o3) $[98.7, 99.3, 95.8, 101.6, 100.4, 98.0, 96.3]$

```
(%i4)  angabe:urliste$
```

5.2 Minima und Maxima

minimalwert

```
(%i5)  smin(angabe);
```

(%o5) 95.8

maximalwert

```
(%i6)  smax(angabe);
```

(%o6) 101.6

5.3 Mittelwert

Arithmetischer Mittelwert

```
(%i7)  mean(angabe);
```

(%o7) 98.58571428571428

Geometrischer Mittelwert

```
(%i8)  geometric_mean (angabe);
```

(%o8) 98.56670574148326

Harmonischer Mittelwert

```
(%i9)  harmonic_mean(angabe);
```

(%o9) 98.54769453535242

5.4 Median

```
(%i10) median(angabe);
```

(%o10) 98.7

Ein Median und ein Mittelwert entsprechen sich dann, wenn man keine Werte hat, bei welchen die Randwerte extrem sind.

5.5 Standardabweichung

```
(%i11) std(angabe);
```

(%o11) 1.935701106966209

5.6 Varianz

Varianz dividiert durch n

```
(%i12) var(angabe);
```

(%o12) 3.746938775510206

Varianz dividiert durch n-1 (= Fuer groesse Mengen an daten)

```
(%i13) var1(angabe);
```

(%o13) 4.371428571428574

5.7 Spannweite

```
(%i14) range(angabe);
```

(%o14) 5.799999999999997

5.8 Quartile

unteres Quartil

```
(%i15) quantile (angabe, 0.25);
```

(%o15) 97.15000000000001

oberes Quartil

```
(%i16) quantile (angabe, 0.75);
```

(%o16) 99.84999999999999

5.9 Grafische Darstellung mittels Boxplot

```
(%i17) wxboxplot(angabe,box_orientation=horizontal,box_width=0.25,color=rand_col(1),line
```

(%t17)