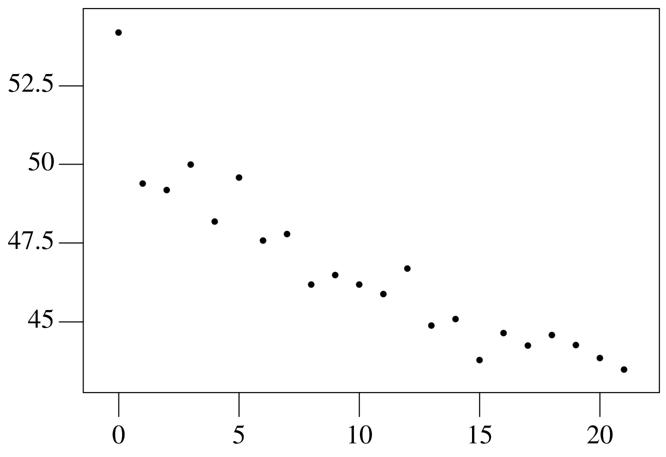
This is a collection of example graphs that are mostly taken from Bentley and Kernighan's paper on *grap* that is referenced from the manual page. Please check that for an explanation of the graphs' meaning. I use them primarily as a regression test suite.

Each graph is accompanied by a brief description and the *grap* code that produced it.

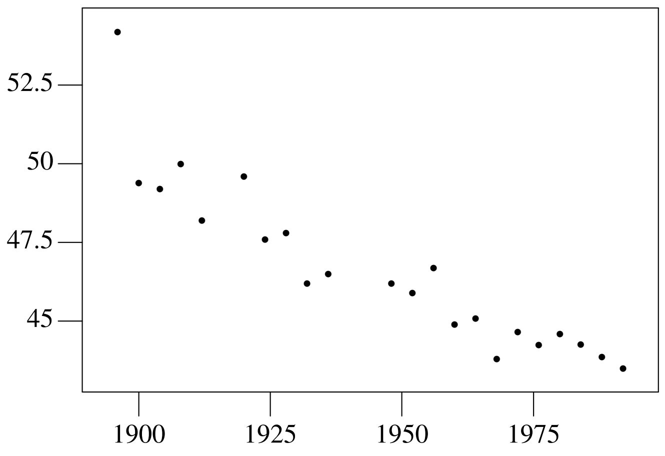
A simple copy. The data is the winning time for the mens' 400m run taken from the Pittsburgh Press's *1994 World Almanac and Book of Facts.*

```
.G1
copy "400mtimes.d"
.G2
```



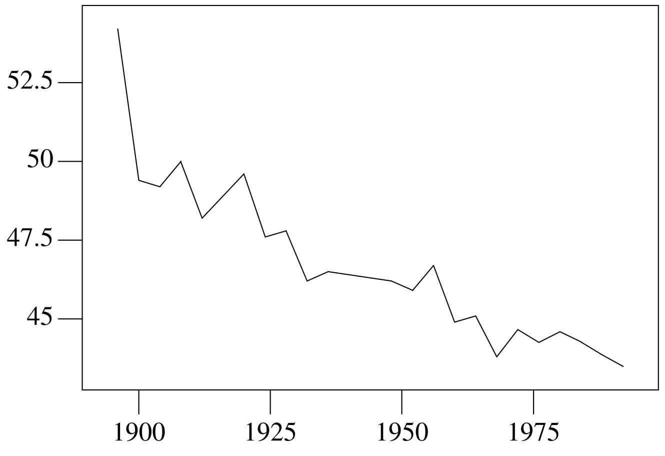
A simple copy with both *x* and *y* coordinates. The data is from the same source, but includes the year.

```
.G1
copy "400mpairs.d"
.G2
```



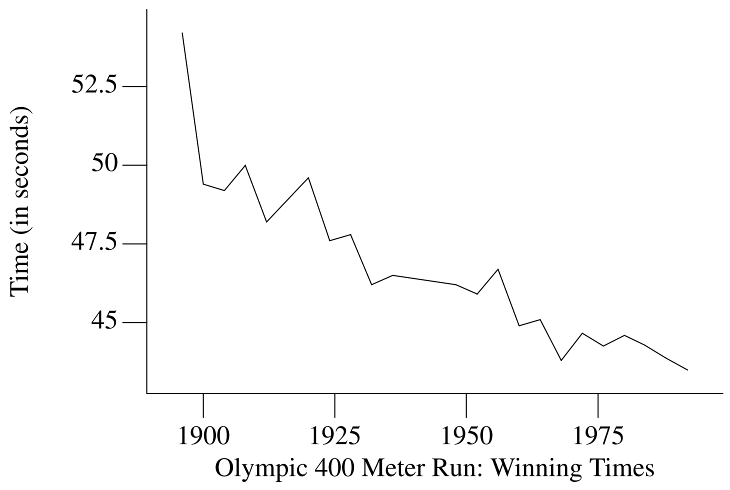
Simple connection of points. Same data.

```
.G1
draw solid
copy "400mpairs.d"
.G2
```



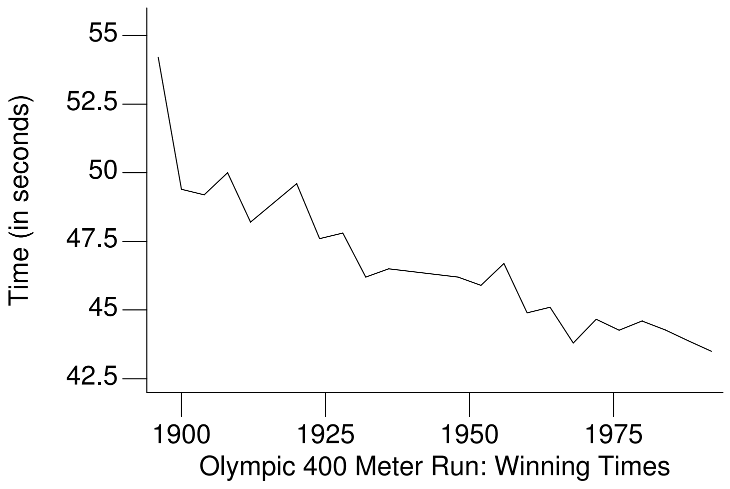
Spruced up a bit. We add axis labels and remove some frame boundaries. Note that my version of *grap* positions left and right axis labels parallel to the axis. That feature can be disabled by specifying `unaligned` as a justification for the label. The same data is used.

```
.G1
frame invis ht 2 wid 3 left solid bot solid
label left "Time (in seconds)" left .25
label bot "Olympic 400 Meter Run: Winning Times"
draw solid
copy "400mpairs.d"
.G2
```



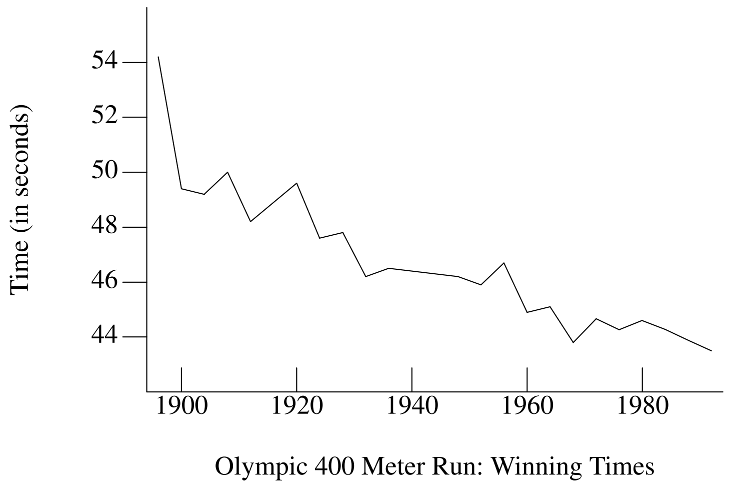
User specified axes, and a font change in the graph. You must restore the font after the terminating `.G2`. Commands to *groff* or *pic* are issued at the same place relative to other *grap* commands in the file (as of version 1.50). The frame is drawn before the first plotted element (invisible lines count) or when the frame statement is executed.

```
.G1
.ft H
frame invis ht 2 wid 3 left solid bot solid
label left "Time (in seconds)" left .25
label bot "Olympic 400 Meter Run: Winning Times"
coord x 1894, 1994 y 42, 56
draw solid
copy "400mpairs.d"
.G2
```



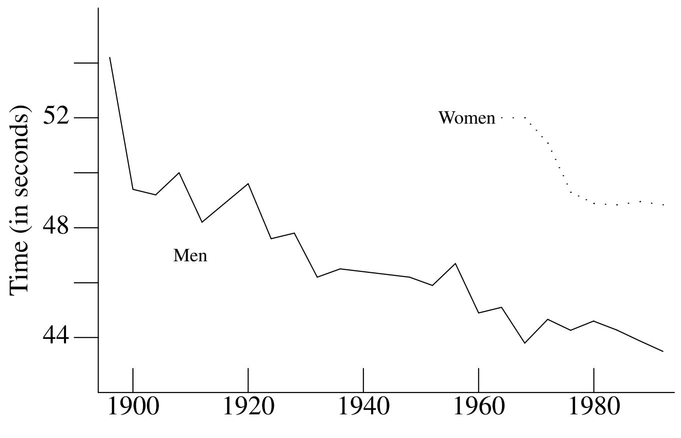
User specified tick placement, and a return to the Roman font. Same data.

```
.G1
frame invis ht 2 wid 3 left solid bot solid
label left "Time (in seconds)" left .25
label bot "Olympic 400 Meter Run: Winning Times"
coord x 1894, 1994 y 42, 56
ticks left out at 44 "44", 46, 48 "48", 50, 52 "52", 54
ticks bot in from 1900 to 1980 by 20
draw solid
copy "400mpairs.d"
.G2
```



This graph adds the womens' times (from the same source) and includes different plotting styles and labels. My version of this graph also uses the `until` keyword to include the data rather than external files. `until` appears before and after the macro definition.

```
.G1
frame invis ht 2 wid 3 left solid bot solid
label left "Time (in seconds)"
label bot "Olympic 400 Meter Run: Winning Times"
coord x 1894, 1994 y 42, 56
ticks left out at 44 , 46 "", 48, 50 "", 52, 54 ""
ticks bot in from 1900 to 1980 by 20
draw solid
copy until "END" thru {next at $1, $2;}
1896 54.2
1900 49.4
[...]
1992 43.50
END
new dotted
copy until "END"
1964 52.
1968 52.
[...]
1992 48.83
END
"Women"  size -3 at 1958,52
"Men" size -3 at 1910,47
.G2
```

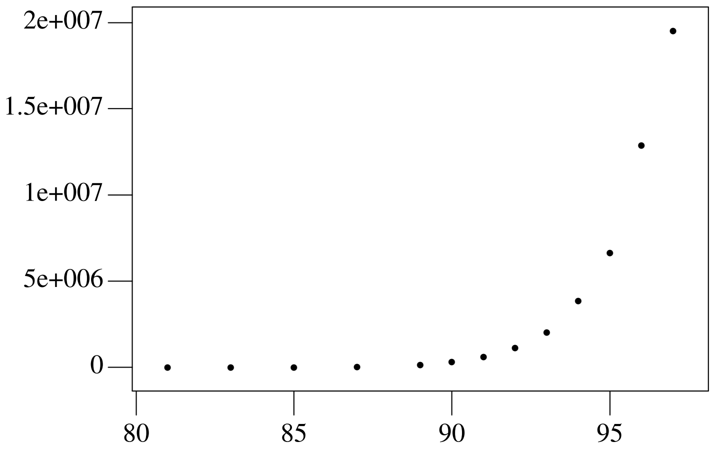


Olympic 400 Meter Run: Winning Times

Another simple copy. Bentley and Kernighan use phone installations, I use numbers of hosts on the Internet. The data is from `ftp://nic.merit.edu/nsfnet/statistics/history.hosts`, maintained by:

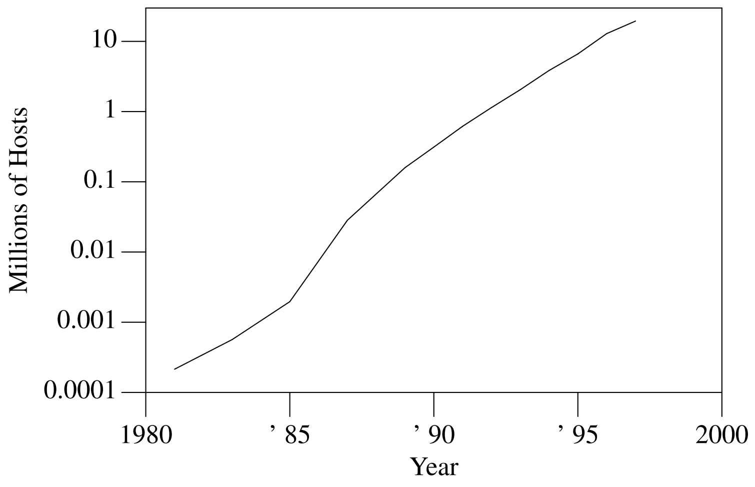
Merit Network
4251 Plymouth Road
Suite C
Ann Arbor, MI 48105-2785
734-764-9430

```
.G1
copy "internet.d"
.G2
```



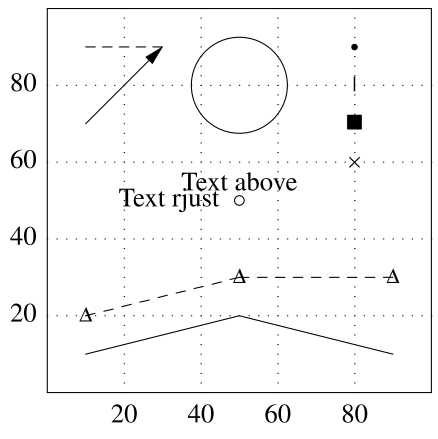
The same data plotted on a logarithmic *y* scale, and rescaled to megahosts, which is less humorous than megaphones. The placement of the bottom ticks shows the `from .. to .. by` construct. The filename is given after the macro for variety (and to test that feature).

```
.G1
coord x 80,100 y 1e-4, 30 log y
ticks bot at 80 "1980", 100 "2000"
ticks bot from 85 to 95 by 5 "' %g"
ticks left
label left "Millions of Hosts" left .25
label bot "Year"
draw solid
copy thru { next at $1, $2/1e6;} "internet.d"
.G2
```



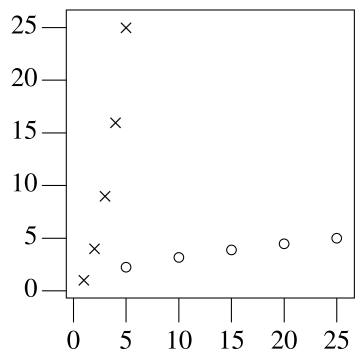
A demo of *grap*'s annotation abilities, mostly. The data is all in the graph specification.

```
.G1
frame ht 2 wid 2
coord x 0,100 y 0,100
grid bot dotted from 20 to 80 by 20
grid left dotted from 20 to 80 by 20

"Text above" above at 50,50
"Text rjust   " rjust at 50,50
bullet at 80,90
vtick at 80,80
box at 80,70
times at 80,60

circle at 50,50
circle at 50,80 radius .25
line dashed from 10,90 to 30,90
arrow from 10,70 to 30,90

draw A solid
draw B dashed delta
next A at 10,10
next B at 10,20
next A at 50,20
next A at 90,10
next B at 50,30
next B at 90,30
.G2
```



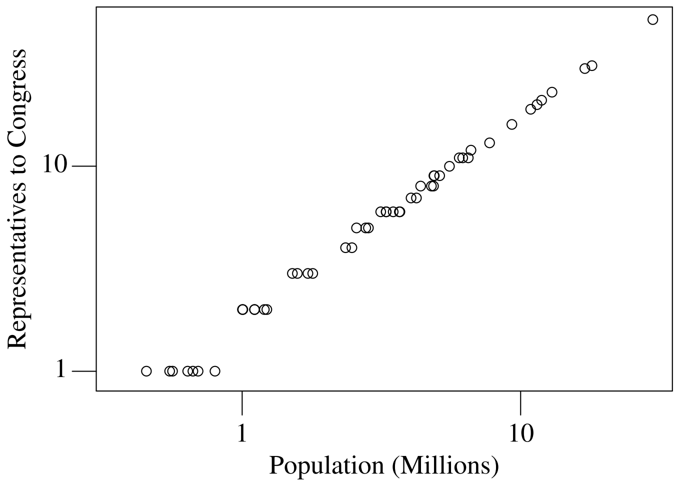
A simple macro demo. Again, no data.

```
.G1
frame ht 1.5 wid 1.5
define square {($1) * ($1)}
define root {($1)^.5 }
define P {
    times at i, square(i); i = i +1;
    circle at j, root(j); j= j+5;
}
i = 1; j = 5
P; P; P; P; P
.G2
```



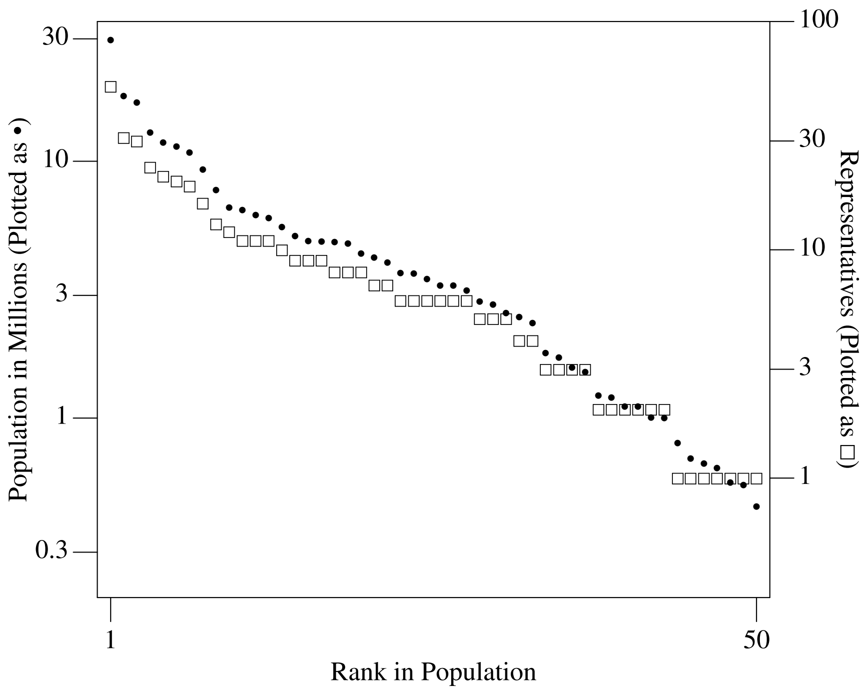
The number of Representatives to the U.S. Congress versus population of the states. My data is more recent than that of Bentley/Kernighan, so the graph is different from theirs. Data is from the U.S. Census Bureau at `http://www.census.gov/`, specificly `http://www.census.gov/population/www/censusdata/apportion-ment.html`.

```
.G1
label left "Representatives to Congress"
label bot "Population (Millions)"
coord x .3, 35 y .8, 60 log log
define PlotState { circle at $3/1e6, $2; }
copy "states.d" thru PlotState
.G2
```



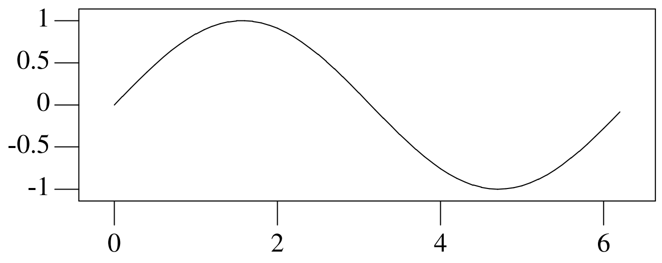
A 2-axis plot. We redefine square because the macro example graph changed it. I advise against changing the predefined macro definitions because macros persist across graphs. The same data is plotted.

```
.G1
define square {"\s-2\(sq\s0"}
frame ht 3 wid 3.5
label left "Population in Millions (Plotted as \(bu)"
label bot "Rank in Population"
label right "Representatives (Plotted as \(sq)"
coord pop x 0,51 y .2,35 log y
coord rep x 0,51 y .3,100 log y
ticks left out at pop .3,1,3,10,30
ticks bot out at pop 1,50
ticks right out at rep 1,3,10,30,100
thisrank = 50
copy "states.d" thru {
    bullet at pop thisrank,$3/1e6
    square at rep thisrank,$2
    thisrank = thisrank -1
}
.G2
```



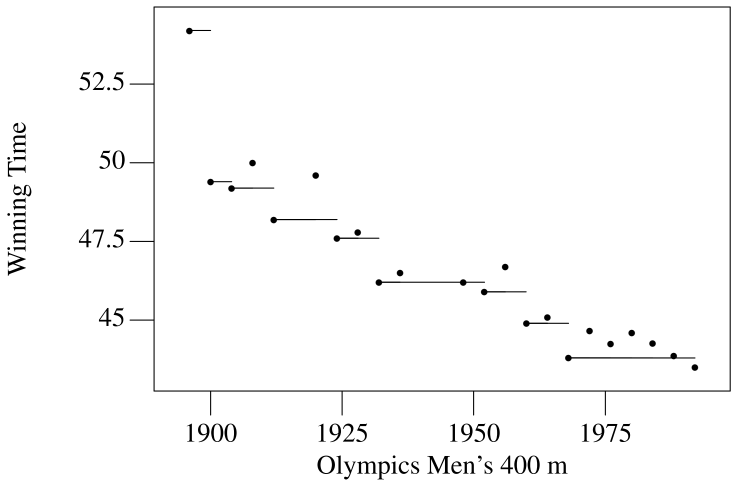
A sine wave plotted by a `for` loop with π calculated with the internal `atan2()` function. No data.

```
.G1
frame ht 1 wid 3
draw solid
pi = atan2(0,-1)
for i from 0 to 2* pi by .1 do { next at i, sin(i); }
.G2
```



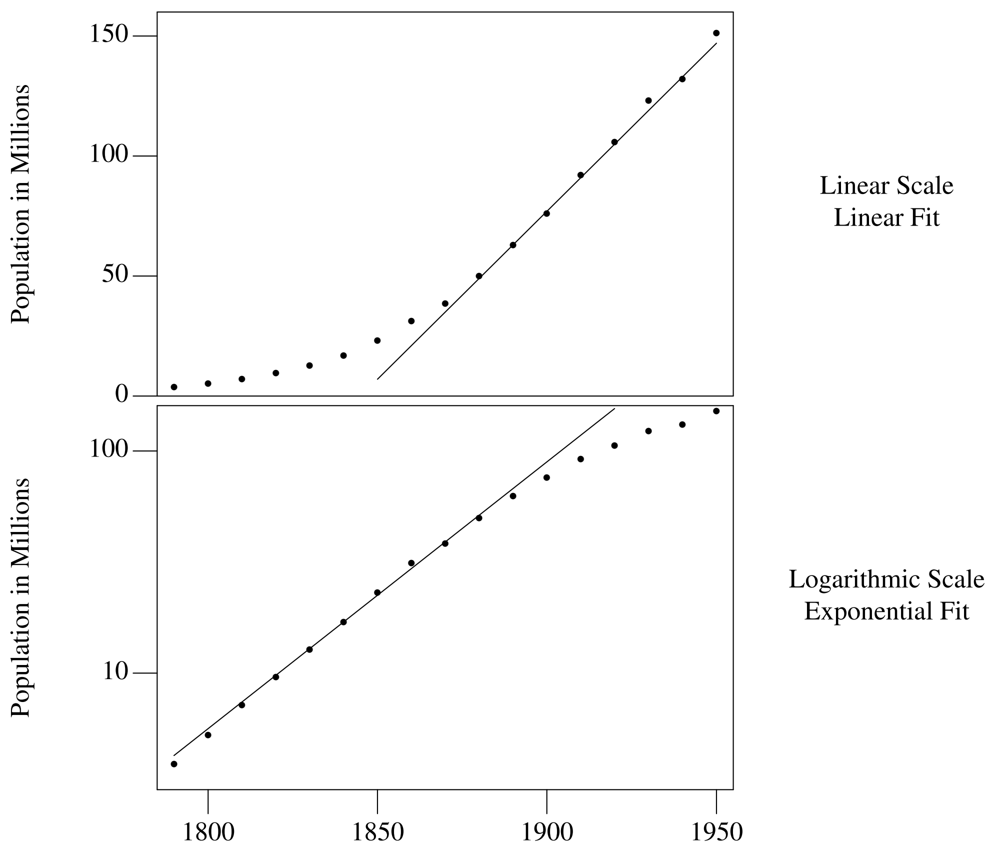
Bentley and Kernighan do this graph with Kentucky Derby winning times. I don't have them, so I used the 400m times again. My program is slightly different because the 400m run times have gaps.

```
.G1
label left "Winning Time" left .3
label bot "Olympics Men's 400 m"
bestsofar = 1000
anchor = 0
copy "400mpairs.d" thru {
    bullet at $1,$2
    if ( anchor != 0 ) then {
        line from anchor, bestsofar to $1,bestsofar
    }
    bestsofar = min(bestsofar,$2)
    if ( bestsofar == $2 ) then {
        anchor = $1
    }
}
.G2
```



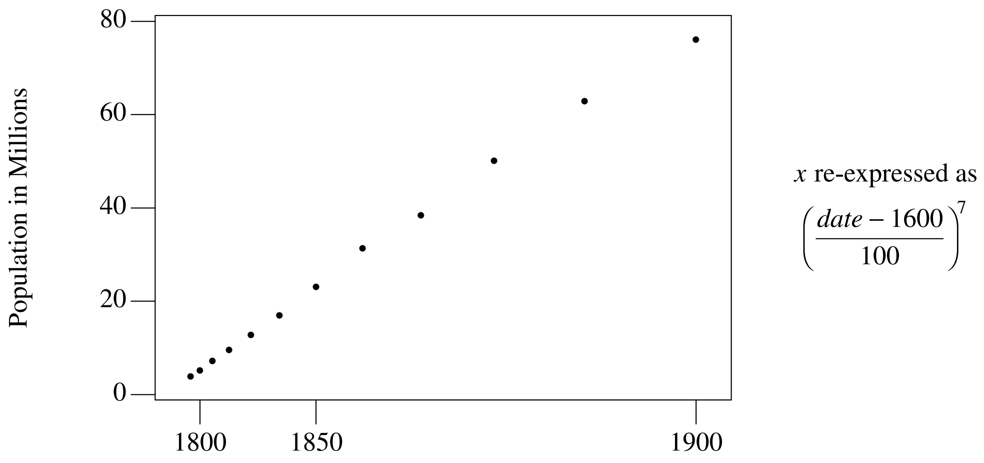
Bentley and Kernighan discuss the regression and modeling that these graphs reflect. The data is the U.S. population from the U.S. Census bureau. This shows off the ability to place two plots relative to each other using the `graph` statement.

```
.G1
graph Linear
coord x 1785, 1955 y 0, 160
label left "Population in Millions" left .3
label right "Linear Scale" unaligned "Linear Fit" right .4
ticks bot off
copy "usapop.d"
define fit { 35 + 1.4 * ($1-1870) }
line from 1850, fit(1850) to 1950,fit(1950)
graph Exponential with .Frame.n at Linear.Frame.s - (0, .05)
coord x 1785, 1955 y 3, 160 log y
label left "Population in Millions" left .3
label right "Logarithmic Scale" unaligned "Exponential Fit" right .4
copy "usapop.d"
define fit { exp(.75 + .012 * ($1-1800)) }
line from 1790, fit(1790) to 1920,fit(1920)
.G2
```



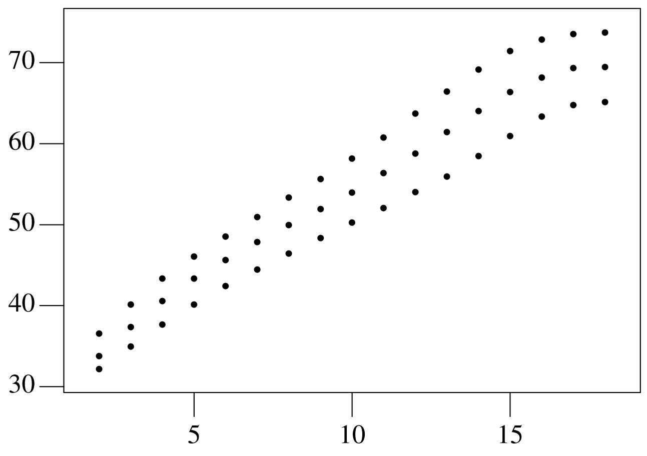
Another re-expression of the U.S. population data. Uses plenty of *grap* arithmetic and an *eqn* axis label (which is `unaligned`).

```
.G1
label left "Population in Millions" left .3
label right "$x$ re-expressed as" unaligned "" "$space 0 left
    ( { date -1600 } over 100 right ) sup 7$" right .4
define newx { exp(7*(log(($1-1600)/100))) }
ticks bot out at newx(1800) "1800", newx(1850) "1850", \
    newx(1900) "1900"
copy "usapop.d" thru {
    if $1 <= 1900 then { bullet at newx($1),$2 }
}
.G2
```



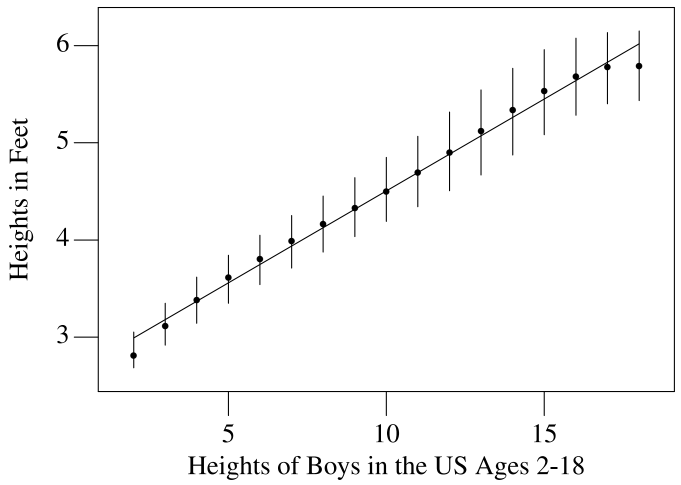
A simple copy of a multiple field data file. The data is the 5th, 50th, and 95th percentiles for the heights of boys in America at different ages. The data is reported from Thomas J. Glover's remarkable *Pocket Ref*, which reports data from the National Center for Health Statistics. *Pocket Ref* is published by Sequoia Publishing, Littleton, CO.

```
.G1
copy "boyhts.d"
.G2
```



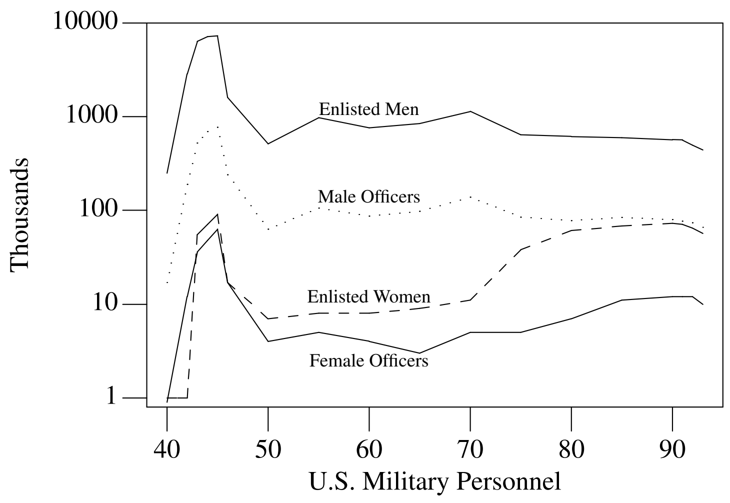
The same data with a linear regression, and the 90% confidence intervals drawn as lines. Note the cascading assignment statements (patch courtesy of Bruce Lilly). (Bentley and Kernighan's data is in centimeters, mine is in inches, so a different conversion to feet is used.)

```
.G1
label left "Heights in Feet"
label bot "Heights of Boys in the US Ages 2-18"
cmpft = 12
minx = 1e12; maxx = -1e12
n = sigx = sigx2 = sigy = sigxy = 0;
copy "boyhts.d" thru {
    line from $1, $2/cmpft to $1, $4/cmpft
    ty = $3 / cmpft
    bullet at $1, ty
    n = n+1
    sigx = sigx + $1; sigx2 = sigx2 + $1 * $1
    sigy = sigy + ty; sigxy = sigxy + $1*ty
    minx = min(minx,$1); maxx = max(maxx,$1);
}
slope = ( n*sigxy - sigx* sigy) / (n*sigx2 - sigx * sigx)
inter = ( sigy - slope * sigx) / n
line from minx, slope * minx+inter to maxx, slope * maxx + inter
.G2
```



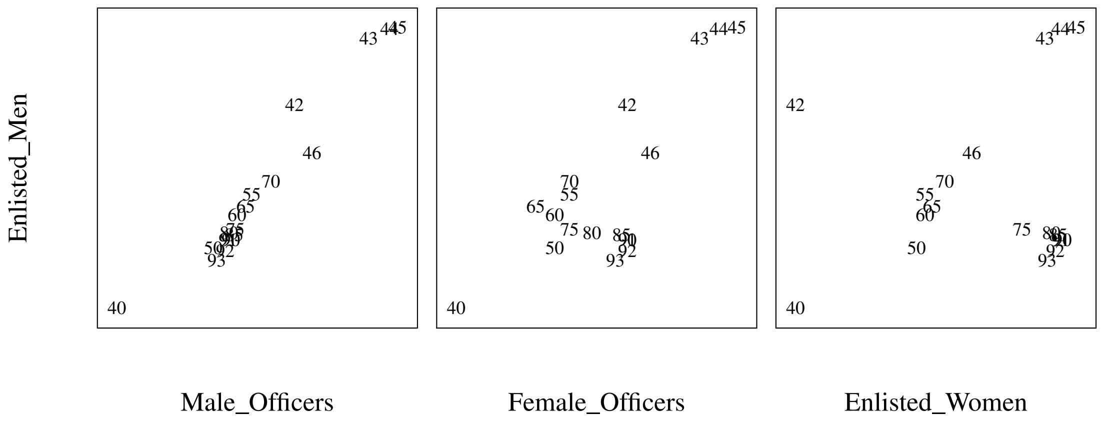
This is a 4 linestyle plot with a copy statement that adds labels. The scales are user-defined, and the *y* axis is logarithmic. The data is the number of male and female officers and enlisted personnel in the U.S. Armed forces from 1940-1993 from the Pittsburgh Press *World Almanac and Book of Facts.* This graph has more data than the equivalent from Bentley and Kernighan.

```
.G1
coord x 38, 95 y .8, 10000 log y
label bot "U.S. Military Personnel"
label left "Thousands" left .25
draw of solid
draw ef dashed
draw om dotted
draw em solid
copy "army.d" thru {
    next of at $1, $3
    next ef at $1, $5
    next om at $1, $2
    next em at $1, $4
}
copy until "XXX" thru { "$1 $2" size -3 at 60, $3; }
Enlisted Men 1200
Male Officers 140
Enlisted Women 12
Female Officers 2.5
XXX
.G2
```



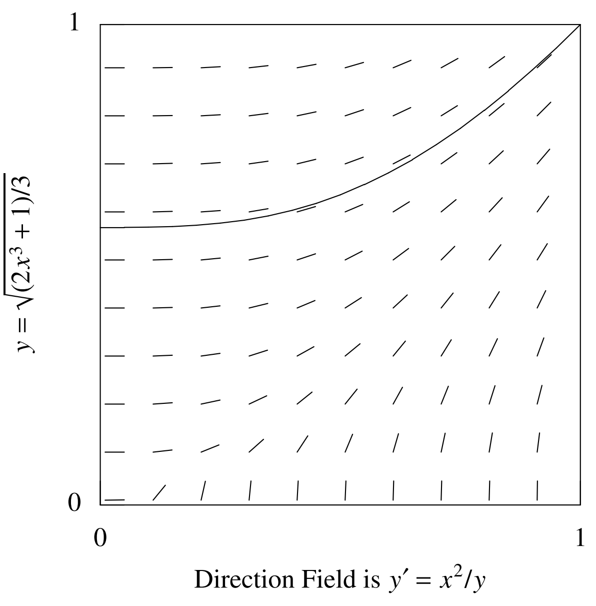
Obfuscation of data via scatter plots. Three aligned graphs are produced that plot the numbers of enlisted men as functions of male officers, female officers, and enlisted women. The plotting symbol is the year in question. Same data as above.

```
.G1
graph A
frame ht 1.6667 wid 1.6667
label bot "Male_Officers"
label left "Enlisted_Men"
coord log log
ticks off
copy "army.d" thru { "$1" at $2,$4; }
graph A with .Frame.w at A.Frame.e +(.1,0)
frame ht 1.6667 wid 1.6667
label bot "Female_Officers"
coord log log
ticks off
copy "army.d" thru { "$1" at $3,$4; }
graph A with .Frame.w at A.Frame.e +(.1,0)
frame ht 1.6667 wid 1.6667
label bot "Enlisted_Women"
coord log log
ticks off
copy "army.d" thru { "$1" at $5,$4; }
.G2
```



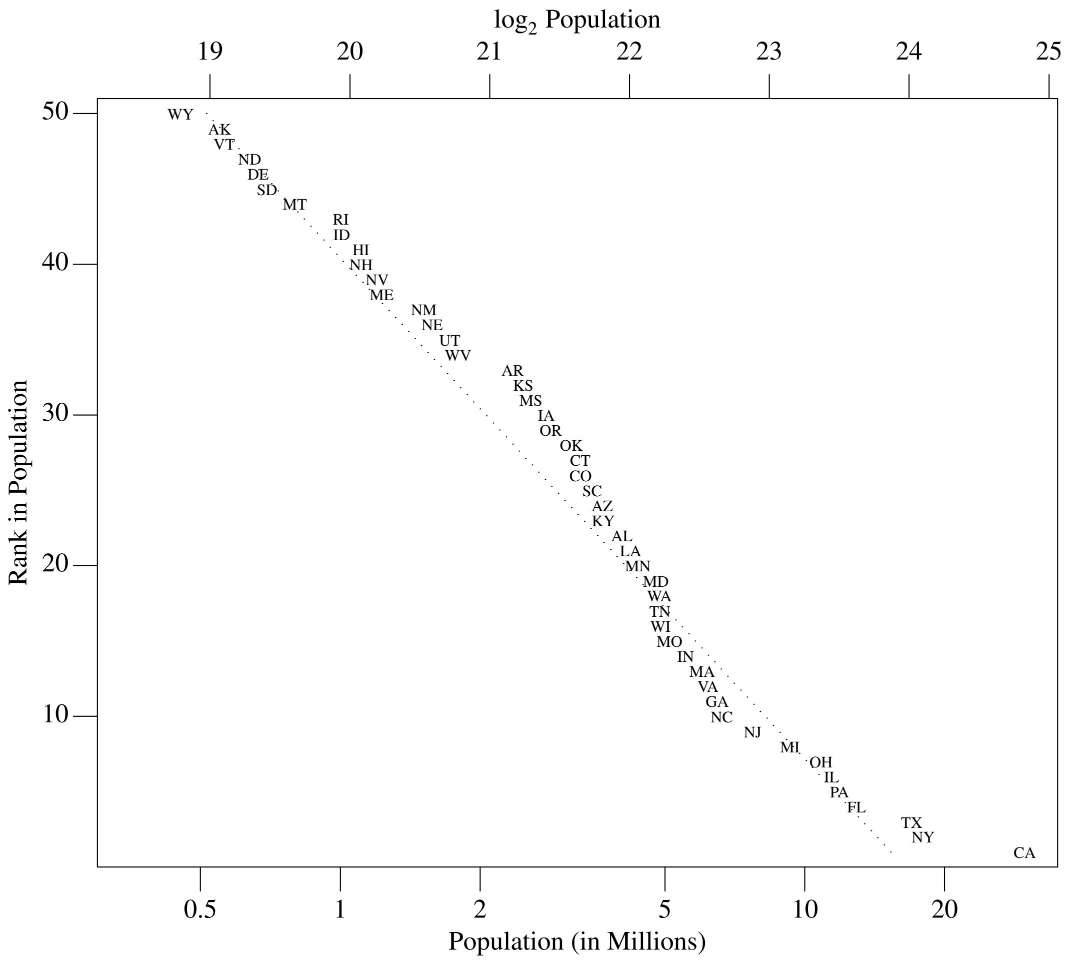
One of my favorites. The solution of a differential equation and the slope field it passes through. It shows off nested `for` loops (one using = as a synonym for from) and *eqn* labels. The data is in the graph description.

```
.G1
frame ht 2.5 wid 2.5
coord x 0,1 y 0,1
label bot "Direction Field is $y prime = x sup 2 / y$"
label left "$y = sqrt { ( 2 x sup 3 + 1 ) / 3 }$"
ticks left in 0 left .1 at 0,1
ticks bot in 0 down .1 at 0,1
len = .04
for tx from .01 to .91 by .1 do {
    for ty from .01 to .91 by .1 do {
        deriv = tx*tx/ty
        scale = len / sqrt(1 + deriv*deriv)
        line from tx,ty to tx+scale, ty+scale*deriv
    }
}
draw solid
for tx = 0 to 1 by .05 do {
    next at tx, sqrt((2*tx*tx*tx+1)/3)
}
.G2
```



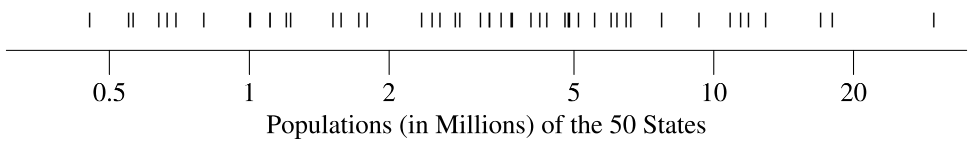
More population studies. State population rank vs. population, with the population on a log scale. A regression line is also plotted. I used the same line as Bentley and Kernighan, although my data is more recent. The top labels are generated by a series of macros, the frame size is enlarged, and the plotting symbol is the state abbreviation. This graph uses the same census data as above.

```
.G1
frame wid 5 ht 4
label left "Rank in Population"
label bot "Population (in Millions)"
label top "$log sub 2$ Population"
coord x .3, 35 y 0, 51 log x
define L { (2.0^$1)/1e6 "$1" }
ticks bot out at .5, 1, 2, 5, 10, 20
ticks left out from 10 to 50 by 10
ticks top out at L(19), L(20), L(21), L(22), L(23), L(24), L(25)
thisy = 50
copy "states.d" thru {
    "$1" size -4 at $3/1e6, thisy
    thisy = thisy-1
}
line dotted from 15.3,1 to .515, 50
.G2
```



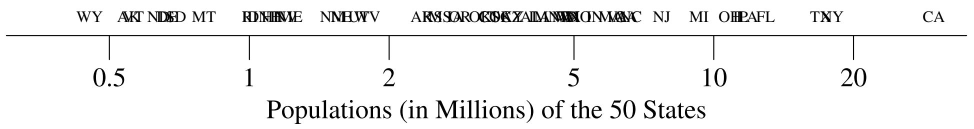
A nearly useless plot of the populations of different states. Same data.

```
.G1
frame invis ht .3 wid 5 bottom solid
label bot "Populations (in Millions) of the 50 States"
coord x .3, 35 y 0, 1 log x
ticks bot out at .5, 1, 2, 5, 10, 20
ticks left off
copy "states.d" thru { vtick at $3/1e6, .5; }
.G2
```



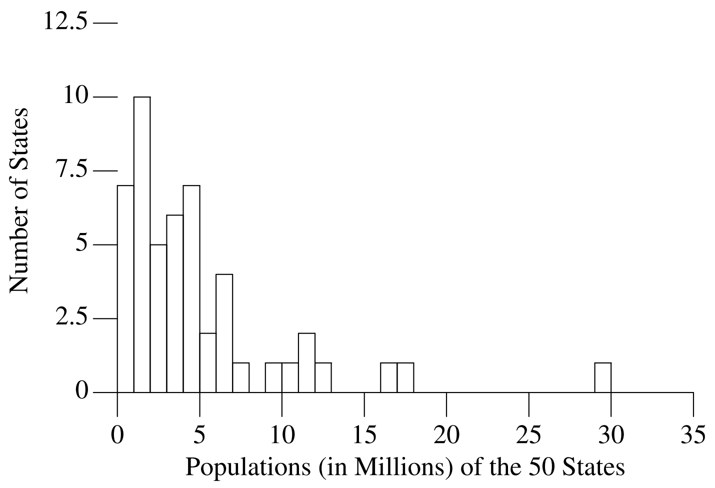
A slight improvement, as the states are spread out and plotted with their symbols. The `rand()` function is used to position them vertically, which shows off the function, but doesn't guarantee a legible graph. Same data.

```
.G1
frame invis ht 1 wid 5 bottom solid
label bot "Populations (in Millions) of the 50 States"
coord x .3, 35 y 0, 1000 log x
ticks bot out at .5, 1, 2, 5, 10, 20
ticks left off
copy "states.d" thru { "$1" size -4 at $3/1e6, 100+900*rand(); }
.G2
```



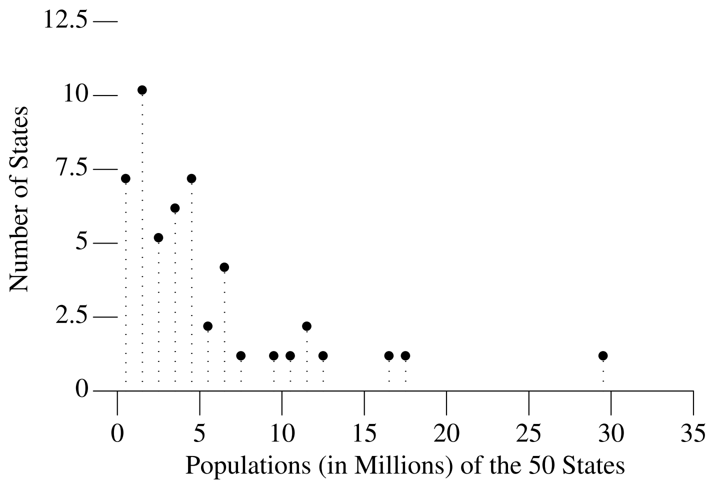
A histogram of the same data. The input file is a result of running the census data through the *awk* script that Bentley and Kernighan describe.

```
.G1
frame invis bot solid
label bot "Populations (in Millions) of the 50 States"
label left "Number of States" left .1
ticks bot out from 0 to 35 by 5
coord x 0, 35 y 0, 13
copy "states2.d" thru {
    line from $1,0 to $1,$2
    line from $1, $2 to $1+1, $2
    line from $1+1,$2 to $1+1,0
}
.G2
```



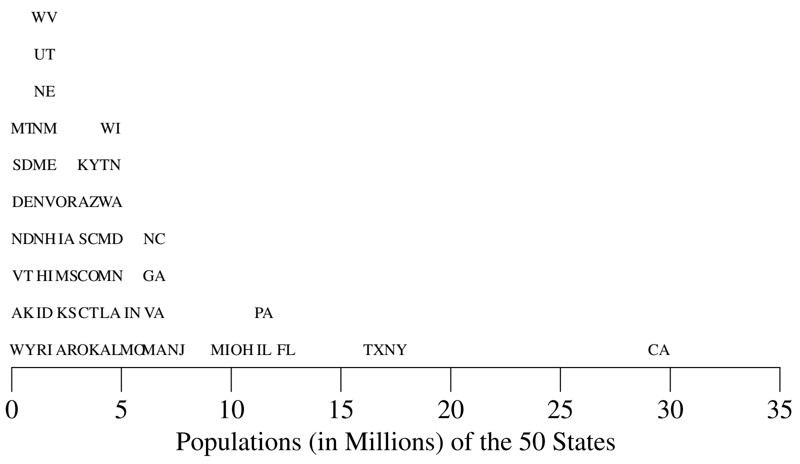
A “lolliplot ” histogram of the same data.

```
.G1
frame invis bot solid
label bot "Populations (in Millions) of the 50 States"
label left "Number of States" left .1
ticks bot out from 0 to 35 by 5
coord x 0, 35 y 0, 13
copy "states2.d" thru {
    line dotted from $1+.5,0 to $1+.5,$2
    "•" size +3 at $1+.5, $2
}
.G2
```



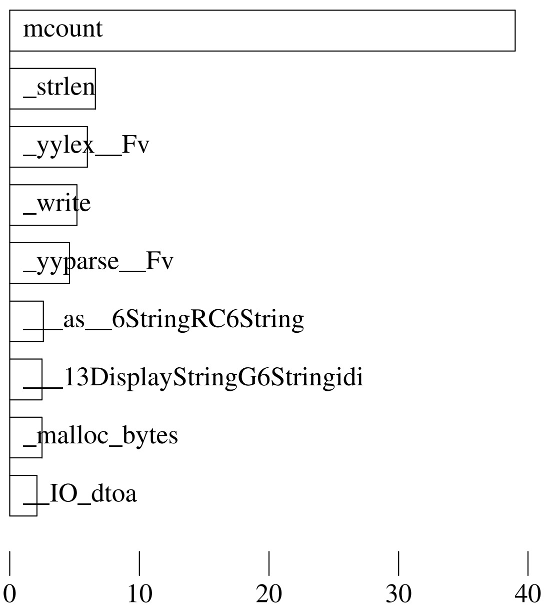
A histogram of state abbreviations. The data has been massaged by the *awk* program described by Bentley and Kernighan.

```
.G1
frame invis wid 4 ht 2.5 bot solid
label bot "Populations (in Millions) of the 50 States"
ticks bot out from 0 to 35 by 5
ticks left off
coord x 0, 35 y 0, 13
copy "states3.d" thru {"$1" size -4 at $2+.5, $3+.5; }
.G2
```



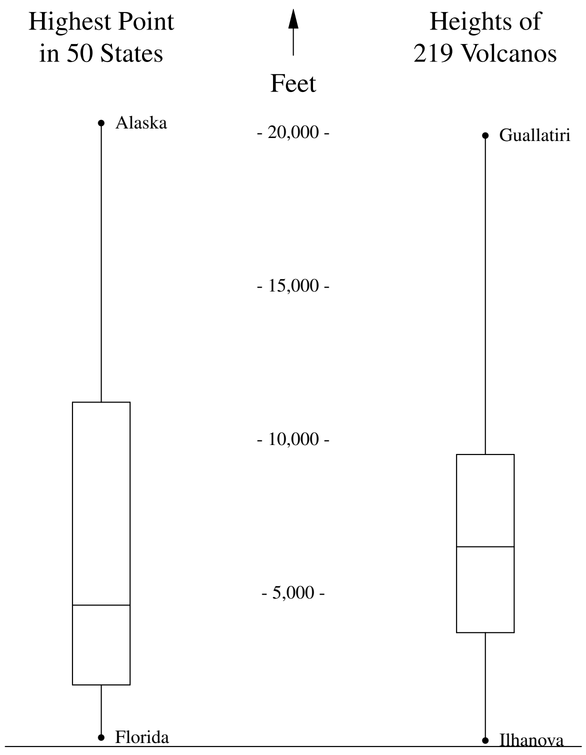
A bar graph of profiler output. The output is from running *grap* on this file.

```
.G1
ticks left off
cury = 0
barht = .7
copy "prof2.d" thru {
    line from 0,cury to $1, cury
    line from $1, cury to $1, cury-barht
    line from 0, cury-barht to $1, cury-barht
    "  $2" ljust at 0, cury-barht/2
    cury = cury-1
}
line from 0,0 to 0,cury+1-barht
bars = -cury
frame invis ht bars/3 wid 3
.G2
```



The creative graph of state heights and volcano heights from their grap paper. The data was included in the graph description, which (according to Bruce Lilly) is from John W. Tukey's classic 1977 text *Exploratory Data Analysis,* Chapter 10. Tukey cites "The World Almanac, 1966, page 269. Their source: National Geographic Society." The format of the graph is also similar to a graph on pg. 40 (exhibit 5, chapter 2). All those attributions are from Bruce Lilly. I don't have the relevant material to verify it, but have no reason to doubt the accuracy of the information.

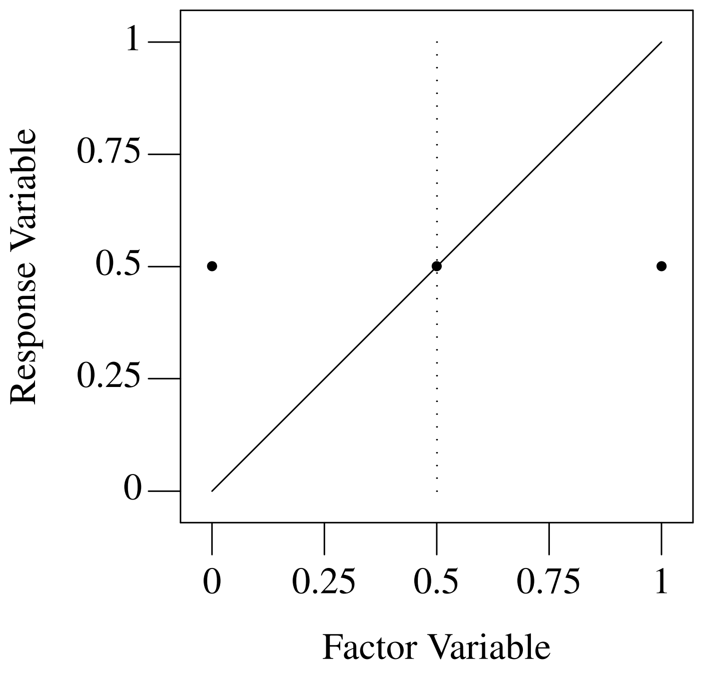
```
.G1
frame invis ht 4 wid 3 bot solid
ticks off
coord x .5, 3.5 y 0,25
define Ht { "- $1,000 -" size -3 at 2, $1 }
Ht(5); Ht(10); Ht(15); Ht(20);
"Highest Point" "in 50 States" at 1,23
"Heights of" "219 Volcanos" at 3,23
"Feet" at 2,21.5; arrow from 2,22.5 to 2,24
define box {
     xc = $1; xl = xc - boxwidth/2; xh = xc+boxwidth/2
     y1 = $2; y2 = $3; y3 = $4; y4= $5; y5 = $6
     bullet at xc,y1
     "  $7" size -3 ljust at xc, y1
     line from (xc,y1) to (xc,y2)
     line from (xl,y2) to (xh,y2)
     line from (xl,y3) to (xh,y3)
     line from (xl,y4) to (xh,y4)
     line from (xl,y2) to (xl,y4)
     line from (xh,y2) to (xh,y4)
     line from (xc,y4) to (xc,y5)
     bullet at xc,y5
     "  $8" ljust size -3  at (xc,y5)
}
boxwidth = .3
box(1, .3, 2.0, 4.6, 11.2,20.3, Florida, Alaska)
box(3,.2, 3.7, 6.5, 9.5, 19.9, Ilhanova, Guallatiri)
.G2
```



Highest Point in 50 States
Feet
Heights of 219 Volcanos
Alaska
- 20,000 -
Guallatiri
- 15,000 -
- 10,000 -
- 5,000 -
Florida
Ilhanova

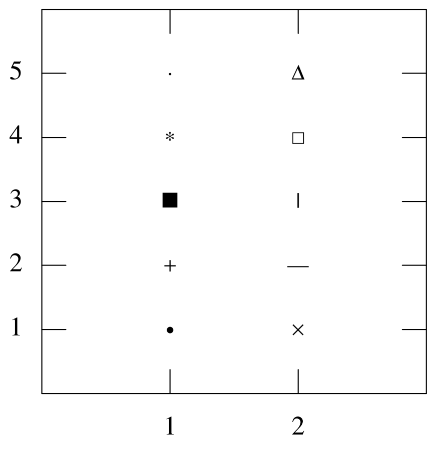
A large but boring graph. No data, but it does show off passing size parameters to *pic*, and using non-brace macro delimiters.

```
.ps 14
.vs 18
.G1 4
frame ht 2 wid 2
label left "Response Variable" left .2
label bot "Factor Variable" down .1
line from 0,0 to 1,1
line dotted from .5,0 to .5,1
define blob X "\v'.1m'\(bu\v'-.1m'" X
blob at 0,.5; blob at .5, .5; blob at 1,.5
.G2
.ps 10
.vs 12
```



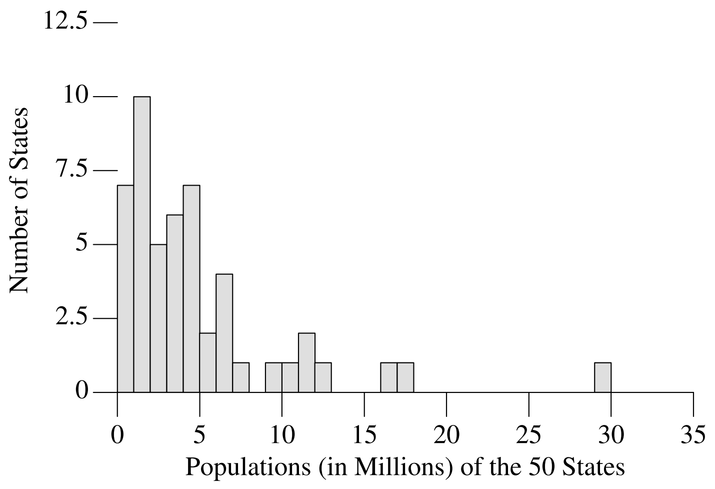
This displays all the macros defined in `grap.defines`. No data.

```
.G1
define box {"\s-2\f(ZD\N'110'\fP\s0"}
frame ht 2 wid 2
coord x 0,3 y 0,6
ticks off
ticks left in left .1 from 1 to 5
ticks right in from 1 to 5 ""
ticks bot in down .1 from 1 to 2
ticks top in from 1 to 2 ""
bullet at 1,1
plus at 1,2
box at 1,3
star at 1,4
dot at 1,5
times at 2,1
htick at 2,2
vtick at 2,3
square at 2,4
delta at 2,5
.G2
```



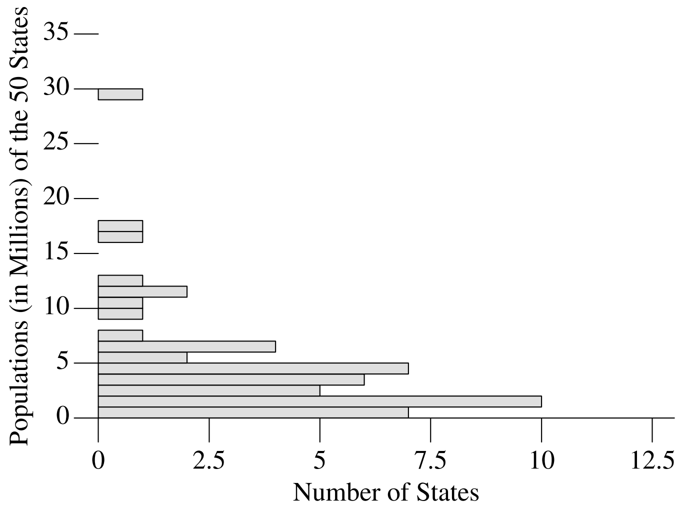
We saw this graph earlier using line drawing commands. This version uses the `bar` extension. Bars are centered, so the 0.5 unit fudge factor is used. If I were graphing this from scratch, I would rewrite the *awk* script to place the bars correctly.

```
.G1
frame invis bot solid
label bot "Populations (in Millions) of the 50 States"
label left "Number of States" left .1
ticks bot out from 0 to 35 by 5
coord x 0, 35 y 0, 13
copy "states2.d" thru {
    bar up $1+0.5 ht $2 fill 0.125
}
.G2
```



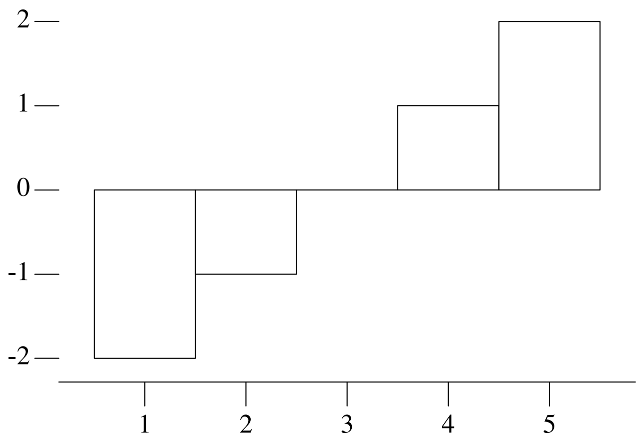
The same graph with bars extending in the x direction.

```
.G1
frame invis bot solid
label left "Populations (in Millions) of the 50 States"
label bot "Number of States"
ticks left out from 0 to 35 by 5
coord x 0, 13  y -0, 35
copy "states2.d" thru {
    bar right $1+0.5 ht $2 fill 0.125
}
.G2
```



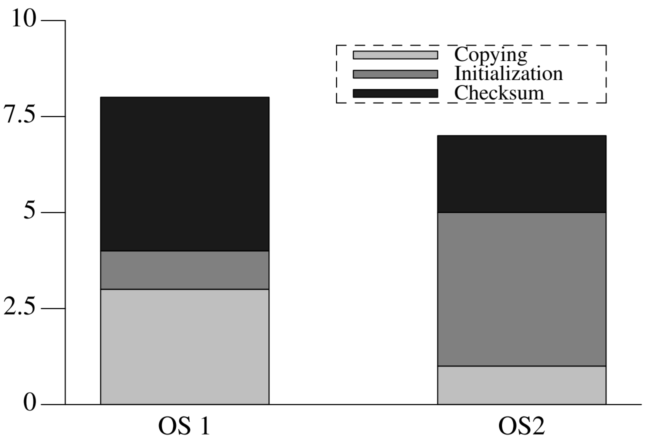
Demonstration of negative bar heights.

```
.G1
frame invis bot solid
copy until "DONE" thru {
    bar up $1 ht $2
}
1 -2
2 -1
3 0
4 1
5 2
DONE
.G2
```



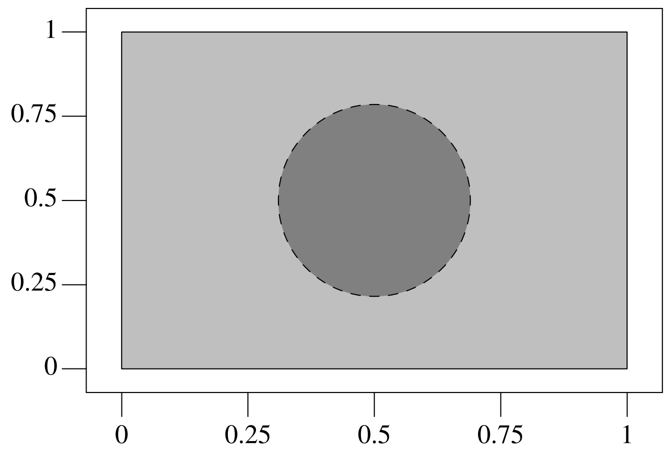
A display of two timing measurements and the components thereof. The data is fabricated, I just wanted to show the format, including both formats of `bar` statement.

```
.G1
frame invis bot solid left solid
ticks bottom off
grid bottom invis at 1 "OS 1", 2 "OS2"
coord y 0, 10
copy until "DONE" thru {
    bar up $1 ht $2 wid 0.5 base $3 fill $4
}
1 3 0 0.25
1 1 3 0.5
1 4 4 0.9
2 1 0 0.25
2 4 1 0.5
2 2 5 0.9
DONE
copy until "DONE" thru {
    bar 1.5,$1, 1.75, $1+0.2 fill $2
    $3 size -2 ljust at 1.8,$1+0.1
}
9 0.25 "Copying"
8.5 0.5 "Initialization"
8  0.9 "Checksum"
DONE
bar 1.45,9.35, 2.25, 7.85 dashed
.G2
```



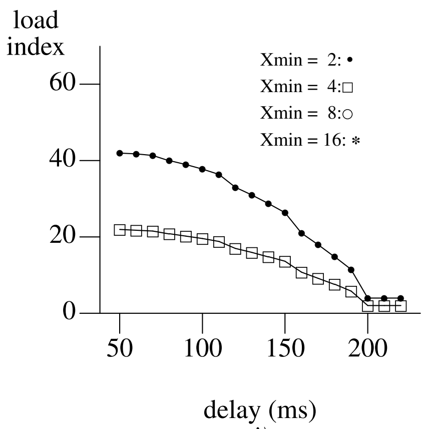
This shows the new filling and linestyle capabilities of the `circle` and `bar` commands

```
.G1
bar (1,1), (0,0) fill 0.25
circle at 0.5,0.5 rad 0.5 dashed 0.05 fill 0.5
.G2
```



This is a more complex example of nested macros from Anindo Banerjea's thesis. This originally caught errors in the use of multiple positioning commands for labels. I don't know what the data are, but a full explanation is in his Ph.D thesis. In-line data and the groff for the caption are elided. Two lines present in the real graph are missing from this one.

```
.ds ** \v'+.2m'\fB*\fP\v'-.2m'
.vs 10
.G1
define myplot % # (symbol)
next linename at $1,$2
symb at $1,$2
%
graph A
frame invis ht 1.667 wid 2.0 left solid bot solid
coord y 0,70
label bot "delay (ms)" "\fIi)\fP" down 0.3
label left "load" unaligned "index" up .9
label right "Xmin =  2: \(bu" unaligned ljust size -3 "Xmin =  4:\(sq" \
     "Xmin =  8:\(ci" "Xmin = 16: \*(**" left 1.4 up 0.5
draw a solid
define symb % bullet %
define linename { a }
copy until "XXX" thru myplot
50  42.000
...
XXX
draw b solid
define symb % square %
define linename { b }
copy until "XX1" thru myplot
50  22.000
...
XX1
.G2
```

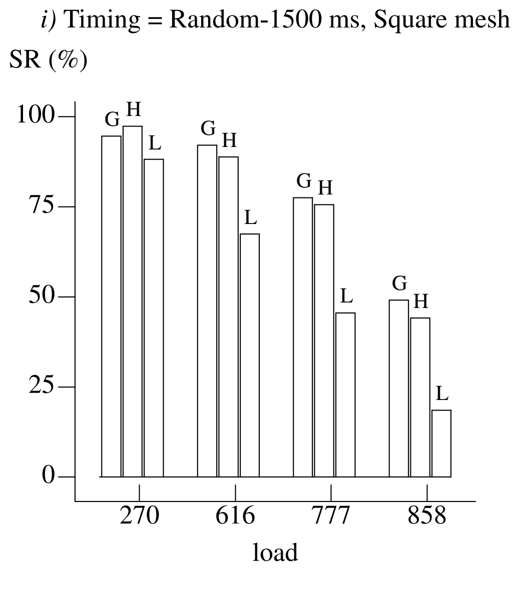


i)

Figure 1. Queueing Delay index vs. other load indices: Graph *i)* shows that the proposed load index is monotonically decreasing as a function of delay if all other factors are constant. Graph *ii)* shows that it is linear in bandwidth and graph *iii)* shows that it is linear in the path length of the route provided the delay requirement per hop remains constant.

This graph is pretty brutal all around. Nested macros and constructed filenames produce a bar graph (some of the components separated by dots in the included filenames were originally pathname components). This source is also from Anindo Banerjea, and caught bugs related to `if` and `for` statements. Anindo wrote this to display a graph for his thesis, not to demonstrate perfect `grap` programming form. His graphs are used with his permission, I've made minor tweaks to his code to make them more pretty under this version of `grap`.

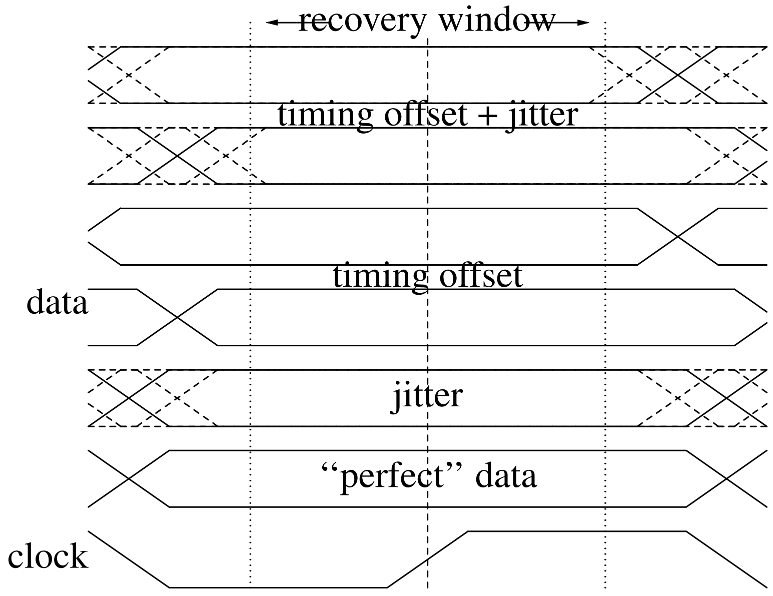
```
.G1
define write_name % #(code,X,Y)
  if ($1 == 1) then {
    "\s-2G\s+2" above at $2,$3
  }
  if ($1 == 2) then {
    "\s-2H\s+2" above at $2,$3
  }
  if ($1 == 3) then {
    "\s-2L\s+2" above at $2,$3
  }
  if ($1 == 4) then {
    "\s-2I\s+2" above at $2,$3
  }
  if ($1 == 5) then {
    "\s-2R1\s+2" above at $2,$3
  }
  if ($1 == 6) then {
    "\s-2R2\s+2" above at $2,$3
  }
  if ($1 == 7) then {
    "\s-2R3\s+2" above at $2,$3
  }
  if ($1 == 8) then {
    "\s-2S\s+2" above at $2,$3
  }
%
define dashedbar % #(X,Y)
  line from $1,0 to $1,$2 dashed
  line from $1,$2 to $1+9,$2 dashed
  line from $1+9,$2 to $1+9,0 dashed
%
define abar % #(X,Y)
  line from $1,0 to $1,$2
  line from $1,$2 to $1+9,$2
  line from $1+9,$2 to $1+9,0
%
define bar_succ % #(load,y,junk)
  abar(curx,$2)
  write_name(name,curx+5,$2)
  curx = curx + jump
  if (Maxy < $2) then { Maxy = $2 }
%
define bar_goodness % #(load,y,junk)
  abar(curx,$3-100)
  write_name(name,curx+5,$3-100)
  curx = curx + jump
  if (Maxy < ($3-100) ) then { Maxy = $3-100 }
%
define bar_avmax % #(load,average,max)
  abar(curx,$2)
  dashedbar(curx,$3)
  write_name(name,curx+5,$3)
  curx = curx+jump
  if (Maxy < $3) then { Maxy = $3 }
%
define bar_set % #(filename,jump,startX,name,function)
  jump = $2
  curx = $3
  name = $4
  copy $1 thru bar_$5
%
define setload % #(load, junk, junk
  ticks bot in 0.0 at ((2*count+1)*jump-gap/2)/2 "$1"
  count = count + 1
%
define mkgraphTiming % #(Timing,Name,file,function,number,ylabel,dir,toplable)
  Maxy = 0.0
  Nbars = 3
  start = 1
  gap = 15
  jump = Nbars * 10 + gap
  bar_set("$7.Global.$1.$3.result",jump,start,1,$4)
  bar_set("$7.Hybrid.$1.$3.result",jump,start + 10,2,$4)
  bar_set("$7.Local.$1.$3.result",jump,start + 2*10,3,$4)
  line from 0,0 to jump*4-gap,0
  count = 0
  copy "$7.Local.$1.$3.result" through setload
  label top "\fI$5)\fP Timing = $2, $8" up .2
  label left $6 unaligned up 1.8 right .2
  label bot "load"
  frame invis ht 3 wid 3 left solid bot solid
%
graph A
   mkgraphTiming(Random1500,Random-1500 ms,succ,succ,i,"SR (%)",\
     result.SQ_MESH.Fail1.S3.R0,Square mesh)
.G2
```



This is a graph donated by Bruce Lilly, after it demonstrated a couple `grap` bugs and incompatibilities with DWB `grap`. But mostly I include it because it's a cool timing diagram, although I have no idea for what. I shrunk it for this example, so this version is a little more cramped than his. Remove the 4 that follows the .G1 to see the diagram in its full glory.

```
.G1 4
.ps 14
frame ht 5 wid 6 top invis right invis left invis bot invis
ticks off
coord x -21, 21 y 0, 70
line dashed from (0,0) to (0,68)
line dotted from (-11,0) to (-11,70)
line dotted from (11,0) to (11,70)
"clock" rjust at -21, 3.5
"data" rjust at -21, 35
"timing reference" below at 0, -1
"recovery window" at 0, 70
arrow from (-6,70) to (-10,70)
arrow from (6,70) to (10,70)
"‘‘perfect’’ data" at 0, 13.5
"jitter" at 0, 23.5
"timing offset" at 0, 38.3
"timing offset + jitter" at 0, 58.3
.\"  clock
draw solid
[inline data elided]
.\"  perfect data
new solid
[inline data elided]
new solid
[inline data elided]
.\"  data with jitter +-3 ns
new solid
[inline data elided]
new solid
[inline data elided]
.\"  +3 ns
new dashed
[inline data elided]
new dashed
[inline data elided]
.\"  -3 ns
new dashed
[inline data elided]
new dashed
[inline data elided]
.\"  timing offset +3 ns
new solid
[inline data elided]
new solid
[inline data elided]
.\"  timing offset -3 ns
new solid
[inline data elided]
new solid
[inline data elided]
.\"  +3 ns timing offset +-3 ns jitter
.\"  +3 ns timing offset
new solid
[inline data elided]
new solid
[inline data elided]
.\"  -3 ns jitter around +3 ns timing offset
new dashed
[inline data elided]
new dashed
[inline data elided]
```

```
.\"  +3 ns jitter around +3 ns timing offset
new dashed
[inline data elided]
new dashed
[inline data elided]
.\"  -3 ns timing offset +-3 ns jitter
.\"  -3 ns timing offset
new solid
[inline data elided]
new solid
[inline data elided]
.\"  -3 ns jitter around -3 ns timing offset
new dashed
[inline data elided]
new dashed
[inline data elided]
.\"  +3 ns jitter around -3 ns timing offset
new dashed
[inline data elided]
new dashed
[inline data elided]
.ps
.G2
```

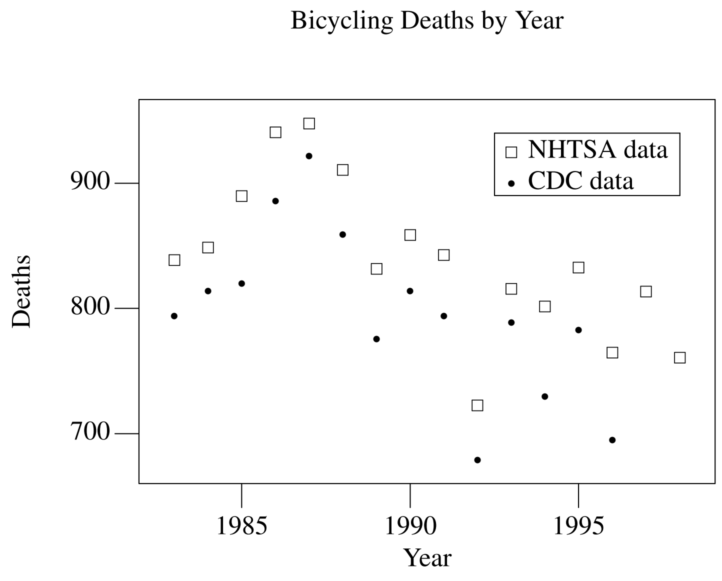


Bruce Lilly also suggests this example. He says:

> I’ve attached a small data file and simple grap specification which shows one way to handle missing data (and perhaps also how to ignore a column header row in a data file). The data is annual US bicyclist fatalities as published by the National Highway Transportation Safety Administration (*http://www.nhtsa.dot.gov/people/ncsa/factshet.html*; the “fact sheets” for “pedalcyclists” for 1998 and for 1993) and by the National Center for Injury Prevention and Control (*http://www.cdc.gov/ncipc/osp/us9693/mvpctr.htm*; change the numbers for different years). CDC data hasn’t yet been published for 1997 or 1998. Don’t ask me why two groups of bureaucrats in the US Federal Government can’t agree on the numbers. The data is, of course, public information, and there’s no reason why you can’t use the trivial grap specification (though you might wish to embellish it).

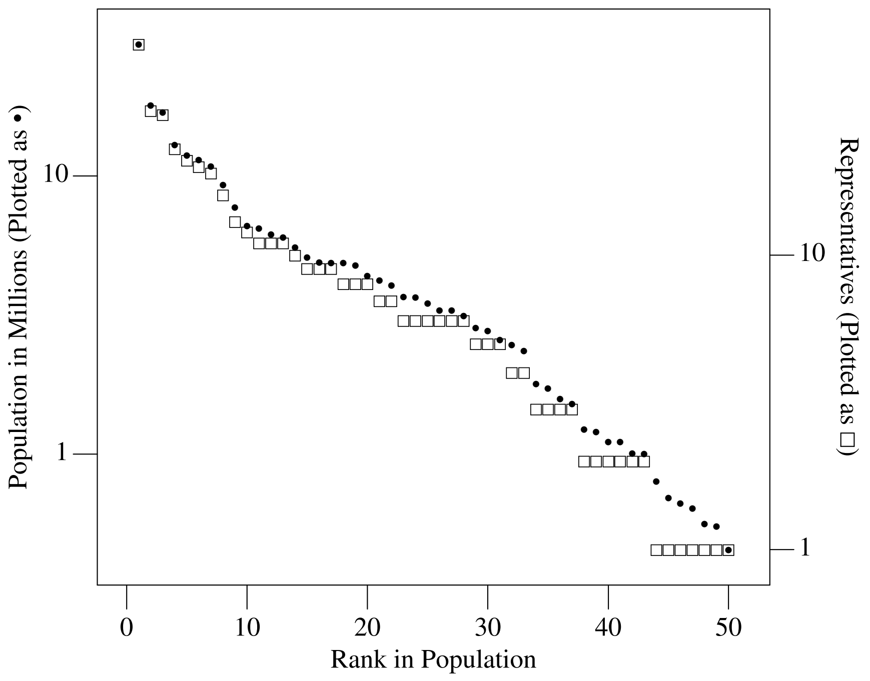
I did embellish it by adding labels and a key.

```
.G1
label top "Bicycling Deaths by Year"
label left "Deaths" left 0.2
label bottom "Year"
copy "cy_fatal.d" through {
    # ignore missing data (zero or negative values
    # used as placeholders). It also happens to ignore column
    # header rows in data.
    if $2 > 0 then {square at $1,$2 }
    if $3 > 0 then {bullet at $1,$3 }
}
square at 1993, 925
"NHTSA data" ljust at 1993.5, 925
bullet at 1993, 900
"CDC data" ljust at 1993.5, 900
line from 1992.5,940 to 1998,940
line from 1998,940 to 1998,890
line from 1998,890 to 1992.5, 890
line from 1992.5, 890 to 1992.5,940
.G2
```



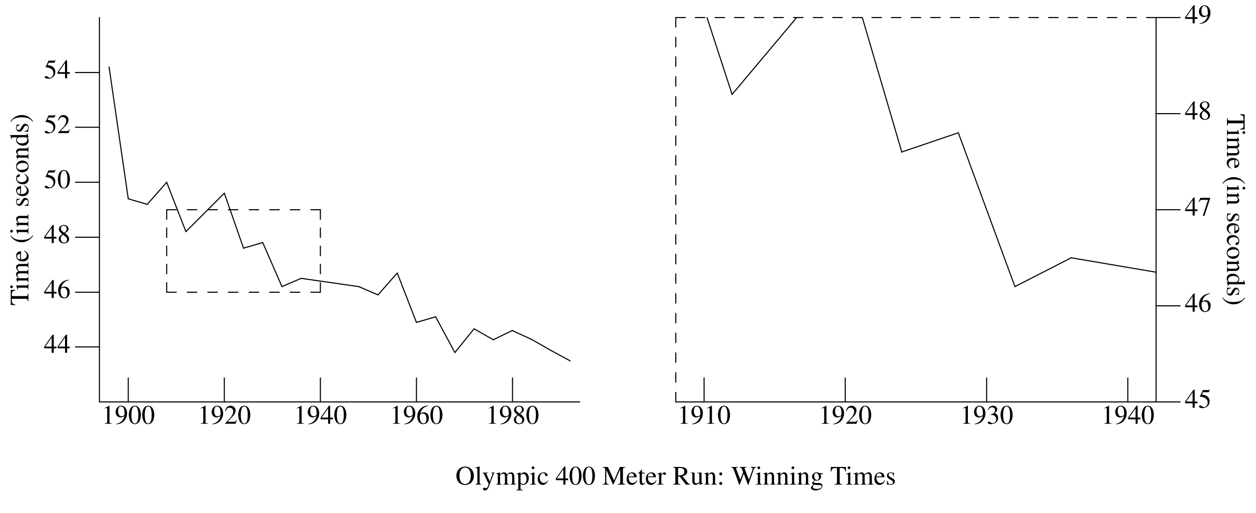
This example shows off the new automatic tick generation for named coordinate systems. It is a re-plot of earlier data. The automatic ticks do not look great here because they are generated on a logarithmic axis. Still, it means that the graph can be generated without constants in the grap code.

```
.G1
define square {"\s-2\(sq\s0"}
frame ht 3 wid 3.5
label left "Population in Millions (Plotted as •)"
label bot "Rank in Population"
label right "Representatives (Plotted as □)"
coord pop log y
coord rep log y
ticks left out auto pop
ticks bot out auto pop
ticks right out auto rep
thisrank = 50
copy "states.d" thru {
    bullet at pop thisrank,$3/1e6
    square at rep thisrank,$2
    thisrank = thisrank -1
}
.G2
```



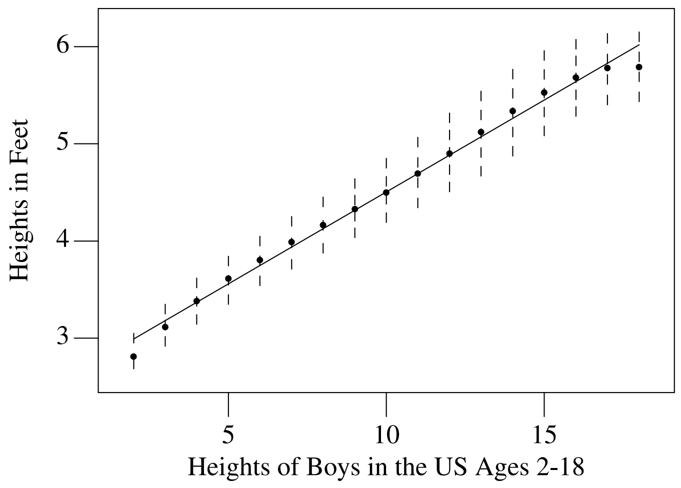
This is a demonstration of the new line clipping. The right graph is a detail of the left, created by simply respecifying the coordinate limits. (The region plotted in the right graph is the dotted rectangle in the left.) Note that the data is clipped leaving the area plotted and returning. The data is the 400 Meter Run Olympic data from before.

```
graph Main
frame invis ht 2 wid 2.5 left solid bot solid
label left "Time (in seconds)"
label bot "Olympic 400 Meter Run: Winning Times" right 1.75
coord x 1894, 1994 y 42, 56
ticks left out at 44 "44", 46, 48 "48", 50, 52 "52", 54
ticks bot in from 1900 to 1980 by 20
draw solid
copy "400mpairs.d"
bar (1908,46), (1940,49) dashed
# detail graph
graph Detail with .w at Main.Frame.e +(0.1,0)
frame dashed ht 2 wid 2.5 right solid bot solid
label right "Time (in seconds)"
ticks left off
ticks right on
ticks bot in
draw solid
copy "400mpairs.d"
coord x 1908,1942 y 45,49
```



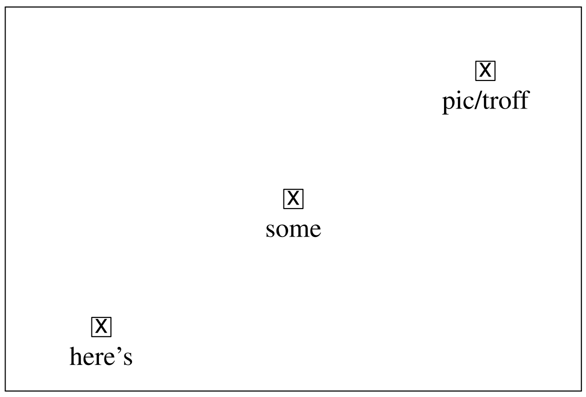
Revisiting the boy heights example to demonstrate some other `grap` features. New to this version is the ability to use x and y directly as variable names anywhere but a `coord` statement. Fixed as of release 1.10 is the ability to start a new line of the named style (or the default if no name is given) using the `new` statement. The variable `y` is directly used in the computation of the regression, and each dashed error bar is drawn using the conf line style. Otherwise this is the earlier graph.

```
.G1
label left "Heights in Feet"
label bot "Heights of Boys in the US Ages 2-18"
cmpft = 12
minx = 1e12; maxx = -1e12
n = sigx = sigx2 = sigy = sigxy = 0;
draw conf dashed
copy "boyhts.d" thru {
    new conf
    next conf at $1, $2/cmpft
    next conf at $1, $4/cmpft
    y = $3 / cmpft
    bullet at $1, y
    n = n+1
    sigx = sigx + $1; sigx2 = sigx2 + $1 * $1
    sigy = sigy + y; sigxy = sigxy + $1*y
    minx = min(minx,$1); maxx = max(maxx,$1);
}
slope = ( n*sigxy - sigx* sigy) / (n*sigx2 - sigx * sigx)
inter = ( sigy - slope * sigx) / n
line from minx, slope * minx+inter to maxx, slope * maxx + inter
.G2
```



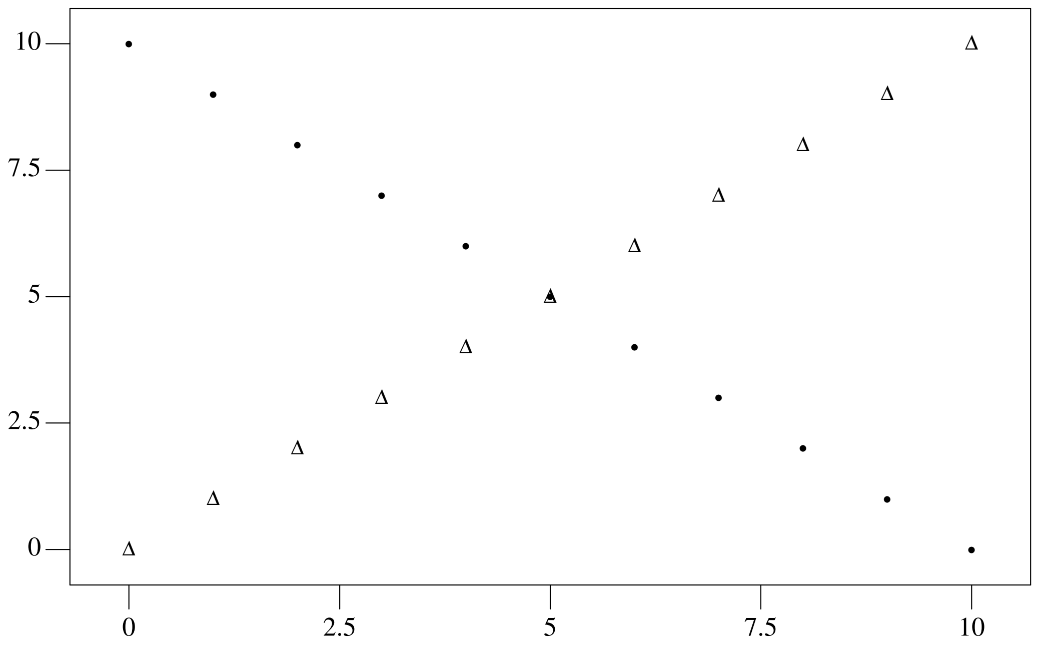
Some fun with embedded pic and troff to show that they work. The plotting points are made by a troff font change to Helvetica, drawing an “x” in grap, then a pic box around it, a second troff font change and another grap label. This example will almost certainly fail under TeX.

```
.G1
coord x 5,35 y 5, 35
ticks off
copy until "END" thru {
.    ft H
     "x" at $1, $2
     pic box ht 0.1 wid 0.1 at Here
.    ft R
     "" "$3" below at $1, $2
}
10 10 here’s
20 20 some
30 30 pic/troff
END
.G2
```



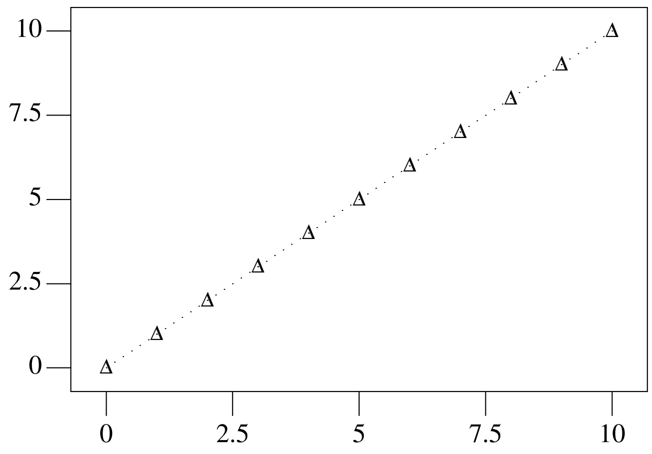
`grap` is now *much* more context sensitive in how it picks out keywords. There are still some places where it will resolutely not allow variables with keyword names, but it's enormously better about it. Here's a short somewhat pathological example. I couldn't decide which for loop was less readable, so I used both.

```
.G1
from=0
to = 10
ht =3
wid = 5
frame ht ht wid wid
for i from from to to do {
   delta at i, i
}
for j = from to to do {
   bullet at to - j, j
}
.G2
```



At the request of Bruce Lilly, coordinate spaces and variables now have distinct name spaces, and those names spaces can (usually) overlap both each other and keywords. The result is that things like this are legal.

```
.G1
from = 0
to = 10
new next dotted delta
for next from from to to do {
    next next at next next, next
}
ticks left on next
ticks bot on next
.G2
```

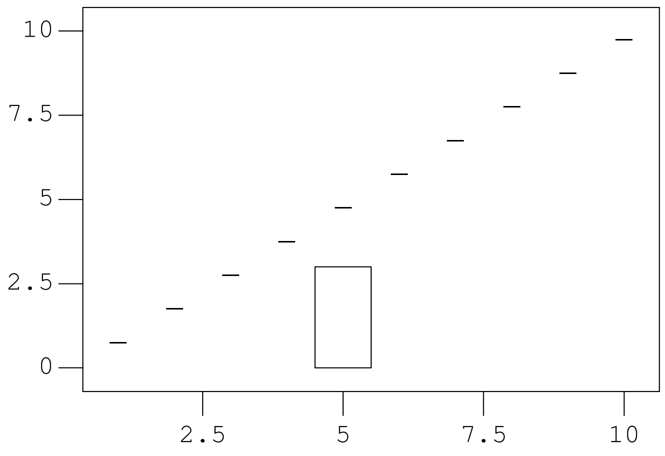


I think we're now ready for the first obfuscated `grap` contest. The following is a more readable version:

```
.G1
start = 0
finish = 10
new line_style dotted delta
coord coord_space
for i from start to finish do {
    next line_style at coord_space i, i
}
ticks left on coord_space
ticks bot on coord_space
.G2
```

One of the few things that one couldn't do under recent versions of `grap` was redefining keywords. Now you can. Here's an example that redefines the bar keyword. This also demonstrates the new undefine keyword that restores the original bar semantics.

```
.G1
define bar { "_" }
for i = 1 to 10 do {
    bar at i,i
}
undefine bar
bar up 5 ht 3
.G2
```

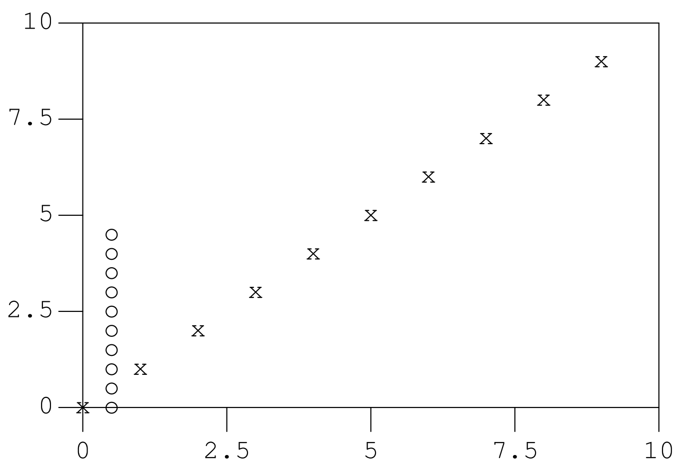


Bruce Lilly tells me that DWB `grap` defines some macros to make embedding pic code easier. The macros are named `x_` *coordname*, `y_` *coordname*, and `xy_` *coordname*. The first 2 return a pic location corresponding to that x or y coordinate in the given name space. The third generates a pair of coordinates in the given coordinate space. The default coordinate space is given by `x_gg`, `y_gg`, and `xy_gg`. If you define a coordinate space named gg, the macros will map one or the other, but I don't know which.

DWB `grap` may use these as part of their implementation. They're an extra feature for compatibility here.

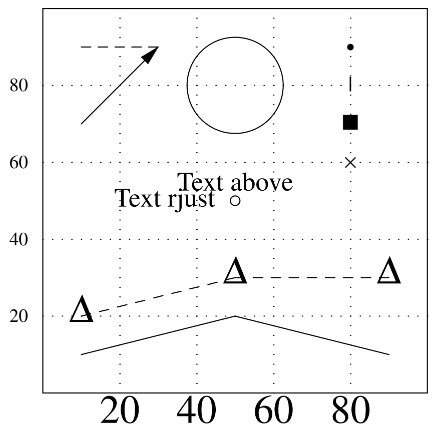
Here's a simple example of using these. Note that the fors loop are pic for loops. You don't have access to grap variables from pic.

```
.G1
coord test x 0, 10 y 0, 10
coord x 0, 20 y 0, 20
frame
ticks bot on test
ticks left on test
pic for i = 0 to 9 do {
pic "x" at Frame.Origin + xy_test(i,i)
pic }
pic for i = 0 to 9 do {
pic "o" at Frame.Origin + xy_gg(1,i)
pic }
.G2
```



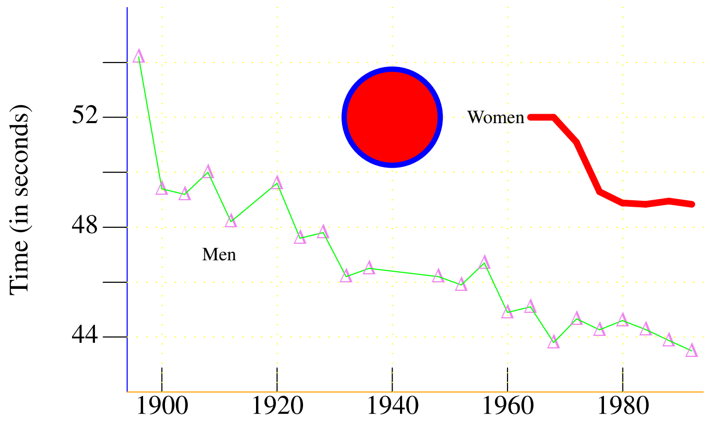
A demo of *grap*'s ability to directly manipulate label and plotting sizes. The data is all in the graph specification.

```
.G1
frame ht 2 wid 2
coord x 0,100 y 0,100
grid bot dotted from 20 to 80 by 20 "%g" size +5
grid left dotted from 20 to 80 by 20 "%g" size -3

"Text above" above at 50,50
"Text rjust   " rjust at 50,50
bullet at 80,90
vtick at 80,80
box at 80,70
times at 80,60

circle at 50,50
circle at 50,80 radius .25
line dashed from 10,90 to 30,90
arrow from 10,70 to 30,90

draw A solid
draw B dashed delta size 15
next A at 10,10
next B at 10,20
next A at 50,20
next A at 90,10
next B at 50,30
next B at 90,30
.G2
```



A gratuitous color graph. Don't ever produce anything this ugly. The data is the 400m times used earlier. There's a circle to show the color and fillcolor attributes. I've also added a thickness to one line and the circle.

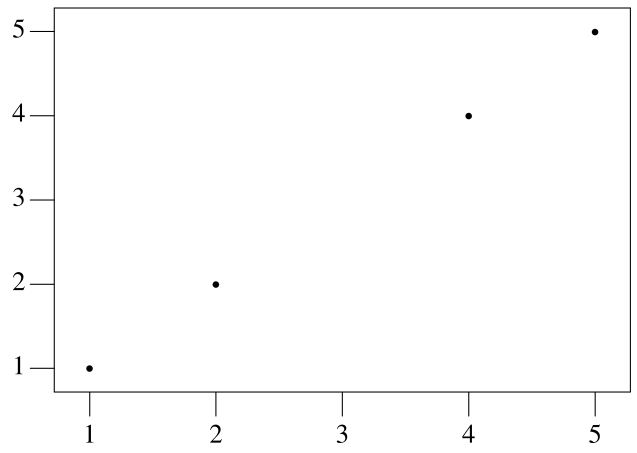
```
.G1
frame invis ht 2 wid 3 left solid color "blue" \
    bot solid color "orange"
label left "Time (in seconds)" left .15
label bot "Olympic 400 Meter Run: Winning Times"
coord x 1894, 1994 y 42, 56
ticks left out at 44 , 46 "", 48, 50 "", 52, 54 ""
ticks bot in from 1900 to 1980 by 20
grid bot ticks off  color "yellow" from 1900 to 1980 by 20
grid left ticks off  color "yellow" at 44, 46, 48, 50, 52, 54
draw solid color "green" delta color "violet"
copy "400mpairs.d"
new color "red" thickness 2.5
copy "400wpairs.d"
"Women" size -3 at 1958,52
"Men" size -3 at 1910,47
circle at 1940, 52 rad 0.25 color "blue" fillcolor "red" \
    fill 0.5 thickness 2
.G2
```



Olympic 400 Meter Run: Winning Times

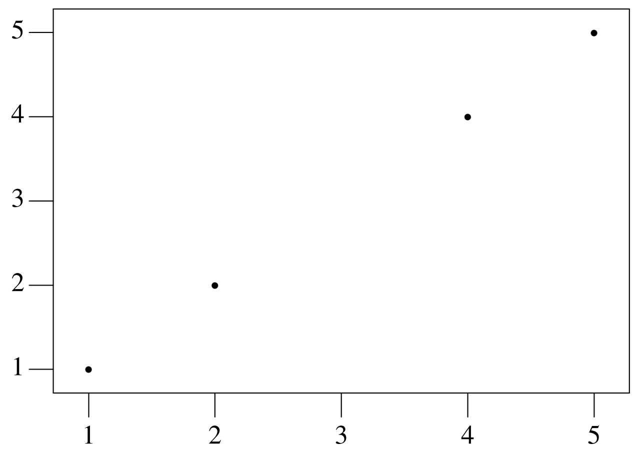
Some people have tripped over grap's fairly limited string manipulation capabilities. Simply grap can only compare two quoted strings for equality. Strings cannot be assigned to variables. Surprisingly, when combined with macros, that's enough to do simple data selection, but not much more. Here's an example.

```
.G1
copy until "END" thru {
    if ( "$1" == "yes" ) then {
        next at $2, $3
    }
}
yes 1 1
yes 2 2
no 3 3
yes 4 4
yes 5 5
END
.G2
```



Note that while data files can contain quotes, putting quotes in the data file will **not** collect words into strings containing whitespace; grap splits lines of data files into words separated by whitespace. Grap's macro substitution and line splitting are very literal and know nothing about the language structure. Overall, if you're planning anything more complex than conditionals based on the contents of a single word parameter to a macro call, do it outside grap. Perl and other scripting languages make complex text manipulations very easy. Here's the same example with quotes in the file; notice the change in the expression in the if.

```
.G1
copy until "END" thru {
    if ( $1 == "yes" ) then {
        next at $2, $3
    }
}
"yes" 1 1
"yes" 2 2
"no" 3 3
"yes" 4 4
"yes" 5 5
END
.G2
```


5
4
3
2
1
1
2
3
4
5