Define a function

```r
library(MASS)

?help

trellis.device()

On a monochrome UNIX system use
trellis.device(color=F)

x <- rnorm(1000)
y <- rnorm(1000)
truehist(c(x,y+3), nbins=25)

?truehist
dd <- con2tr(kde2d(x,y))
contourplot(z ~ x + y, data=dd, aspect=1)
wireframe(z ~ x + y, data=dd, drape=T)
levelplot(z ~ x + y, data=dd, aspect=1)

x <- seq(1, 20, 0.5)
x
w <- 1 + x/2
y <- x + w*rnorm(x)
dum <- data.frame(x, y, w)
dum
rm(x, y, w)
fm <- lm(y ~ x, data=dum)
summary(fm)

\[ y = x + (1 + \frac{w}{2})z = x(1 + \frac{w}{2}) + z = x + x \cdot \frac{w}{2} + z \]
```

Introduction

A command to make our datasets available. Your local advisor can tell you the correct form for your system.

Read the help page about how to use help.

Turn on the graphics window. You may need to re-position and re-size to make it convenient to work with both windows. Try not to change the aspect ratio of the graphics window when you re-size it.

Generate 1000 pairs of normal variates

Histogram of a mixture of normal distributions. Experiment with the number of bins (25) and the shift (3) of the second component.

Read about the optional arguments.

2D density plot. We convert its output into a form suitable for the Trellis visualization routines.

Perspective plot with superimposed coloured levels.

Greyscale or pseudo-colour plot.

Make \( x = (1, 1.5, 2, \ldots, 19.5, 20) \) and list it.

\( w \) will be used as a 'weight' vector and to give the standard deviations of the errors.

Make a data frame of three columns named \( x \), \( y \) and \( w \), and look at it. Remove the original \( x \), \( y \) and \( w \).

Fit a simple linear regression of \( y \) on \( x \) and look at the analysis.

Figure 1.1: For data.
1.3 An introductory session

fm1 <- lm(y ~ x, data=dum, weight=1/w^2)
summary(fm1)
lrf <- loess(y ~ x, dum)
attach(dum)
plot(x, y)

lines(spline(x, fitted(lrf)), col=2)
abline(0, 1, lty=3, col=3)

abline(fm, col=4)

abline(fm1, lty=4, col=5)

Since we know the standard deviations, we can do a weighted regression.

Fit a smooth regression curve using a modern regression function.

Make the columns in the data frame visible as variables.

Make a standard scatterplot. To this plot we will add the three regression lines (or curves) as well as the known true line.

First add in the local regression curve using a spline interpolation between the calculated points.

Add in the true regression line (intercept 0, slope 1) with a different line type and colour.

Add in the unweighted regression line, abline() is able to extract the information it needs from the fitted regression object.

Finally add in the weighted regression line, in line type 4. This one should be the most accurate estimate, but may not be, of course. One such outcome is shown in Figure 1.1.

You may be able to make a hardcopy of the graphics window by selecting the Print option from a menu.

Figure 1.1: Four fits and two residual plots for the artificial heteroscedastic regression data.
plot(fitted(fm), resid(fm),
    xlab="Fitted Values",
    ylab="Residuals")

qqnorm(resid(fm))
qqline(resid(fm))
detach()
rm(fm,fm1,lrf,dum)

A standard regression diagnostic plot to
cHECK for heteroscedasticity, that is, for
unequal variances. The data are gen-
erated from a heteroscedastic process, so
can you see this from this plot?

A normal scores plot to check for skew-
ness, kurtosis and outliers. (Note that
the heteroscedasticity may show as ap-
parent non-normality.)

Remove the data frame from the search
path and clean up again.

We look next at a set of data on record times of Scottish hill races against
distance and total height climbed.

hills
splom(~ hills)

brush(as.matrix(hills))

Click on the Quit button in the
graphics window to continue.

attach(hills)

plot(dist, time)
identify(dist, time,
    row.names(hills))
abline(lm(time ~ dist))
abline(lqs(time ~ dist),
    lty=3, col=4)
detach()

List the data.

Show a matrix of pairwise scatterplots
(Figure 1.2).

Try highlighting points and see how
they are linked in the scatterplots (Fig-
ure 1.3). Also try rotating the points in
3D.

Make columns available by name.

USE mouse button 1 to identify outlying
points, and button 2 to quit. Their row
numbers are returned.

Show least-squares regression line.

Fit a very resistant line. See Figure 1.4.

Clean up again.

We can explore further the effect of outliers on a linear regression by designing
our own examples interactively. Try this several times.

plot(c(0,1), c(0,1), type="n")
xy <- locator(type="p")

Make our own dataset by clicking with
button 1, then with button 2 to finish.
Figure 1.4: Annotated plot of time versus distance for hills with regression line and resistant line (dashed).

```
abline(lm(y ~ x, xy), col=4)
abline(rlm(y ~ x, xy, method="MM"),
   lty=3, col=3)
abline(lqs(y ~ x, xy),
   lty=2, col=2)
rml(xy)
```

Fit least-squares, a robust regression and a resistant regression line. Repeat to try the effect of outliers, both vertically and horizontally.

```
minimize some sum of sorted squared residuals
floor((n+p+1)/2)
```

Clean up again.

We now look at data from the 1879 experiment of Michelson to measure the speed of light. There are five experiments (column Expt); each has 20 runs (column Run) and Speed is the recorded speed of light, in km/sec, less 299,000. (The currently accepted value on this scale is 734.5.)

```
attach(michelson)
search()

plot.factor(Expt, Speed, 
   main="Speed of Light Data",
   xlab="Experiment No.")

fm <- aov(Speed ~ Run + Expt)
summary(fm)
```

Make the columns visible by name.

The search path is a sequence of places, either directories or data frames, where S-PLUS looks for objects required for calculations.

Compare the five experiments with simple boxplots. (Prior to S-PLUS 5.x, just plot suffices.) The result is shown in Figure 1.5.

Analyse as a randomized block design, with runs and experiments as factors.

```
1.4 What next?
```

We hope that you go deeply. We suggest Sections 3.1–3, and after, tackle the stati fairly independently, chapters 7 and 8 build on.

Chapter 4 comes useful to advanced u
1.4 What next?

![Boxplot](image)

**Figure 1.5**: Boxplots for the speed of light data.

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum of Sq</th>
<th>Mean Sq</th>
<th>F Value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>19</td>
<td>113344</td>
<td>5965</td>
<td>1.1053</td>
</tr>
<tr>
<td>Expt</td>
<td>4</td>
<td>94514</td>
<td>23629</td>
<td>4.3781</td>
</tr>
<tr>
<td>Residuals</td>
<td>76</td>
<td>410166</td>
<td>5397</td>
<td></td>
</tr>
</tbody>
</table>

fm0 <- update(fm, ~ - Run)
anova(fm0, fm)

Fit the sub-model omitting the nonsense factor, `runs`, and compare using a formal analysis of variance.

```
Analysis of Variance Table
Response: Speed

Terms Resid. Df RSS Test Df Sum of Sq F Value Pr(>F)
1 Expt 95 523610
2 Run + Expt 76 410166 +Run 19 113344 1.1053 0.36321
```

detach()
rm(fm, fm0)

Clean up before moving on.

Q() Quit S-PLUS.

1.4 What next?

We hope that you now have a flavour of S-PLUS and are inspired to delve more deeply. We suggest that you read Chapter 2, perhaps cursorily at first, and the Sections 3.1-3, and 3.5, plus Appendix B if you are using S-PLUS 4.x. Thereafter, tackle the statistical topics that are of interest to you. Chapters 5 to 14 are fairly independent, and contain cross-references where they do interact. Chapters 7 and 8 build on Chapter 6, especially its first two sections.

Chapter 4 comes early, because it is about S not about statistics, but is most useful to advanced users who are trying to find out what the system is really doing.