Statistical Tables and Plots using S and LaTeX

FE Harrell
Department of Biostatistics
Vanderbilt University School of Medicine
f.harrell@vanderbilt.edu

biostat.mc.vanderbilt.edu[*]

February 15, 2004

Contents

[*]Document Address: http://biostat.mc.vanderbilt.edu/twiki/pub/Main/StatReport/summary.pdf. This document was produced using TeTEX on RedHat 8.0 Linux using R version 1.6.1 and version 1.5-0 (31Jan03) of the `Hmisc` library. All commands and output will be the same for S-Plus except that Greek letters, superscripts, and subscripts will not appear in plots.

List of Tables

List of Figures

1 Introduction to LATEX

LATEX is a public domain document processing system developed by Lamport (which uses TEX by Knuth) that is used heavily in the sciences and by journal and book publishers[1]. LaTeX is a *markup language* that is compiled similar to programming languages such as C. LaTeX is particularly strong in layouts, cross–referencing, typesetting equations, making tables, bibliographic citations, indexes and tables of contents, and allowing for insertion of graphics in documents. This makes LaTeX very suitable for compiling long statistical reports such as those used to support drug licensing. For this purpose, major advantages of LaTeX include the ability to automatically create cross–references and to automatically update a report if any of its component graphics figures or tables changes. To accomplish the latter capability, the analyst merely re–runs the statistical program that produced the graphics or table components. These graphics and tables are read respectively by LaTeX by an `\includegraphics{}` or `\input{}` command, so running the `latex` command to recompile to report will make any needed updates. This is in distinction to Microsoft Word, which does not have a batch inclusion capability.

Everything in a LaTeX source document is plain text, so you can edit these documents using any text editor[2] and E–mail them to anyone. LaTeX is based on the philosophy that the writer should have an easy time composing and editing text[3] but she should not have to spend time making text look good on the screen. Instead the writer needs to concentrate on the logical elements of composition; LaTeX's job is to make the final output look good.

1.1 Two LaTeX Output Modes

When the `latex` command is run to compile your LaTeX source code, LaTeX produces a dvi ("device independent") file containing the typeset document in a very compact form. Graphics are not included in the dvi file, but pointers to the graphics files are included. The dvi file can be printed directly, or it can be converted into a self–contained postscript or pdf file. Here are some example LaTeX-related system commands.

[1]LaTeX is available on many platforms. Excellent free versions for Microsoft Windows are FPTeX by Fabrice Popineau and MikTeX, both available at `www.ctan.org`. An excellent free book on LaTeX is available at `ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf`

[2]The `Emacs` editor has a special mode for editing LaTeX text that makes composing text much easier.

[3]For example, with one `Emacs` command you can change the first word of every figure caption to be in another font, or change the size of all included figures.

```
latex myfile                    % create myfile.dvi from myfile.tex
dvips myfile                    % send myfile to a postscript printer
dvips -o myfile.ps myfile       % convert myfile.dvi to myfile.ps, with
                                % graphics
dvips -Pwww -o myfile.ps myfile    % use Type 1 fonts
dvipdfm myfile                  % convert myfile.dvi to myfile.pdf
pdflatex myfile                 % creates myfile.pdf directly if no
                                % postscript graphics are referenced
```

Creation of a static document in one of these ways is the usual mode of LATEX usage. There is also a way of using LATEX to create "live" documents that are viewed on a monitor (either locally or over the web) or printed. These pdf documents may contain bookmarks, hyperlinks to external URLs, links to E–mail addresses, etc. If you use the `hyperref` package in LATEX, the system will automatically make all pertinent elements of your document cross–indexed and hyperlinked, and you can also insert special commands to link to areas outside the document such as URLs and E–mail.

When viewing the document using Adobe Acrobat Reader, bookmarks can appear in the left margin, allowing the user to click to jump to any major section of the document. Sections having sub–sections can have their bookmarks expanded so that you can jump to the sub–sections. You can jump to any figure while viewing the `List of Figures` and to any table while viewing the `List of Tables`, in addition to jumping to any area while viewing the `Table of Contents`. If your document is indexed, you can jump to any page for which an indexed phrase is discussed. You can optionally jump to pages in which a given article is cited while viewing the `Bibliography`, in addition to the more standard jump from a citation to the bibliographic reference. If the `colorlinks` option is selected (see code below), symbols that are hyperlinked appear in color; clicking on them will cause the jump. All of this is set up automatically by `hyperref`, unlike the large number of flags that must be put in a document manually if using Microsoft Word.

Instead of compiling the document using the `latex` system command, you use the `pdflatex` command to create the pdf file directly, with all bookmarks and hyperlinks.

This document was created in the fashion just described. `PDF` graphics files were created directly using an S `pdf` device driver. Below you will find the code in the preamble of the document that set up the `pdf` document with hyper–referencing.

```
\usepackage[pdftex,bookmarks,pagebackref,colorlinks,pdfpagemode=UseOutlines,
    pdfauthor={Frank E Harrell Jr},
    pdftitle={Statistical Tables and Plots using S and LaTeX}]{hyperref}
```

1.2 Basic Table Making in LaTeX

LaTeX has excellent facilities for composing and typesetting tables. Table 1 is an example of a user–specified table using three macros — `btable` (begin table), `etable` (end table), and `mc` (headings that span multiple columns). These macros save repetitive operations. Macros are usually defined at the top of the document.

```
%Usage: \btable{table specs}{caption}{reference label}
\newcommand{\btable}[3]{
        \begin{table}[htbp]
        \begin{center}
        \caption{#2\label{#3}}
        \begin{tabular}{#1}}

\newcommand{\etable}{\end{tabular}
        \end{center}
        \end{table}}

%Usage: \mc{number of columns spanned}{major column heading}
\newcommand{\mc}[2]{\multicolumn{#1}{c}{#2}}

\btable{l|ccccc}{Overall Results}{results} \hline\hline
%6 fields, justified left, center x 5
%double horizontal line at top, 1 vertical bar
 & \mc{2}{Females} & & \mc{2}{Males} \\   % column 4 blank, for spacing
\cline{2-3} \cline{5-6}      % horizontal lines connecting cols. 2-3, 5-6
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \\ \hline
Placebo       & 0.21 & 163 & & 0.22 & 164 \\
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \\
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \\ \hline
\etable
```

Table 1: Overall Results

	Females		Males	
Treatment	Mortality	Mean Pressure	Mortality	Mean Pressure
Placebo	0.21	163	0.22	164
ACE Inhibitor	0.13	142	0.15	144
Hydralazine	0.17	143	0.16	140

The result is Table 1. However, the `ctable` style, available from `www.ctan.org` can produce prettier tables more flexibly:

```
\ctable[caption={Overall Results},label=resultsb,pos=hbp!]{lccccc}{}{
\FL
& \mc{2}{Females} & & \mc{2}{Males} \NN
\cmidrule{2-3}\cmidrule{5-6}  % Important: no space before \cmidrule
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \ML
Placebo       & 0.21 & 163 & & 0.22 & 164 \NN
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \NN
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \LL
}
```

The result is shown in Table 2.

Table 2: Overall Results

	Females		Males	
Treatment	Mortality	Mean Pressure	Mortality	Mean Pressure
Placebo	0.21	163	0.22	164
ACE Inhibitor	0.13	142	0.15	144
Hydralazine	0.17	143	0.16	140

2 Using S to Fill in Cells in LaTeX Tables

For most statistical tables a better idea is to avoid transcription of calculated values by having the values inserted into tables automatically. The `Hmisc` library (see `biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc`) contains several S functions by R Heiberger and F Harrell that automatically make LaTeX tables from S objects[4]. S functions that automatically produce LaTeX code from S objects (matrices, fitted models, data summaries, etc.) have names that start with `latex`. Tables produced by the `latex.*` functions in `Hmisc` meet the stylistic requirements of most journals, i.e., by default they do not use vertical lines and they use horizontal lines only when needed. In this way the lines do not distract from delivering the statistical information.

Suppose that some calculations have already been made using S, and these calculations were not stored. For example, you may have estimated various effects and standard errors but forgot to store the S regression fit objects so that you can pull these values into tables automatically. You can use the `latex.default` function that is part of `Hmisc` for automatic conversion of the calculations into LaTeX, after entering basic statistics manually. Let us have S calculate odds ratios and P–values to avoid transcribing them after we print $\hat{\beta}$ and standard errors. Here is the S program for creating the table that is inserted into this document as Table 3.

```
lor ← c(.2362, .1131, .4621, .3351)
se ← c(.1234, .0989, .1812, .1612)

chisq ← (lor/se)^2
summary.stats ← cbind(
    'Log Odds Ratio'=lor,
    'Standard Error'=se,
    'Odds Ratio'    =exp(lor),
    '$\\chi^2$'     =chisq,
    '$P$--value'    =1-pchisq(chisq,1) )
```

[4]More advanced applications of this are found in the `Design` library, such as automatic LaTeX typesetting of fitted regression models with simplification of interaction and spline terms, and typesetting of χ^2 tables showing all regression effects. These examples are beyond the scope of this document. See `biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf`, Chapter 9.

Table 3: Statistical Results

	Log Odds Ratio	Standard Error	Odds Ratio	χ^2	P–value
Fatal Events					
Death (all cause)	0.236	0.123	1.27	3.66	0.0556
Cancer Death	0.113	0.099	1.12	1.31	0.2528
Non–fatal Events					
Relapse	0.462	0.181	1.59	6.50	0.0108
Hospitalization	0.335	0.161	1.40	4.32	0.0376

```
# $..$  : puts .. in math notation  (^=superscript)
# --    : LaTeX medium length dash

summary.stats      # ordinary print

library(Hmisc)  # get access to library

w ← latex(summary.stats, cdec=c(3, 3, 2, 2, 4), col.just=rep('c',5),
          rowname=c('Death (all cause)','Cancer Death',
                    'Relapse','Hospitalization'),
          rgroup=c('Fatal Events','Non--fatal Events'),
          rowlabel='', caption='Statistical Results',
          ctable=TRUE)
# Table 3

# Assign the latex to an object (w) so that it doesn't try to print now
# cdec  : Number of decimal places for the different columns
# col.just: justification of columns in table (all centered here)
```

There are many other options to the basic `latex` function. Type `?latex` to access the online help. You may be particularly interested in the `longtable` option, which can be used to easily break a long table into multiple pages (with repetitions of key header information).

You can have your S program print hardcopy LaTeX output directly using the `prlatex` function. More typically though you will want the program to create LaTeX files (with suffix `.tex`) that will be put together later. In this way you can add title pages, running headers or footers, and other text, and refer to tables by symbolic names. This document serves as an example of how this is done, with its LaTeX code listed in Section 10.

If you like to specify table layouts inside the LaTeX source file rather than inside S, you can have your S program output symbolic values to a file that is `\input{}`'d in LaTeX as shown in the following example. A restriction is that variable names defined to LaTeX may contain only letters and they should not coincide with names of LaTeX commands.

```
chisq <- (beta/se)^2
pval  <- 1 - pchisq(chisq, 1)
cat('\\def\\chisq{',round(chisq,2),'}\\n',   # \\ -> \ in parms.tex
    '\\def\\pval{',round(pval,4),'}\n', sep='', file='parms.tex')
```

If LaTeX variables are named the same as S variables, and the names contain only letters, code can be simplified using a little function. This function can also convert `NA`s to blanks.

```
lvar <- function(x, digits=2)
  paste('\\def\\',substitute(x),'{',
        ifelse(is.na(x),'',round(x,digits)),'}', sep='')

cat(lvar(chisq), lvar(pval,4), sep='\n', file='parms.tex')
```

The contents of file `parms.tex` will look like the following:

```
\def\chisq{3.84}
\def\pval{0.05}
```

Inside the main LaTeX source file use for example

```
\input{parms}
\ctable[caption={Main Results},label=resultsc]{lcc}{}{
Test        & $\chi^2$    & $P$--value \ML
Age effect  & \chisq      & \pval      \LL
}
```

3 Using S to Create Graphics for LaTeX

PostScript is a graphics format accepted by all versions of LaTeX as long as you have a PostScript printer or have GhostScript or Adobe Acrobat Distiller to convert postscript output to other formats. The basic graphics driver in S for creating postscript files is the `postscript` function. For creating 35mm slides, overhead transparencies, or 5×7 glossy figures, the `ps.slide` function in the Hmisc library assists in setting up nice defaults for postscript images. For reports and books, the Hmisc `setps` function makes creating of individual postscript graphics easy. `setps` uses reasonable defaults and sets up for a minimally sized bounding box. It tries not to waste space between axes and axis labels. In the following example, a file called `test.ps` is created in the current working directory. Note the absence of quote marks around the word `test` below.

```
setps(test)        # use setps(test, trellis=T) if using Trellis (R Lattice)
plot(....)
dev.off()          # close file, creating test.ps
```

As you will see later, we can symbolically label this figure using the word `test` in LaTeX. By default, `setps` uses Helvetica font and makes small book–style figures. There are many options to override these and other settings.

If you are using `pdflatex`, graphics files must be in Adobe `pdf` format. You can create `pdf` files directly in S-Plus using the builtin `pdf.graph` function, in R using the `pdf` function, or using the Hmisc `setpdf` function. In older versions of S-Plus, better results are obtained by creating postscript and converting the graph to `pdf`. If you have Ghostscript installed and have used `setps` followed by `dev.off`, you can type `topdf()` with no arguments to invoke Ghostscript from S to create, in this case, `test.pdf`. You can also convert from postscript to `pdf` using Adobe Acrobat Distiller, which produces more compact `pdf` files. In R, direct creation of `.pdf` files seems to work well.

3.1 Inserting Graphics Files into LaTeX Documents

The standard `graphics` and `graphicx` packages in LaTeX provide all you need to insert postscript and `pdf` graphics into a document in a flexible fashion.

This is not to say that it is as easy as using Word; frequently some trial and error is required to get graphics to have an appropriate scaling (magnification) factor. Inclusion of `pdf` graphics in older versions of S-Plus and LaTeX frequently resulted in much wasted spaced before and after the graph when using `pdflatex`, so you often had to use `\vspace` commands with negative arguments when including `pdf` files.

Here is a LaTeX macro for inserting `pdf` graphics files, which are assumed to have a `.pdf` suffix.

```
% Usage: \fig{label=.pdf prefix}{caption}{short caption for list of
% figures}{scalefactor}
\newcommand{\fig}[4]{\begin{figure}[hbp!]
 \leavevmode\centerline{\includegraphics[scale=#4]{#1.pdf}}
 \caption[#3]{\small #2}
 \label{#1}
 \end{figure}}
```

For example, `\fig{test}{long caption}{short caption}{.8}` will insert `test.pdf` and reduce its size by 20%. You can refer to this figure in the text using for example `see Figure~\ref{test}`.

4 Making S Compose LaTeX Tables

In many cases S functions can be used to make all calculations for the table and then to create the LaTeX table. Harrell's S `summary` function for formulas (actually `summary.formula`) is one function that will do this when what you need is descriptive statistics (including statistics computed by functions you create). `summary` is in the Hmisc library available at `biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc`. It has three methods for computing descriptive statistics on univariate or multivariate responses, subsetted by categories of other variables. See *An Introduction to S and the Hmisc and Design Libraries* by CF Alzola and FE Harrell (`biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf`) for more information about `summary.formula` and S usage in general, especially information on how to recode and reshape data to be used in reports.

The output from `summary.formula` can be printed (for ordinary text file printouts), plotted (dot charts or occasionally box-percentile plots), or typeset using LaTeX, as there are several `print`, `plot`, and `latex` methods for objects created by `summary.formula`. The `latex` methods create all the needed table elements, then invoke the `latex.default` method in `Hmisc` to build the complete set of LaTeX commands to make each table.

The method of data summarization to be done by `summary.formula` is specified in the parameter `method`. These methods are defined below. For the first and third methods, the statistics used to summarize the data may be specified in a flexible manner by the user (e.g., the geometric mean, 33^{rd} percentile, or Kaplan–Meier 2–year survival estimate, mixtures of several statistics). The default summary statistic is the mean, which for a binary response variable is the proportion of positive responses.

`method='response'`: The response variable may be multivariate, and any number of statistics may be used to summarize the responses. Sometimes dependent variables are multivariate because they indicate follow–up time and censoring, and sometimes they are multivariate because there are several response variables (e.g., systolic and diastolic blood pressure). The responses are summarized separately for each independent variable (independent variables are not cross–classified). Continuous independent variables are automatically stratified into quantile groups. One or more of the independent variables may be stratification factors, in which all computations are done separately by levels of these categorical variables. The stratification variables form major column groupings in tables. For multivariate responses, subjects are considered to be missing if *any* response variable is missing.

`method='reverse'`: This format is typical of baseline characteristic tables describing the usual success of randomization. Here the single dependent variable must be categorical (e.g., treatment assignment), and the "independent" variables are broken down separately by the dependent variable. Continuous independent variables are described by three quantiles (quartiles by default), and categorical ones are described by counts and percentages. There is an option to automatically generate test statistics for testing across columns of `'reverse'` tables.

`method='cross'`: The `'cross'` method allows allows for multiple dependent variables and multiple statistics to summarize each one. If there is more than one independent variable (up to three is allowed), statistics are computed separately for all cross–classifications of the independent variables, and marginal and overall statistics may optionally be computed. `summary.formula` for this method outputs a data frame containing the combinations of predictors along with the response summaries. This data frame may be summarized graphically in various ways using the S-Plus `trellis` library or R `lattice` package[5]. A LaTeX printing method, for the case where there is exactly two predictors, typesets a two–way table where the first predictor forms rows and the second forms columns. Like `method='response'`, continuous variables are automatically divided into quantile groups.

The `latex` methods in the `Hmisc` library create tables using standard LaTeX commands. These tables are inserted into the master document at the desired location using an `\input{}` command. `latex` methods allow a font `size` argument. For example, you may specify `size='small'` to `latex()`, or you may want to use a generic size that is set at LaTeX run time in the document preamble. For example, specify `\def\tsz{small}` in the master document and specify `size='tsz'` to `latex()`. Then you can define (and redefine) the size for tables without modifying the individual `.tex` files created by `latex()`. Another approach using LaTeX's `relsize` style is discussed on P. 30.

4.1 Reports Formatted to Describe Responses

Tables 4–8 were produced by the S `latex` function (actually, `latex.summary.formula.response`), which is run on an object created by the `summary` function with `method='response'`, the default.

Table 4 presents Kaplan–Meier 2 and 5 year survival estimates and mean life length of subjects in the Mayo Clinic primary biliary cirrhosis dataset available from `biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets`. The calculations are subsetted on various patient characteristics. For estimating mean life length, an exponential survival model was assumed (the estimate is years per event). Continuous variables are categorized into quartiles automatically. Each quartile group is identified using the upper and lower endpoints within that quartile. The code for this example follows.

[5]For this purpose, the Hmisc `summarize` function may be more useful, if you don't want marginal statistics computed.

```
getHdata(pbc)     # getHdata is in Hmisc; downloads datasets from UVa web site

# Variables in pbc had units in ( ) inside variable labels.  Move
# these units of measurements to separate units attributes

pbc ← upData(pbc, moveUnits=TRUE)

attach(pbc)

# Example 1: For each variable level, estimate Kaplan-Meier 2 and 5-year
# survival probabilities and mean life length assuming an exponential
# distribution.  Note the 3 names given to the computations.  These
# will be used in column headings.

# Function to efficiently use Therneau's survfit.km to estimate
# survival at fixed time points for a single stratum.  Assumes S is a
# Surv object

kmsurv ← function(S, times) {
  f ← survfit.km(factor(rep(1,nrow(S))), S)
  tt ← c(0, f$time)
  ss ← c(1, f$surv)    # add first point to survival curve
  approx(tt, ss, xout=times, method='constant', f=0)$y
}

# Put probability estimates together with mean life length

describe.survival ← function(y) {
  km ← kmsurv(y, c(2,5))
  c('2 Year'=km[1], '5 Year'=km[2], 'Mean, y'=sum(y[,1])/sum(y[,2]))
}

library(survival)
S ← Surv(fu.days/365.25, status)

s1 ← summary(S ~ age + albumin + ascites + bili + drug + edema + chol,
              fun=describe.survival)

# Make 2 graphs: (1) survival probabilities  (2) mean life length
# First graph contains results from 2 calls to plot (add=T 2nd time)
```

```
for(w in 1:2) {
  if(w==1) setpdf(f1a,sublines=1,h=5.25) else
           setpdf(f1b,sublines=1,h=5)
  plot(s1, which=if(w==1)1:2 else 3,
       cex.labels=.7, subtitles=T, main='',
       pch=if(w==2) 16 else c('2','5'),        # 16=solid circle
       xlab=if(w==2)'Survival Time' else 'Survival Probability')
  dev.off()
}

w ← latex(s1, cdec=c(2,2,1), ctable=TRUE, caption='Survival')
# Creates s1.tex (Table 4)
```

This table is converted to two dot plots (Figures 1 and 2) using the `plot` method for an object created by `summary` with `method='response'` (see previous code). The Hmisc `setpdf` function is used to create the `pdf` graphics files. See Section 10 for the LaTeX code used to insert these graphics.

Table 5 is similar to Table 4 except that the Kaplan–Meier estimates are not shown, life length estimates are also stratified by treatment assigned (using the `stratify` function), and continuous variables are grouped into tertiles.

```
# Example 2: Stratify mean life length (only) by drug groups.  These will
# become major column groupings.  Use tertiles instead of quartiles.

life.expect ← function(y) c(Years=sum(y[,1])/sum(y[,2]))

s2 ← summary(S ∼ age + albumin + ascites + edema + stratify(drug),
              fun=life.expect, g=3)
# Note: You can summarize other response variables using the same independent
# variables using e.g. update(s2, response∼.), or you can change the
# list of independent variables using e.g. update(s2, response ∼.- ascites)
# or update(s2, .∼.-ascites)

setpdf(f2, h=4)
plot(s2, cex.labels=.6, xlim=c(0,40),   # Figure 3
     xlab='Mean Life Length', main='')
Key(-.56,.06)
dev.off()
```

Table 4: Survival N=418

	N	2 Year	5 Year	Mean, y
Age				
[26.3,43.0)	105	0.98	0.86	27.5
[43.0,51.2)	107	0.85	0.71	15.0
[51.2,58.3)	102	0.88	0.71	12.5
[58.3,78.4]	104	0.81	0.54	8.4
Albumin gm/dl				
[1.96,3.25)	105	0.72	0.45	5.8
[3.25,3.54)	105	0.88	0.64	11.9
[3.54,3.78)	104	0.97	0.85	24.5
[3.78,4.64]	104	0.95	0.85	22.5
ascites				
absent	288	0.94	0.76	16.3
present	24	0.38	0.17	2.3
Missing	106	0.84	0.68	13.4
Serum Bilirubin mg/dl				
[0.3, 0.9)	123	0.96	0.94	44.7
[0.9, 1.5)	95	0.95	0.86	24.4
[1.5, 3.5)	98	0.90	0.66	10.8
[3.5,28.0]	102	0.71	0.28	4.3
drug				
D-penicillamine	154	0.88	0.71	14.0
placebo	158	0.91	0.71	13.4
not randomized	106	0.84	0.68	13.4
edema				
no edema	354	0.93	0.77	17.1
edema, no diuretic therapy	44	0.73	0.47	6.9
edema despite diuretic therapy	20	0.30	0.08	1.8
Cholesterol mg/dl				
[120, 250)	71	0.80	0.71	13.9
[250, 310)	71	0.93	0.87	20.8
[310, 404)	72	0.92	0.73	15.2
[404,1775]	70	0.94	0.54	8.4
Missing	134	0.84	0.67	13.7
Overall				
	418	0.88	0.70	13.6

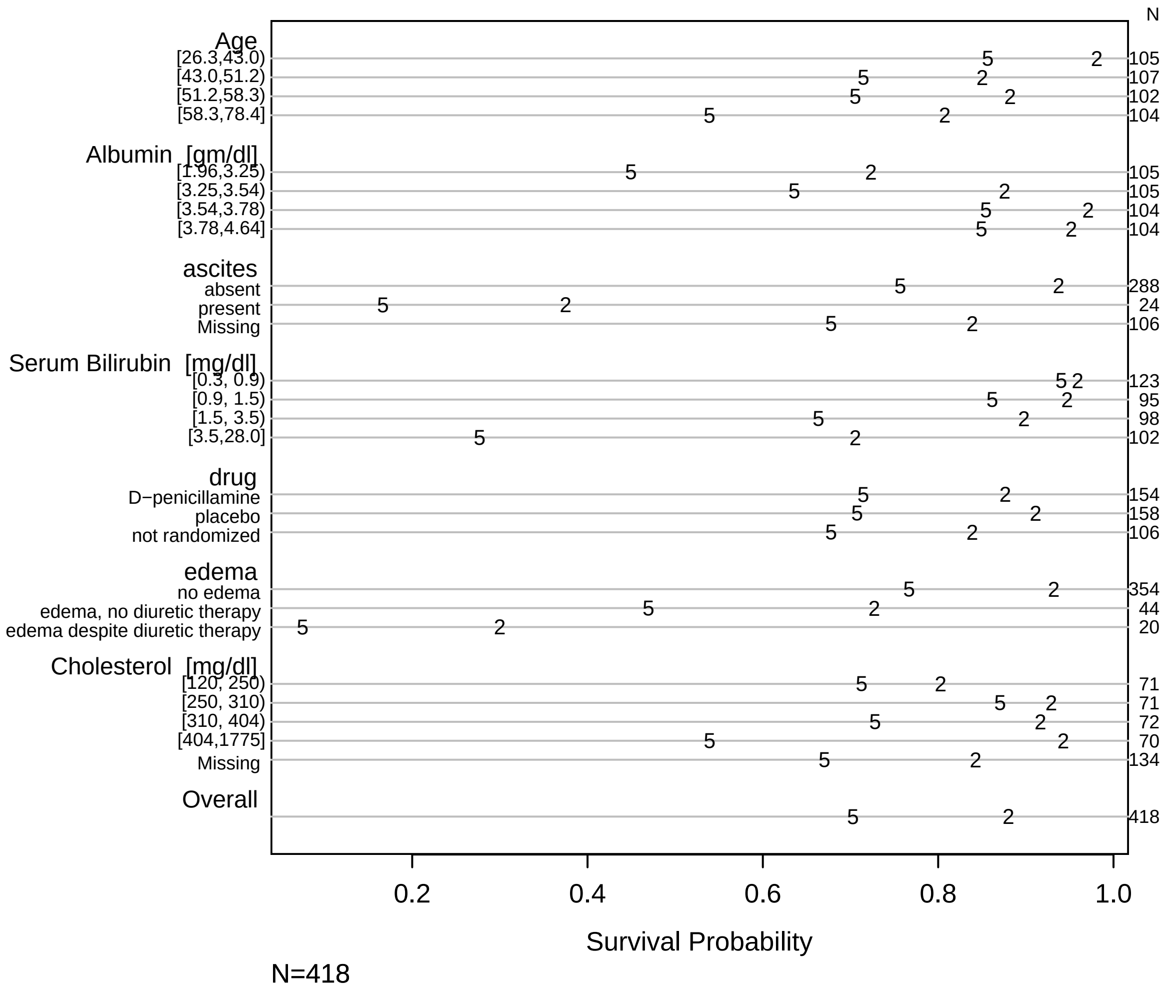

Figure 1: Two and five–year Kaplan–Meier survival probability estimates

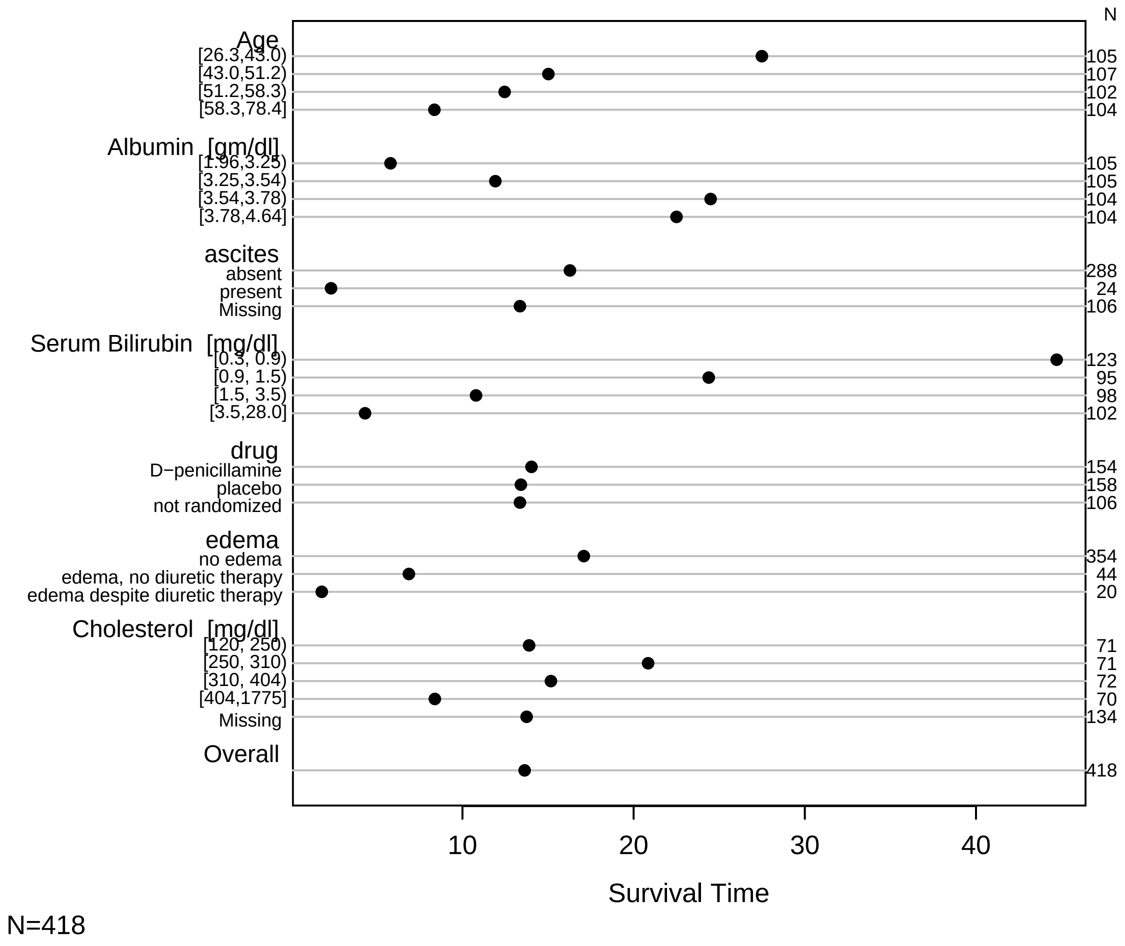

Figure 2: Estimated mean life length from an exponential survival model

Table 5: S by drug N=418

	D-penicillamine		placebo		not randomized	
	N	Years	N	Years	N	Years
Age						
[26.3,46.0)	65	24.1	50	18.6	25	60.7
[46.0,55.5)	48	13.1	51	11.7	40	10.8
[55.5,78.4]	41	7.9	57	12.0	41	10.3
Albumin gm/dl						
[1.96,3.36)	45	6.8	52	5.9	43	9.2
[3.36,3.69)	60	15.5	45	18.8	34	12.1
[3.69,4.64]	49	25.4	61	23.3	29	25.5
ascites						
absent	144	16.5	144	16.1	0	
present	10	1.6	14	2.9	0	
Missing	0		0		106	13.4
edema						
no edema	131	17.2	132	17.7	91	15.9
edema, no diuretic therapy	13	7.3	16	7.5	15	5.8
edema despite diuretic therapy	10	2.5	10	1.2	0	
Overall						
	154	14.0	158	13.4	106	13.4

```
w ← latex(s2, cdec=1, ctable=TRUE)
```

This table is converted to a dot plot in Figure 3.

Table 6 displays quartiles of cholesterol and bilirubin by various patient characteristics. To compute statistics simultaneously for cholesterol and bilirubin, we must use the S `cbind` function to create a bivariate response variable (a 2–column matrix). To compute quantiles for this new 2–variable entity we have to use the `apply` function instead of a simple invocation to `quantile`. For `age`, pre–specified intervals are used.

```
# Example 3: Take control of groups used for age.  Compute 3 quartiles for
# both cholesterol and bilirubin (excluding observations that are missing
# on EITHER ONE)
```

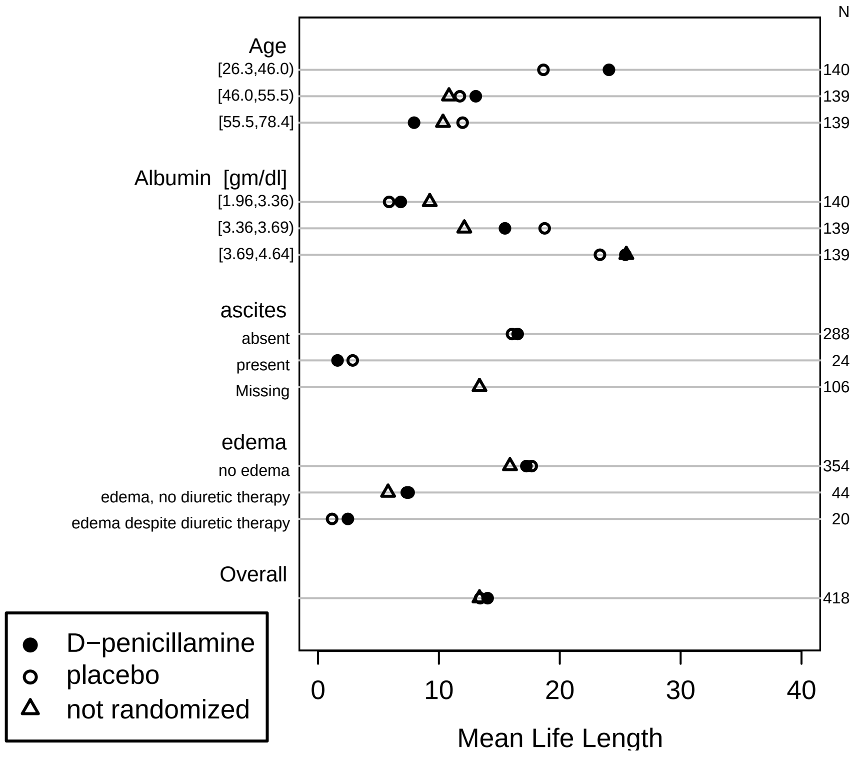

Figure 3: Estimated mean life length from an exponential survival model

```
age.groups ← cut2(age, c(45,60))

g ← function(y) apply(y, 2, quantile, c(.25,.5,.75))

y ← cbind(Chol=chol,Bili=bili)
# You can give new column names that are not legal S names
# by enclosing them in quotes, e.g. 'Chol (mg/dl)'=chol

vars ← c(label(chol), label(bili))
label(y) ← paste(vars, collapse=' and ')   # Will make nice caption in table
s3 ← summary(y ~ age.groups + ascites, fun=g)
s3

setpdf(f3, h=5, w=6.5)
par(mfrow=c(1,2), oma=c(3,0,3,0))   # allow outer margins for overall
for(ivar in 1:2) {                   # title
  isub ← (1:3)+(ivar-1)*3         # *3=number of quantiles/var.
  plot(s3, which=isub, main='', xlab=vars[ivar],
       pch=c(91,16,93))   # [, solid circle, ]
}
mtitle(paste('Quartiles of', label(y)))   # Overall (outer) title
dev.off()

w ← latex(s3, trios=vars, ctable=TRUE)  # trios → collapse 3 quartiles
```

Table 6 is shown as a graphic in Figure 4.

Tables 7 and 8 summarizes only bilirubin, but both the mean and median are printed. Separate tables are made for the two arms of the randomized study. For the active arm, the data are shown in Figure 5.

```
# Example 4: Summarize only bilirubin, but do it with two statistics:
# the mean and the median.  Make separate tables for the two randomized
# groups and make plots for the active arm.

g ← function(y) c(Mean=mean(y), Median=median(y))

for(sub in c("D-penicillamine", "placebo")) {
  s4 ← summary(bili ~ age.groups + ascites + chol, fun=g,
                subset=drug==sub)
```

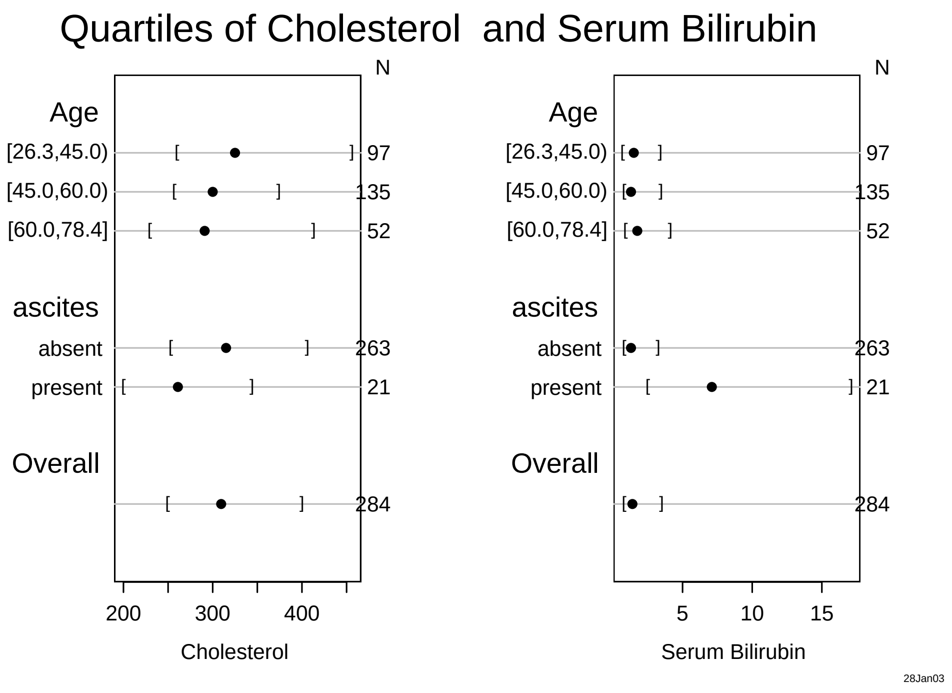

Figure 4: Quartiles of cholesterol and bilirubin

Table 6: Cholesterol and Serum Bilirubin N=284, 134 Missing

	N	Cholesterol	Serum Bilirubin
Age			
[26.3,45.0)	97	260.0 325.0 456	0.7 1.50 3.40
[45.0,60.0)	135	257.0 300.0 374	0.8 1.30 3.45
[60.0,78.4]	52	229.5 291.0 413	0.9 1.75 4.12
ascites			
absent	263	253.0 315.0 406	0.8 1.30 3.25
present	21	200.0 261.0 344	2.5 7.10 17.10
Overall			
	284	249.5 309.5 400	0.8 1.40 3.50

a *b* *c* represent the lower quartile *a*, the median *b*, and the upper quartile *c*.

```
  cat('\n',sub,'\n\n')
  print(s4)

  if(sub=='D-penicillamine') {
    setpdf(f4, h=4.5)
    plot(s4, which=1:2, pch=c(16,1), subtitles=F, main='', # 1=mean, 2=median
         xlab=label(bili))    # Figure 5
    dev.off()
  }

  w ← latex(s4, append=sub=='placebo', ctable=TRUE, size='scriptsize',
            label=if(sub=='placebo') 's4b' else 's4a',
            caption=paste(label(bili),' {\\em (',sub,')}', sep=''))
  # Note symbolic labels for tables for two subsets: s4a, s4b
}
```

Table 7: Serum Bilirubin *(D-penicillamine)* N=154

	N	Mean	Median
Age			
[26.3,45.0)	58	3.43	1.30
[45.0,60.0)	76	4.09	1.30
[60.0,78.4]	20	2.61	1.20
ascites			
absent	144	3.09	1.30
present	10	11.66	14.90
Cholesterol **mg/dl**			
[120, 255)	36	3.13	0.75
[255, 304)	36	1.54	0.85
[304, 383)	36	2.91	1.30
[383,1775]	36	6.96	4.05
Missing	10	3.89	1.25
Overall			
	154	3.65	1.30

Table 8: Serum Bilirubin *(placebo)* N=158

	N	Mean	Median
Age			
[26.3,45.0)	48	2.63	1.75
[45.0,60.0)	73	2.70	1.10
[60.0,78.4]	37	3.54	2.00
ascites			
absent	144	2.43	1.30
present	14	7.41	6.50
Cholesterol **mg/dl**			
[127, 248)	35	2.45	1.10
[248, 316)	35	1.55	1.10
[316, 420)	35	2.75	2.00
[420,1712]	35	4.89	3.20
Missing	18	2.59	1.15
Overall			
	158	2.87	1.40

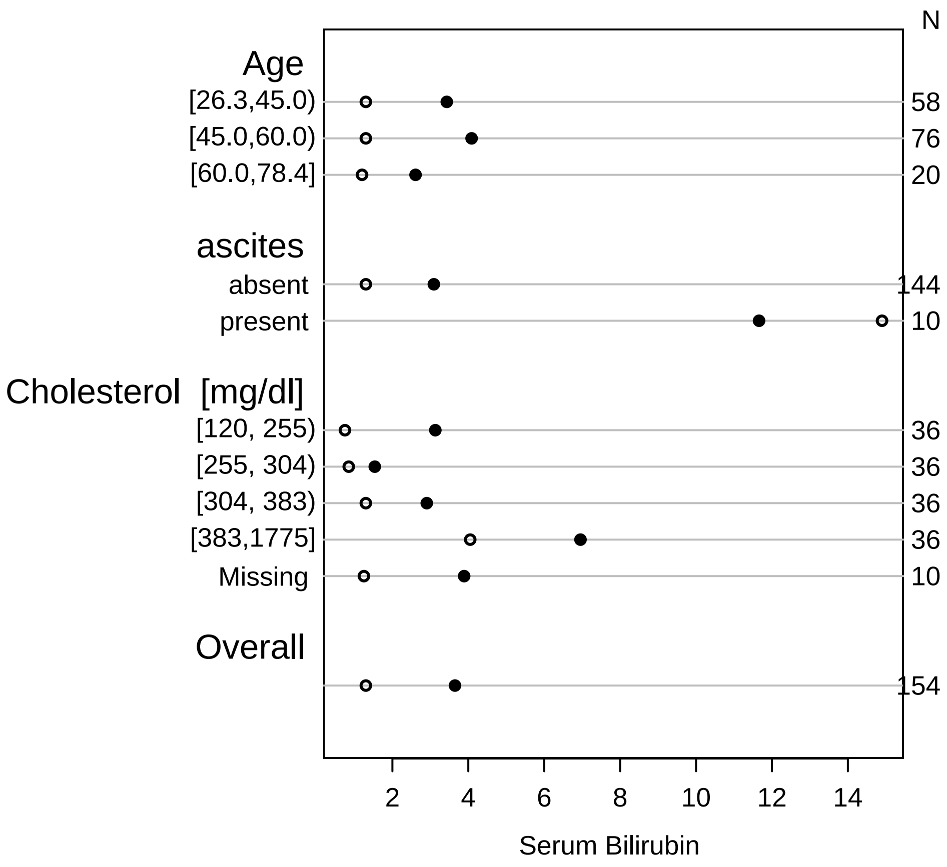

Figure 5: Mean (solid circle) and median (open circle) bilirubin for D–penicillamine patients

4.2 Baseline Characteristic Tables

Here the S `summary` function is used with the parameter `method='reverse'`, which reverses the role of the dependent variable and the independent variables. The dependent variable is assumed to be categorical; in clinical trials it will be the treatment assignment.

The next example again uses the primary biliary cirrhosis dataset. The result is in Table 9. It is printed in landscape mode using the LaTeX `lscape` package, and using the LaTeX `small` font. For `'reverse'`-type tables, an option `test=TRUE` will cause `summary.formula` to compute test statistics for testing across columns. Default tests are Wilcoxon or Kruskal-Wallis for continuous variables and Pearson χ^2 for categorical ones, but users may specify their own statistical tests[6].

```
# Now consider examples in 'reverse' format, where the lone dependent
# variable tells the summary function how to stratify all the 'independent'
# variables.  This is typically used to make tables comparing baseline
# variables by treatment group, for example.

label(stage) ← 'Histologic Stage\nLudwig Criteria'
# split into 2 lines
s5 ← summary(drug ∼ bili + albumin + stage + protime + sex + age + spiders,
             method='reverse', test=TRUE)
# To summarize all variables, use summary(drug ∼., data=pbc)

options(digits=1)
print(s5, npct='both')
# npct='both' : print both numerators and denominators

options(digits=3)
w ← latex(s5, npct='both', npct.size='normalsize',
          size='small', ctable=TRUE)
# creates s5.tex using normalsize font for numerator and
# denominator of percents
# Specify prtest='P' to just print P-values, prtest='stat' to just
# print test statistics
```

[6]In randomized trials, tests for baseline imbalance are unwarranted and difficult to interpret, in addition to causing multiple comparison problems (see Stephen Senn, *Statistical Issues in Drug Development*).

```
setpdf(f5a, h=7, pointsize=14)
plot(s5, which='categorical')    # Figure 6
Key(.72,.65)
dev.off()
setpdf(f5b, h=7, pointsize=16)
# Use box-percentile plot option
plot(s5, which='continuous', conType='bp')    # Figure 7
dev.off()
```

Table 9: Descriptive Statistics by drug

		N	D-penicillamine $N = 154$	placebo $N = 158$	not randomized $N = 106$	Test Statistic
Serum Bilirubin	mg/dl	418	0.725 1.300 3.600	0.800 1.400 3.200	0.725 1.400 3.075	$F_{2,415} = 0.03,\ P = 0.972^1$
Albumin	gm/dl	418	3.34 3.54 3.78	3.21 3.56 3.83	3.12 3.47 3.72	$F_{2,415} = 2.13,\ P = 0.12^1$
Histologic Stage Ludwig Criteria : 1		412	3% $\frac{4}{154}$	8% $\frac{12}{158}$	5% $\frac{5}{100}$	$\chi^2_6 = 5.33,\ P = 0.502^2$
2			21% $\frac{32}{154}$	22% $\frac{35}{158}$	25% $\frac{25}{100}$	
3			42% $\frac{64}{154}$	35% $\frac{56}{158}$	35% $\frac{35}{100}$	
4			35% $\frac{54}{154}$	35% $\frac{55}{158}$	35% $\frac{35}{100}$	
Prothrombin Time	sec.	416	10.0 10.6 11.4	10.0 10.6 11.0	10.1 10.6 11.0	$F_{2,413} = 0.23,\ P = 0.795^1$
sex : female		418	90% $\frac{139}{154}$	87% $\frac{137}{158}$	92% $\frac{98}{106}$	$\chi^2_2 = 2.38,\ P = 0.304^2$
Age		418	41.4 48.1 55.8	43.0 51.9 58.9	46.0 53.0 61.0	$F_{2,415} = 6.1,\ P = 0.00245^1$
spiders		312	29% $\frac{45}{154}$	28% $\frac{45}{158}$		$\chi^2_1 = 0.02,\ P = 0.885^2$

a b c represent the lower quartile *a*, the median *b*, and the upper quartile *c* for continuous variables. *N* is the number of non–missing values.
Tests used: [1]Kruskal-Wallis test; [2]Pearson test

To convert Table 9 to graphical form, `plot.summary.formula.reverse` constructs two pages. The first page contains statistics for all of the categorical variables, as all of these statistics are on the same scale (proportion or percent in each category). The second page contains a matrix of dot charts showing (by default) the 3 quartiles of each right–hand–side variable (on the x–axis), stratified by the left–hand variable (on the y–axis of each dot plot). The second set of plots is scaled to the most extreme 0.025 to 0.975 quantiles of the variable over all treatment groups. R can plot Greek letters, superscripts, subscripts, and mathematical operators, and Figure 6 and 7 take advantage of this capability. S-Plus does not have this capability, so simpler output would appear.

Table 10 presents a description of data from a trial for prostate cancer (from Byar and Green). The `prostate` data frame is available from `biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets`. The `overall` option is used to add a final column of statistics for the whole sample. The following listing contains code that produced all the tables and figures for the `prostate` data. This is a good application of the LaTeX `relsize` style. Specifying an overall size of the table of `smaller[3]` causes `latex()` to issue the command `\smaller[3]` at the start of the table and changes the overall table's font size to three levels below `normalsize`, which is LaTeX's `scriptsize`. Specifying `outer.size` and `Nsize` as `smaller` means to use one size smaller than this within the table, for 25th and 75th percentiles and for the sample sizes above the columns. One advantage of `relsize` is that if you use for example `{\smaller foo}` within a footnote, the next smaller size than is used for the overall footnoted text will be the size for `foo`.

```
# Consider another dataset
library(Hmisc)
getHdata(prostate)

# Variables in prostate had units in ( ) inside variable labels. Move
# these units of measurements to separate units attributes
# wt is an exception. It has ( ) in its label but this does not denote units
# Also make hg have a legal R plotmath expression

prostate ← upData(prostate, moveUnits=TRUE,
                  units=c(wt='', hg='g/100*ml'),
                  labels=c(wt='Weight Index = wt(kg)-ht(cm)+200'))
```

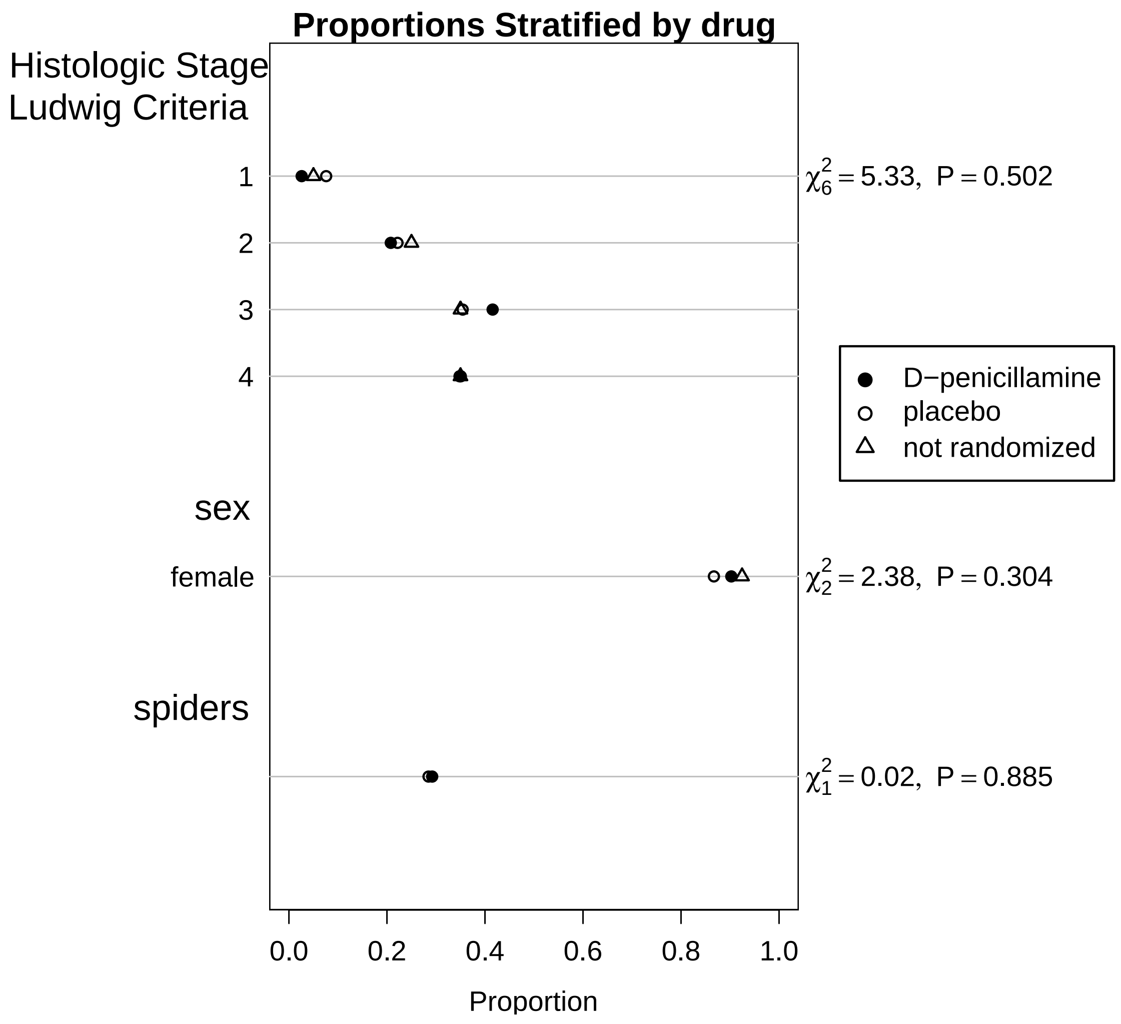

Figure 6: Proportions of patients in various categories of baseline variables, stratified by drug. Pearson χ^2 test results are given.

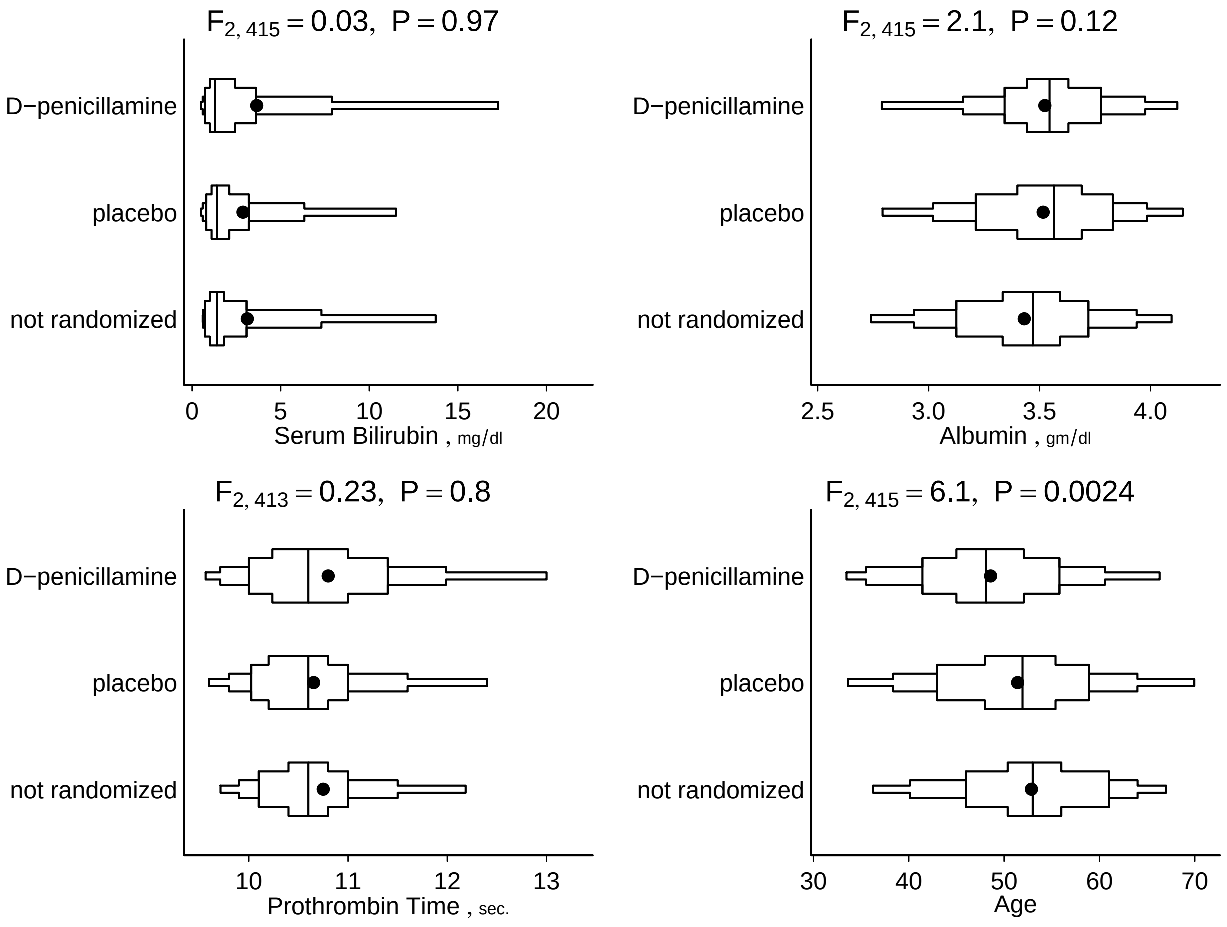

Figure 7: Box-percentile plots for continuous baseline variables in prostate cancer trial. 0.90, 0.75, 0.50, and 0.25 coverage intervals are shown. The solid circle depicts the mean and the vertical line the median. Kruskal-Wallis tests are also shown.

```
attach(prostate)

stage ← factor(stage, 3:4, c("Stage 3","Stage 4"))

s6 ← summary(stage ∼ rx + age + wt + pf + hx + sbp + dbp + ekg +
                      hg + sz + sg + ap + bm,
              method='reverse', overall=TRUE, test=TRUE)

options(digits=2)

w ← latex(s6, size='smaller[3]', outer.size='smaller', Nsize='smaller',
          long=TRUE, prmsd=TRUE, msdsize='smaller',
          middle.bold=TRUE, ctable=TRUE)    # Table 10

# smaller   :from relsize LaTeX style
# long=TRUE :put first category on a row by itself
# prmsd=TRUE:print means and S.D.

setpdf(f6a, h=7)
plot(s6, which='categorical', cex=.8)    # Figure 8
Key(-.02, 1)
dev.off()
setpdf(f6b, h=7)
plot(s6, which='continuous')    # Figure 9
dev.off()

# --------------------------------------------------------------------------
# Final examples use cross-classifications on possibly more than one
# independent variable.  The summary function with method='cross' produces
# a data frame containing the cross-classifications.  This data frame is
# suitable for multi-panel trellis displays.

bone ← factor(bm, 0:1, c("no mets","bone mets"))

s7 ← summary(ap>1 ∼ sz + bone, method='cross')

options(digits=3)
print(s7, twoway=F)
s7    # same as print(s7)
w ← latex(s7)    # Make s7.tex for Figure 11
```

```
library(lattice)                  # S-Plus: trellis (automatically attached)
setpdf(f7, h=6, w=6, trellis=T)  # Figure 10
Dotplot(sz ~ S | bone, data=s7,  # s7 is name of summary stats
        xlab="Fraction ap>1", ylab="Quartile of Tumor Size")
# Dotplot is Hmisc version of dotplot in lattice (S-Plus trellis)
dev.off()


summary(age ~ stage, method='cross')
summary(age ~ stage, fun=quantile, method='cross')
summary(age ~ stage, fun=function(x) c(Mean=mean(x), Median=median(x)),
        method='cross')
summary(cbind(age,ap) ~ stage + bone,
        fun=function(y) apply(y, 2, quantile, c(.25,.75)),
        method='cross')
options(digits=2)
summary(log(ap) ~ sz + bone,
        fun=function(y) c(Mean=mean(y), quantile(y)),
        method='cross')
```

Table 10: Descriptive Statistics by stage

	N	Stage 3 $N = 289$	Stage 4 $N = 213$	Combined $N = 502$	Test Statistic
rx	502				$\chi^2_3 = 0.22,\ P = 0.975^1$
placebo		26% (74)	25% (53)	25% (127)	
0.2 mg estrogen		25% (73)	24% (51)	25% (124)	
1.0 mg estrogen		25% (71)	26% (55)	25% (126)	
5.0 mg estrogen		25% (71)	25% (54)	25% (125)	
Age in Years	501	70.0 **73.0** 76.0 (71.8± 6.7)	69.0 **73.0** 76.0 (71.0± 7.6)	70.0 **73.0** 76.0 (71.5± 7.1)	$F_{1,499} = 0.2,\ P = 0.657^2$
Weight Index = wt(kg)-ht(cm)+200	500	91 **99** 109 (100± 13)	89 **97** 105 (97± 14)	90 **98** 107 (99± 13)	$F_{1,498} = 5.4,\ P = 0.021^2$
pf	502				$\chi^2_3 = 11,\ P = 0.0118^1$
normal activity		93% (268)	85% (182)	90% (450)	
in bed < 50% daytime		6% (18)	9% (19)	7% (37)	
in bed > 50% daytime		1% (3)	5% (10)	3% (13)	
confined to bed		0% (0)	1% (2)	0% (2)	
History of Cardiovascular Disease	502	46% (134)	37% (79)	42% (213)	$\chi^2_1 = 4.3,\ P = 0.0376^1$
Systolic Blood Pressure/10	502	13.0 **14.0** 16.0 (14.4± 2.6)	13.0 **14.0** 16.0 (14.3± 2.2)	13.0 **14.0** 16.0 (14.4± 2.4)	$F_{1,500} = 0.01,\ P = 0.907^2$
Diastolic Blood Pressure/10	502	7.0 **8.0** 9.0 (8.2±1.6)	7.0 **8.0** 9.0 (8.1±1.3)	7.0 **8.0** 9.0 (8.1±1.5)	$F_{1,500} = 0.43,\ P = 0.511^2$
ekg	494				$\chi^2_6 = 6.7,\ P = 0.347^1$
normal		35% (98)	33% (70)	34% (168)	
benign		5% (14)	4% (9)	5% (23)	
rhythmic disturb & electrolyte ch		8% (22)	14% (29)	10% (51)	
heart block or conduction def		6% (17)	4% (9)	5% (26)	
heart strain		30% (85)	31% (65)	30% (150)	
old MI		17% (47)	13% (28)	15% (75)	
recent MI		0% (1)	0% (0)	0% (1)	
Serum Hemoglobin g/100 ml	502	12.5 **13.8** 14.9 (13.7± 1.8)	11.8 **13.4** 14.6 (13.1± 2.1)	12.3 **13.7** 14.7 (13.4± 2.0)	$F_{1,500} = 11,\ P < 0.001^2$
Size of Primary Tumor cm^2	497	4 **8** 16 (12±11)	7 **17** 26 (18±13)	5 **11** 21 (15±12)	$F_{1,495} = 39,\ P < 0.001^2$
Combined Index of Stage and Hist. Grade	491	8.0 **9.0** 9.0 (9.1± 1.3)	11.0 **12.0** 13.0 (12.0± 1.5)	9.0 **10.0** 11.0 (10.3± 2.0)	$F_{1,489} = 605,\ P < 0.001^2$
Serum Prostatic Acid Phosphatase	502	0.40 **0.50** 0.70 (0.66± 1.75)	1.60 **4.20** 20.00 (27.80±93.29)	0.50 **0.70** 2.97 (12.18±62.17)	$F_{1,500} = 802,\ P < 0.001^2$
Bone Metastases	502	0% (1)	38% (81)	16% (82)	$\chi^2_1 = 127,\ P < 0.001^1$

a b c represent the lower quartile *a*, the median *b*, and the upper quartile *c* for continuous variables. $x \pm s$ represents $\bar{X} \pm 1$ SD. N is the number of non–missing values. Numbers after percents are frequencies. Tests used: [1]Pearson test; [2]Wilcoxon test

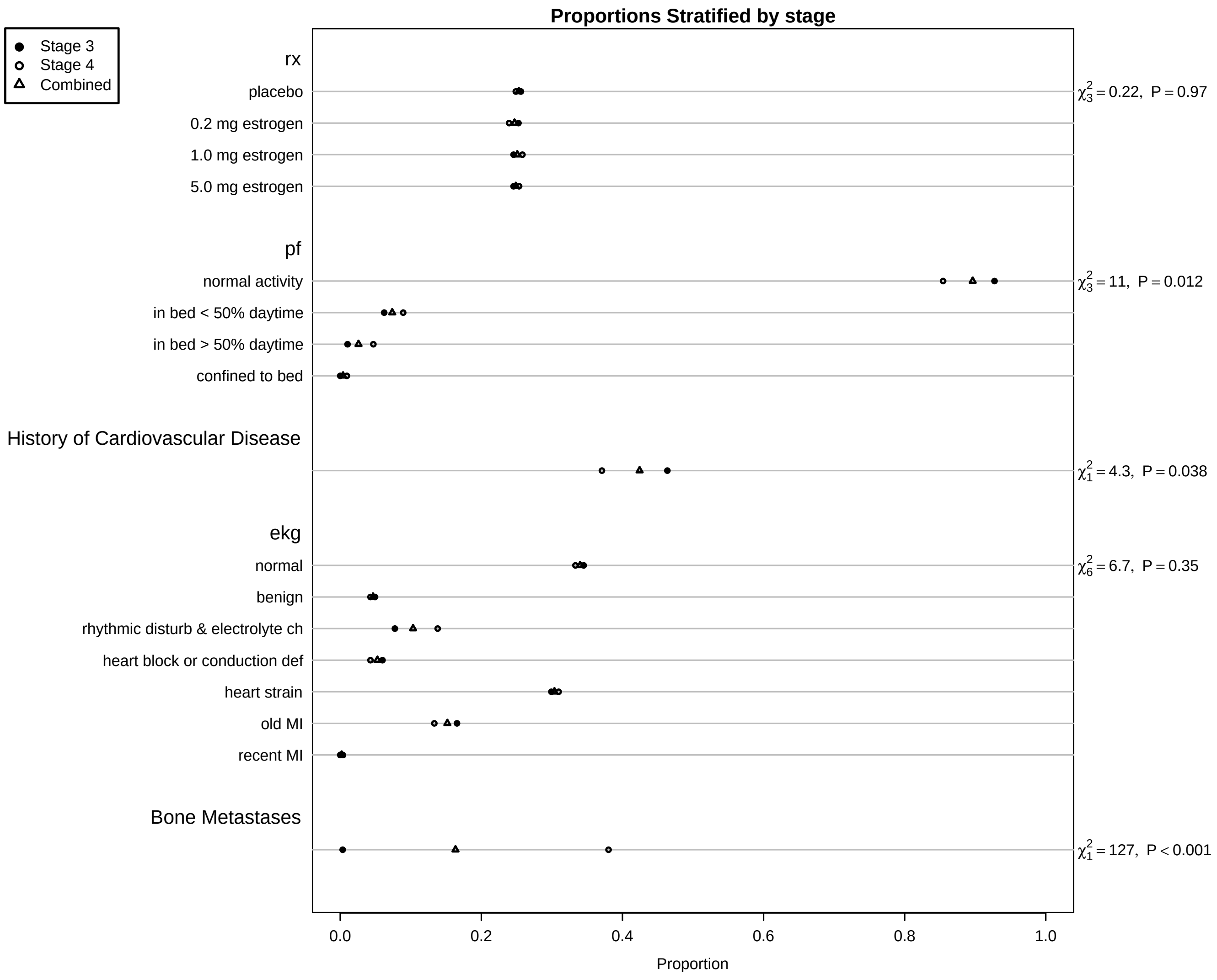

Figure 8: Distribution of categorical baseline variables in prostate cancer trial

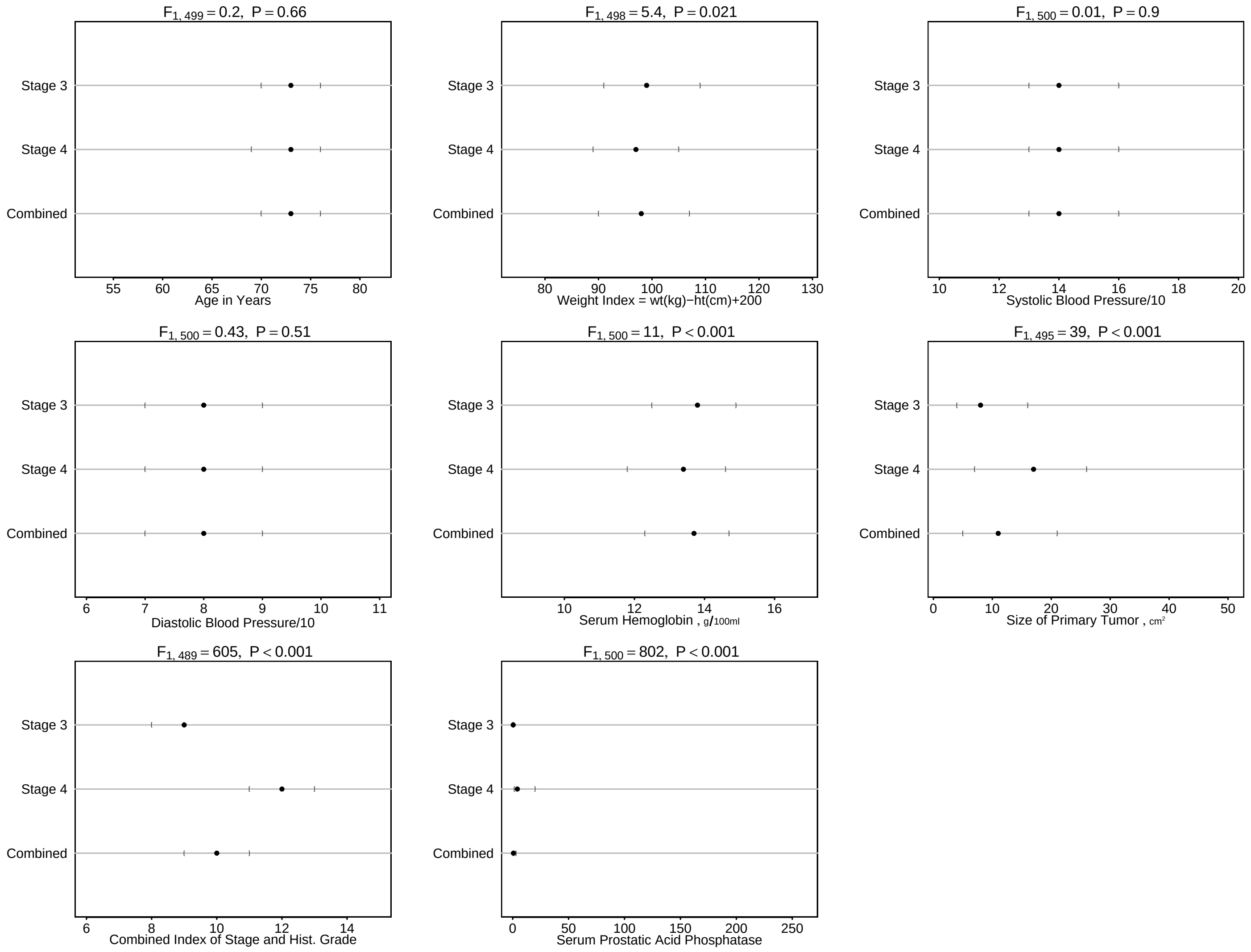

Figure 9: Quartiles of continuous variables in prostate cancer trial. x–axes are scaled to the lowest 0.025 and highest 0.975 quantiles over all groups for each variable.

4.3 Data Displays from Cross–Classifying Variables

The final examples use cross–classification on possibly more than one independent variable. The summary function with `method='cross'` produces a data frame containing the cross–classifications. This data frame is suitable for multi-panel trellis displays although if marginal statistics are not needed, the Hmisc `summarize` function is better. The first example in this series was LaTeX'ed to create Table 11 (the code is listed above).

Table 11: Fraction of ap > 1 by sz, bone

Size of Primary Tumor (cm^2)	**no mets**		**bone mets**		**Total**	
	N	ap > 1	N	ap > 1	N	ap > 1
$[0, 5)$	105	0.248	5	0.8	110	0.273
$[5, 11)$	119	0.21	17	0.765	136	0.279
$[11, 21)$	103	0.301	19	0.947	122	0.402
$[21, 69]$	88	0.5	41	0.902	129	0.628
Missing	5	0.4	0		5	0.4
Total	420	0.305	82	0.878	502	0.398

There is no `plot` method for `method='cross'` tables, but you can use Trellis graphics on the data frame that is created by `summary` (see code above). For this purpose, the `Hmisc` `summarize` function might be better than `summary.formula` for producing the needed aggregated data.

5 Handling Special Variables

5.1 Multiple Choice Variables

Clinical reports frequently must summarize "checklist" or multiple–choice variables. Such variables are typically listed on a case report form using one of two methods:

1. Specify up to three primary presenting symptoms:

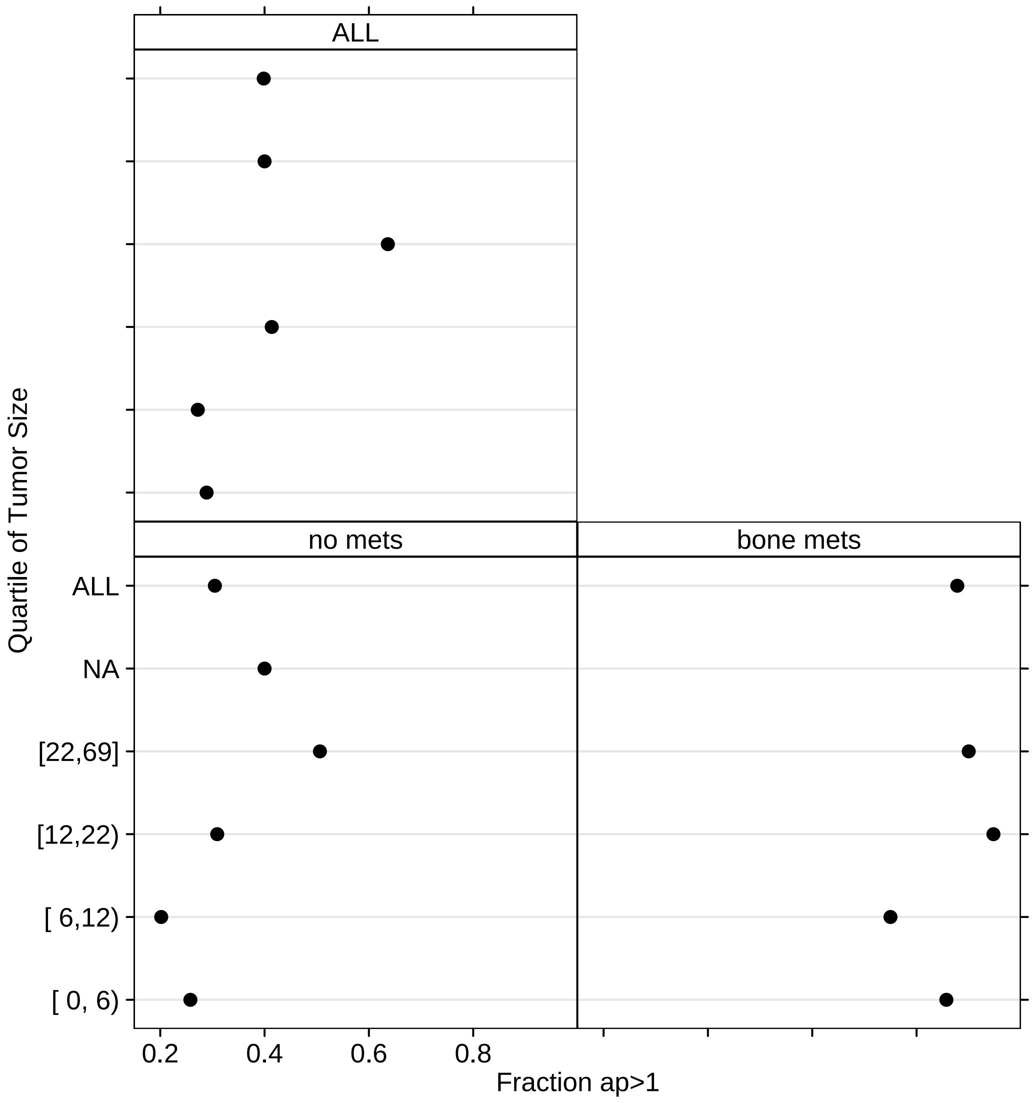

Figure 10: Proportion of patients with acid phosphatase exceeding 1.0, cross–classified by tumor size and bone metastasis

```
---------  -------- --------
```

Here the respondent writes in up to three symptom codes from a list of perhaps 15 integer codes defined below the question.

2. Check symptoms that are present:

```
headache   __   stomach ache __   hangnail __
back pain  __   neck ache    __   wheezing __
```

When such data are processed, either a series of three categorical variables or 6 binary variables is created. In what follows we assume that the binary variables are coded as numeric 0/1 or as character variables with values (ignoring case) of `'yes'` and `'present'` denoting a positive response. In composing a report, we usually want to consider all of these component variables under the umbrella of `'Presenting Symptoms'`. If using presenting symptoms as stratification (independent) variables, we will want to know an outcome statistic computed separately for those subjects having headache, those having stomach ache, etc. These categories will overlap for some subjects. When summarizing presenting symptoms stratified by treatment, we will want to know the proportion of subjects in each treatment group having headache, the proportion having stomach ache, etc., with the proportions summing to > 1.0 if any subject had more than one symptom.

The Hmisc `summary.formula` function can handle multiple choice / checklist variables after they are combined into a matrix. The Hmisc `mChoice` function will take as input a series of categorical vector variables (using the first input format above), and make a matrix with the number of columns equal to the number of choices that were actually selected in the data[7]. This new matrix consists of logical `T/F` values. You can also give `summary.formula` a matrix you create, if using input format two above. The elements of this matrix need to be numeric with values 0 and 1, logical `F/T`, or character with values (ignoring case) of `'yes'` or `'present'`.

Here is an example of the use of `mChoice` from its help file.

```
> options(digits=3)
> set.seed(173)
```

[7]There is also an option to create a column for `'none'` for subjects for whom no choices were selected. The input variables need not have the same levels. A master list of categories is constructed by finding all unique categories in the levels of all variables combined, preserving the order of levels for the factor variables.

```
> sex ← factor(sample(c("m","f"), 500, rep=T))
> age ← rnorm(500, 50, 5)
> treatment ← factor(sample(c("Drug","Placebo"), 500, rep=T))

> # Generate a 3-choice variable; each of 3 variables has 5 possible levels
> symp ← c('Headache','Stomach Ache','Hangnail',
+           'Muscle Ache','Depressed')
> symptom1 ← sample(symp, 500, T)
> symptom2 ← sample(symp, 500, T)
> symptom3 ← sample(symp, 500, T)
> Symptoms ← mChoice(symptom1, symptom2, symptom3, label='Primary Symptoms')

> # Note: In this example, some subjects have the same symptom checked
> # multiple times; in practice these redundant selections would be NAs
> # mChoice will ignore these redundant selections
> # If the multiple choices to a single survey question were already
> # stored as a series of T/F yes/no present/absent questions we could do:
> # Symptoms <- cbind(headache,stomach.ache,hangnail,muscle.ache,depressed)
> # where the 5 input variables are all of the same type: 0/1,logical,char.
> # These variables cannot be factors in this case as cbind would
> # store integer codes instead of character strings.
> # To give better column names can use
> # cbind(Headache=headache, 'Stomach Ache'=stomach.ache, ...)

> # Following 8 commands only for checking mChoice
> data.frame(symptom1,symptom2,symptom3)[1:10,]

       symptom1     symptom2     symptom3
 1     Headache Stomach Ache     Headache
 2    Depressed  Muscle Ache    Depressed
 3 Stomach Ache  Muscle Ache Stomach Ache
 4     Hangnail  Muscle Ache     Headache
 5  Muscle Ache     Headache    Depressed
 6     Headache     Headache     Headache
 7 Stomach Ache Stomach Ache  Muscle Ache
 8  Muscle Ache     Headache    Depressed
 9     Hangnail     Hangnail     Hangnail
10    Depressed  Muscle Ache    Depressed

> Symptoms[1:10,]  # Print first 10 subjects' new binary indicators
```

```
Primary Symptoms
      Depressed Hangnail Headache Muscle Ache Stomach Ache
 [1,]         F        F        T           F            T
 [2,]         T        F        F           T            F
 [3,]         F        F        F           T            T
 [4,]         F        T        T           T            F
 [5,]         T        F        T           T            F
 [6,]         F        F        T           F            F
 [7,]         F        F        F           T            T
 [8,]         T        F        T           T            F
 [9,]         F        T        F           F            F
[10,]         T        F        F           T            F

> meanage ← single(5)
> for(j in 1:5) meanage[j] ← mean(age[Symptoms[,j]])
> names(meanage) ← dimnames(Symptoms)[[2]]
> meanage

 Depressed Hangnail Headache Muscle Ache Stomach Ache
      49.9     49.8     49.9        50.3         49.8
> # Manually compute mean age for 2 symptoms
> mean(age[symptom1=='Headache' | symptom2=='Headache' | symptom3=='Headache'])
[1] 49.9
> mean(age[symptom1=='Hangnail' | symptom2=='Hangnail' | symptom3=='Hangnail'])
[1] 49.8

> #Frequency table sex*treatment, sex*Symptoms
> summary(sex ∼ treatment + Symptoms, fun=table)
> # could also do summary(sex ∼ treatment + mChoice(symptom1,...),...)

sex    N=500

----------------+------------+---+---+---+
                |            |N  |f  |m  |
----------------+------------+---+---+---+
treatment       |Drug        |246|123|123|
                |Placebo     |254|129|125|
----------------+------------+---+---+---+
Primary Symptoms|Depressed   |242|130|112|
                |Hangnail    |238|125|113|
                |Headache    |236|110|126|
                |Muscle Ache |255|127|128|
                |Stomach Ache|252|125|127|
----------------+------------+---+---+---+
Overall         |            |500|252|248|
----------------+------------+---+---+---+

> #Compute mean age, separately by 3 variables
> summary(age ~ sex + treatment + Symptoms)

age    N=500

----------------+------------+---+----+
                |            |N  |age |
----------------+------------+---+----+
sex             |f           |252|49.8|
                |m           |248|49.9|
----------------+------------+---+----+
treatment       |Drug        |246|49.7|
                |Placebo     |254|50.0|
----------------+------------+---+----+
Primary Symptoms|Depressed   |242|49.9|
                |Hangnail    |238|49.8|
                |Headache    |236|49.9|
                |Muscle Ache |255|50.3|
                |Stomach Ache|252|49.8|
----------------+------------+---+----+
Overall         |            |500|49.9|
----------------+------------+---+----+

> f ← summary(treatment ~ age + sex + Symptoms, method="reverse")

Descriptive Statistics by treatment

----------------------------+--------------+--------------+
                            |Drug          |Placebo       |
                            |(N=246)       |(N=254)       |
----------------------------+--------------+--------------+
                         age|46.5/49.8/52.5|46.4/50.1/53.4|
-----------------------------+--------------+--------------+
                      sex : m|   50% (123)  |   49% (125)  |
-----------------------------+--------------+--------------+
Primary Symptoms : Depressed|   50% (122)  |   47% (120)  |
-----------------------------+--------------+--------------+
                     Hangnail|   47% (116)  |   48% (122)  |
-----------------------------+--------------+--------------+
                     Headache|   45% (110)  |   50% (126)  |
-----------------------------+--------------+--------------+
                  Muscle Ache|   48% (117)  |   54% (138)  |
-----------------------------+--------------+--------------+
                 Stomach Ache|   53% (130)  |   48% (122)  |
-----------------------------+--------------+--------------+
```

5.2 Conditionally Defined Variables

Another type of variable that is common in clinical reports is a variable that is of no interest unless another variable equalled a certain value. A common example is cause of death. We may want our report to contain the proportion of patients dying on each treatment, and for the deaths, we may want to know the proportions of deaths due to each cause. For the latter calculation, the denominator is not the number of subjects in a treatment but rather the number of subjects who died on that treatment. `summary.formula` will handle such variables correctly as long as they have missing values when they are not pertinent. For example, suppose that the variable `death.cause` is `NA` if `death` is `F` (false) and `death.cause` is a categorical (or `mChoice`) variable if `death` is `T`. Then a `'reverse'` type summary will produce the needed proportions of `death` as well as `death.cause`.

6 Alternate Approaches

6.1 Literate Programming

In *literate programming* as used in reproducible research (see `biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatReport`), a single source document contains analysis code as well as text for the report. This has been found to be easier to maintain and to result in better documentation. Under R, the `Sweave` package provides a concise syntax for mixing S and LaTeX code for producing reports, as discussed in Section 16.3 of the course notes at biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse. `Sweave` will run the S code chunks through R, include S printed output in the report, and will generate LaTeX commands to automatically include graphics generated by the S code. One especially nice feature of `Sweave` is the ease with which users can insert variables computed by S into LaTeX text without the need of the `\def\varname{value}` approach described earlier.

`Sweave` is particularly well suited for non-recurring statistical reports. Reports that are run after periodic data updates, for which the time spent polishing the report is well spent, are sometimes better suited to the customized programming methods described earlier in this document.

6.2 LaTeX Server

The UVa Biostatistics LaTeX server allows the user to upload S output that contains a mixture of S commands and printed output and to upload a `.zip` file containing all the postscript graphics files for the report, and will run LaTeX on the server, automatically including graphics and making it easy for the user to provide legends for the plots. The user can then download a `.pdf` document containing the typeset report. See Chapters 2, 6, and 11 of the course notes at biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse for more information.

7 Data Preparation

For making nice–looking tables, as well as for having self–documenting variables, it is important to spend time defining good variable and value labels. If you are managing the data in SAS, for example, specify nice variable labels in a DATA step or using PROC DATASETS, and specify pretty value labels using PROC FORMAT. Both variable and value labels should use letter cases carefully. Don't use all upper case for either kinds of labels. Variable labels should often contain units of measurements. An example of a good label is `'Serum Cholesterol, mg/dl'`. Better still, separate the `'units'` attribute from the `'label'` attribute of a variable:

```
label(chol) ← 'Serum Cholesterol'
units(chol) ← 'mg/dl'
# Alternate approach:
mydata ← upData(mydata, labels=c(chol='Serum Cholesterol'),
                        units =c(chol='mg/dl'))
```

Some of the `latex` and `plot` methods in the `Hmisc` and `Design` libraries make special use of `units` attributes by typesetting them in a different font or by right-justifying units in cells of LaTeX tables.

Binary variables are often coded 0/1. Good variable labels for these are of the form `'Nocturnal angina present'`. Sometimes you may want printouts to be more self–documenting. Then consider defining a SAS format of the form `0='Angina absent' 1='Angina present'`.

You can always change labels and value labels after data are imported into S. Here are some examples.

```
  label(age) ← 'Age (y)'
  levels(pain) ← c('None','Mild','Moderate','Severe')
  levels(pain) ← list('Moderate/Severe'=c('Moderate','Severe'))
  #Combines last two levels for subgroup analyses in which
  #there were two few patients with severe pain

  levels(symptom)[3] ← 'Night sweats'   # fix one level

  #Give fuller labels to levels of a binary variable
  nangina ← factor(nangina, 0:1, c('Absent','Present'))
```

The Hmisc `upData` function provides a more general approach for changing variable attributes. See Section 4.1.5 of Alzola and Harrell.

The Hmisc `sas.get` function is used to translate SAS data to an S data frame, carrying all data attributes. There are options to handle special missing values. A typical procedure is to make an S program called `create.s` for each project directory. This program is run only whenever the SAS data changes. The create program should run the Hmisc `describe` function (and possibly the `hist.data.frame` or `datadensity` function) to check each variable being analyzed for valid values and to make sure that key data are seldom missing. Here is a typical `create.s`:

```
  rct ← sas.get('/my/data/path', 'rct', format.library='/my/formats',
                var=Cs(age,sex,treatment,dtime,death,pressure),
                uncompress=T)  #automatically uncompresses .ssd01 files
  #Cs() quotes all names (doesn't work if SAS names contain underscores)

  describe(rct)
```

If you run S interactively to develop and debug your reporting programs, you will find it handy to make a pop–up window showing variable names, labels, and value levels. To do this, issue the command `contents(rct)` after getting access to the `Hmisc` library, where `rct` is the name of your randomized trial data frame. To pop–up a more detailed window with distributions for each variable, use for example `page(contents(rct), multi=T)` (in S-Plus). There is also an `html` method for the results of `contents`, to allow you to view metadata in a browser (with hyperlinks between variables and value labels). See `biostat.mc.vanderbilt.edu/twiki/pub/Main/DataSets/Cpbc.html` for example HTML output from `contents()`.

If you want to make variable label or value label changes in S permanent, one option is to add the following type of statements after the `sas.get` command above.

```
  attach(rct, pos=1, use.names=F)
  label(trt) ← 'Treatment'
  sex ← factor(sex, c('f','m'), c('Female','Male'))
  xx ← factor(xx, c('a','b'), c('A label','B label'), exclude='Unknown')
  # Treat 'Unknown' as a missing value instead of a level
  ...
  detach(1, 'rct')
```

A safer approach follows.

```
rct ← upData(rct,
              labels=c(trt='Treatment'),
              sex=factor(sex,c('f','m'),c('Female','Male')),
              xx =factor(xx, c('a','b'), c('A label','B label'),
                         exclude='Unknown'))
```

See the Alzola and Harrell online text for much more information about modifying and recoding variables and reshaping data.

The Hmisc function `Label` will generate S assignment statements containing all `label`s for variables in a specified data frame. You can edit the file output by `Label` to easily modify labels you don't like. Look at the help file for `label` for more information.

If you run `summary` output through `latex()`, caret signs in variable labels and sometimes in value labels will cause the word after the caret (up to the next space, comma, or end of string) to be superscripted. Also, the symbols `< <= > >=` will be translated to the proper math–mode symbols such as $\geq$. There are other cases in which you may want to embed LaTeX codes inside labels, e.g.:

```
label(x2) ← '$X_2$'
```

which results in `x2` being typeset as X_2.

8 Inserting LaTeX Output into non–LaTeX Applications

You can use LaTeX to create tables and other text or graphics and convert the output file to encapsulated postscript (EPS) for insertion into Word or Wordperfect "pictures". These pictures will not be viewable on the screen (a blank box with be displayed) but they will print correctly as long as you remember to set your printer to a postscript printer before actually printing. Once you import the picture you can re–size it (if you use a 300 dpi postscript driver, making the image larger will result in fuzzy printing).

Use the `dvips` program to make an EPS file from a LaTeX dvi file, using the `E` option. Here is an example for the simple case in which the document is only one page long (e.g., it consists of a single table).

```
dvips -E -o doc.eps doc    # creates doc.eps from doc.dvi
```

If you have a multiple–page LaTeX document, you can tell `dvips` which page to store in a separate EPS file, for example, page 9:

```
dvips -E -p 9 -l 9 -o nine.eps doc
```

You can even have `dvips` put every page of the document into a separate file. The files will be numbered e.g. `doc.001, doc.002, doc.003, ...`:

```
dvips -E -S 1 -i -o doc.0 doc
```

Note that S plots are already in EPS, so you can include them in any document with no extra steps, as long as you stored only one plot in the EPS file. A nice way to pick out individual plots and store them in a separate `.ps` file is to use a postscript utility program called `psselect`, e.g. if you created 3 pages of plots in `myplots.ps` use

```
psselect -p1 myplots.ps myplots1.ps
```

to put the first page of `myplots.ps` into `myplots1.ps`. `psselect` can also be used to split out desired pages from a postscripted version of a LaTeX document as an alternative to using the page number or section splitting options to `dvips`. Michael Stevens of Duke University has written a program called `oneperpg` which will go through a multiple–page postscript file and automatically create separate files each containing one page of output, using `psselect`. For example, typing

```
oneperpg myplots
```

creates `myplots1.ps, myplots2.ps, myplots3.ps`.

Another way, at least for UNIX or Linux users, to use LaTeX output in other applications, is to run the `latex2html` program to convert the .tex files into an .html file. This was done with Table 6 by running the following code (`test.tex`) through `latex2html` using the system command `latex2html test`:

```
\documentclass{article}
\begin{document}
\input{s3}
\end{document}
```

The result can be found in `hesweb1.med.virginia.edu/biostat/s/doc/s3.html`.

You can insert the HTML file into Microsoft Word 97 documents, but if you save the document as a Word file rather than as HTML, special formatting such as LaTeX font size changes will be lost. This is because Microsoft is not consistent in how enhanced HTML commands are implemented in Internet Explorer and in Word. In addition to this problem, `latex2html` does not convert all table commands properly; sometimes the program just stops in the middle of the conversion. If you have any math commands in the document, `latex2html` has to convert these to GIF images. See `www.tex2html.com` for more information.

In general, HeVeA (`http://pauillac.inria.fr/~maranget/hevea/`) does an excellent job in converting LaTeX code to HTML, without the need for graphics images for math commands. For some applications the resulting HTML can easily be inserted into Word documents.

9 S Documention

Use the command `?summary.formula` under S to get detailed document of `summary.formula` and its `print`, `plot`, and `latex` methods.

10 LaTeX Code for This Document

```
% Usage: pdflatex --shell-escape summary   --shell-escape enables sinput

\documentclass[11pt]{article}

%     Style           Purpose
%     ------------    -------------------------------------------------
%     graphicx        LaTeX graphics package with rotation etc.
%     ctable          Nice tables with bolder initial horizontal line
%     moreverb        Inclusion of text files (verbatimtabinput)
%     fancyhdr        Headers, footers (rhead)
%     lscape          Landscape model (landscape)
%     sinput          Inclusion of S code with automatic reformatting
%     hyperref        Hyper--referencing for electronic documents (pdf)
%     url             Split long URLs (part of hyperref)
%     relsize         Specify font sizes as relative to current normalsize
```

```
\usepackage{graphicx}
\usepackage{ctable}
\usepackage{moreverb}
\usepackage{fancyhdr}
\usepackage{lscape}
\usepackage{sinput}
\usepackage{relsize}

\newcommand{\splus}{{S-P\sc{lus}}}
\newcommand{\R}{{\normalfont\textsf{R}}{}}
\newcommand{\scom}[1]{{\rm\scriptsize \# #1}}
% defines how sinput prints S comments
\newcommand{\code}[1]{\texttt{\smaller #1}}  % format software names
% smaller implemented by relsize: use 1 size smaller than current font

%\newcommand{\titl}{Statistical Tables and Plots using S and LaTeX}
\usepackage[pdftex,bookmarks,pagebackref,colorlinks,pdfpagemode=UseOutlines,
    pdfauthor={Frank E Harrell Jr},
    pdftitle={Statistical Tables and Plots using S and LaTeX}]{hyperref}

% Macros to start and end in-line S code listings (assumes sinput in effect)
\newcommand{\bex}{
 \begin{list}{}{\setlength{\leftmargin}{\parindent}}%
 \item%
 \begin{alltt}%
}
\newcommand{\eex}{
 \end{alltt}%
 \end{list}%
}


% The following macro makes insertion of pdf figures easy.
% Usage: \fig{label=.pdf prefix}{caption}{short caption for list of
% figures}{scalefactor}
\newcommand{\fig}[4]{\begin{figure}[hbp!]
 \leavevmode\centerline{\includegraphics[scale=#4]{#1.pdf}}
 \caption[#3]{\small #2}
 \label{#1}
 \end{figure}}

% The following macro makes insertion of pdf figures easy.  Unfortunately,
% inserting .pdf figures results in wasted space before and after the
% graph.  This can be fixed by providing vspace commands with negative
% heights.
```

```
%\pdfig{label=.pdf prefix}{caption}{short caption}{scalefactor}{leftshift}
%{vspace before figure (try -2in)}{vspace before caption(try -2in)}
\newcommand{\pdfig}[7]{\begin{figure}[htbp]\begin{center}
        \vspace{#6}\scalebox{#4}{\hspace{#5}\includegraphics{#1.pdf}}
        \vspace{#7}
        \caption[#3]{\small #2}
        \label{#1}
        \end{center}\end{figure}}


\setlength{\parindent}{0ex}    % don't indent first line of paragraph
\setlength{\parskip}{2ex}      % do skip 2 spaces between paragraphs

\pagestyle{fancy}              % used for running headers, footers (rhead)
\renewcommand{\subsectionmark}[1]{} % suppress subsection titles in headers

\begin{document}

\title{Statistical Tables and Plots using S and \LaTeX}

\author{FE Harrell \\
        Department of Biostatistics \\
        Vanderbilt University School of Medicine \\
\href{mailto:f.harrell@vanderbilt.edu}{\url{f.harrell@vanderbilt.edu}} \\ ~ \\
\href{http://biostat.mc.vanderbilt.edu}
{\url{biostat.mc.vanderbilt.edu}}\footnote{Document Address:
\href{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/StatReport/summary.pdf}
{\url{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/StatReport/summary.pdf}}.  This
document was produced using Te\TeX\ on RedHat 8.0 Linux using \R\
version 1.6.1 and version 1.5-0 (31Jan03) of the \code{Hmisc}
library.  All commands and output will be the same for \splus\ except
that Greek letters, superscripts, and subscripts will not appear in
plots.}
}
\date{\today}
\maketitle

\tableofcontents
\listoftables
\listoffigures

\section{Introduction to \LaTeX}
\LaTeX\ is a public domain document processing system developed by
Lamport (which uses \TeX\ by Knuth) that is used heavily in the
sciences and by journal and book publishers\footnote{\LaTeX\ is
available on many platforms.  Excellent free versions for Microsoft
Windows are FP\TeX\ by Fabrice Popineau and Mik\TeX, both available at
\href{http://www.ctan.org}{\url{www.ctan.org}}.
An excellent free book on \LaTeX\ is available at
\href{http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf}
{\url{ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf}}}.
\LaTeX\ is a {\em markup language} that is compiled similar to
programming languages such as C.  \LaTeX\ is particularly strong in
layouts, cross--referencing, typesetting equations, making tables,
bibliographic citations, indexes and tables of contents, and allowing
for insertion of graphics in documents.  This makes \LaTeX\ very
suitable for compiling long statistical reports such as those used to
support drug licensing.  For this purpose, major advantages of \LaTeX\
include the ability to automatically create cross--references and to
automatically update a report if any of its component graphics figures
or tables changes.  To accomplish the latter capability, the analyst
merely re--runs the statistical program that produced the graphics or
table components.  These graphics and tables are read respectively by
\LaTeX\ by an \verb|\includegraphics{}| or \verb|\input{}| command, so
running the \code{latex} command to recompile to report will make any
needed updates.  This is in distinction to Microsoft Word, which does
not have a batch inclusion capability.

Everything in a \LaTeX\ source document is plain text, so
you can edit these documents using any text editor\footnote{The
  \code{Emacs} editor has a special mode for editing \LaTeX\ text that
  makes composing text much easier.} and E--mail them to
anyone. \LaTeX\ is based on the philosophy that the writer should have
an easy time composing and editing text\footnote{For example, with one
  \code{Emacs} command you can change the first word of every figure
  caption to be in another font, or change the size of all included
  figures.} but she should not have to spend time making text look
good on the screen. Instead the writer needs to concentrate on the
logical elements of composition; \LaTeX's job is to make the final
output look good.

\subsection{Two \LaTeX\ Output Modes}
When the \code{latex} command is run to compile your \LaTeX\ source
code, \LaTeX\ produces a dvi (``device independent'') file containing
the typeset document in a very compact form.  Graphics are not
included in the dvi file, but pointers to the graphics files are
included.  The dvi file can be printed directly, or it can be
converted into a self--contained postscript or pdf file.  Here are
some example \LaTeX-related system commands.
\begin{verbatim}
```

```
latex myfile                  % create myfile.dvi from myfile.tex
dvips myfile                  % send myfile to a postscript printer
dvips -o myfile.ps myfile     % convert myfile.dvi to myfile.ps, with
                              % graphics
dvips -Pwww -o myfile.ps myfile    % use Type 1 fonts
dvipdfm myfile                % convert myfile.dvi to myfile.pdf
pdflatex myfile               % creates myfile.pdf directly if no
                              % postscript graphics are referenced
\end{verbatim}

Creation of a static document in one of these ways is the usual mode
of \LaTeX\ usage.  There is also a way of using \LaTeX\ to create
``live'' documents that are viewed on a monitor (either locally or
over the web) or printed.  These pdf documents may contain bookmarks,
hyperlinks to external URLs, links to E--mail addresses, etc.  If you
use the \code{hyperref} package in \LaTeX, the system will
automatically make all pertinent elements of your document
cross--indexed and hyperlinked, and you can also insert special
commands to link to areas outside the document such as URLs and
E--mail.

When viewing the document using Adobe Acrobat Reader, bookmarks can
appear in the left margin, allowing the user to click to jump to any
major section of the document.  Sections having sub--sections can have
their bookmarks expanded so that you can jump to the sub--sections.
You can jump to any figure while viewing the \code{List of Figures} and
to any table while viewing the \code{List of Tables}, in addition to
jumping to any area while viewing the \code{Table of Contents}.  If
your document is indexed, you can jump to any page for which an
indexed phrase is discussed.  You can optionally jump to pages in
which a given article is cited while viewing the \code{Bibliography},
in addition to the more standard jump from a citation to the
bibliographic reference.  If the \code{colorlinks} option is selected
(see code below), symbols that are hyperlinked appear in color;
clicking on them will cause the jump.  All of this is set up
automatically by \code{hyperref}, unlike the large number of flags that
must be put in a document manually if using Microsoft Word.

Instead of compiling the document using the \code{latex} system
command, you use the \code{pdflatex} command to create the pdf file
directly, with all bookmarks and hyperlinks.

This document was created in the fashion just described.  \code{PDF}
graphics files were created directly using an S \code{pdf} device
driver.  Below you will find the code in the preamble of the document
```

```
that set up the \code{pdf} document with hyper--referencing.
\begin{verbatim}
\usepackage[pdftex,bookmarks,pagebackref,colorlinks,pdfpagemode=UseOutlines,
    pdfauthor={Frank E Harrell Jr},
    pdftitle={Statistical Tables and Plots using S and LaTeX}]{hyperref}
\end{verbatim}

\subsection{Basic Table Making in \LaTeX}
\LaTeX\ has excellent facilities for composing and typesetting tables.
Table \ref{results} is an example of a user--specified table using
three macros --- \code{btable} (begin table), \code{etable} (end table),
and \code{mc} (headings that span multiple columns).  These macros save
repetitive operations.  Macros are usually defined at the top of
the document.
\begin{verbatim}
%Usage: \btable{table specs}{caption}{reference label}
\newcommand{\btable}[3]{
        \begin{table}[htbp]
        \begin{center}
        \caption{#2\label{#3}}
        \begin{tabular}{#1}}

\newcommand{\etable}{\end{tabular}
        \end{center}
        \end{table}}

%Usage: \mc{number of columns spanned}{major column heading}
\newcommand{\mc}[2]{\multicolumn{#1}{c}{#2}}

\btable{l|ccccc}{Overall Results}{results} \hline\hline
%6 fields, justified left, center x 5
%double horizontal line at top, 1 vertical bar
 & \mc{2}{Females} & & \mc{2}{Males} \\   % column 4 blank, for spacing
\cline{2-3} \cline{5-6}      % horizontal lines connecting cols. 2-3, 5-6
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \\ \hline
Placebo       & 0.21 & 163 & & 0.22 & 164 \\
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \\
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \\ \hline
\etable
\end{verbatim}

%Usage: \btable{table specs}{caption}{reference label}
\newcommand{\btable}[3]{
        \begin{table}[htbp]
        \begin{center}
        \caption{#2\label{#3}}
        \begin{tabular}{#1}}

\newcommand{\etable}{\end{tabular}
        \end{center}
        \end{table}}

%Usage: \mc{number of columns spanned}{major column heading}
\newcommand{\mc}[2]{\multicolumn{#1}{c}{#2}}

\btable{l|ccccc}{Overall Results}{results} \hline\hline
%6 fields, justified left, center x 5
%double horizontal line at top, 1 vertical bar
 & \mc{2}{Females} & & \mc{2}{Males} \\   % column 4 blank, for spacing
\cline{2-3} \cline{5-6}      % horizontal lines connecting cols. 2-3, 5-6
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \\ \hline
Placebo       & 0.21 & 163 & & 0.22 & 164 \\
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \\
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \\ \hline
\etable
The result is Table~\ref{results}.
However, the \texttt{ctable} style, available from
\href{http://www.ctan.org}{\url{www.ctan.org}} can produce prettier tables
more flexibly:
\begin{verbatim}
\ctable[caption={Overall Results},label=resultsb,pos=hbp!]{lccccc}{}{
\FL
& \mc{2}{Females} & & \mc{2}{Males} \NN
\cmidrule{2-3}\cmidrule{5-6}  % Important: no space before \cmidrule
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \ML
Placebo       & 0.21 & 163 & & 0.22 & 164 \NN
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \NN
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \LL
}
\end{verbatim}
The result is shown in Table~\ref{resultsb}.
\ctable[caption={Overall Results},label=resultsb,pos=hbp!]{lccccc}{}{
\FL
& \mc{2}{Females} & & \mc{2}{Males} \NN
\cmidrule{2-3}\cmidrule{5-6}
Treatment     & Mortality & Mean Pressure & & Mortality & Mean Pressure \ML
Placebo       & 0.21 & 163 & & 0.22 & 164 \NN
ACE Inhibitor & 0.13 & 142 & & 0.15 & 144 \NN
Hydralazine   & 0.17 & 143 & & 0.16 & 140 \LL
}
```

```
\section{Using S to Fill in Cells in \LaTeX\ Tables}
For most statistical tables a better idea is to avoid transcription of
calculated values by having the values inserted into tables
automatically.  The \texttt{Hmisc} library (see
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc}})
contains several S functions by R Heiberger and F Harrell that
automatically make \LaTeX\ tables from
S objects\footnote{More advanced applications of this are found
in the \code{Design} library, such as automatic \LaTeX\ typesetting of
fitted regression models with simplification of interaction and spline
terms, and typesetting of $\chi^2$ tables showing all regression
effects.  These examples are beyond the scope of this document.  See
\href{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf}
{\url{biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf}}, Chapter 9.}.
S functions that automatically produce \LaTeX\ code from S
objects (matrices, fitted models, data summaries, etc.) have names
that start with \code{latex}.  Tables produced by the \code{latex.*}
functions in \code{Hmisc} meet the stylistic requirements of
most journals, i.e., by default they do not use vertical lines and
they use horizontal lines only when needed.  In this way the lines do
not distract from delivering the statistical information.

Suppose that some calculations have already been made using S,
and these calculations were not stored.  For example, you may have
estimated various effects and standard errors but forgot to store the
S regression fit objects so that you can pull these values into
tables automatically.  You can use the \code{latex.default} function
that is part of \code{Hmisc} for automatic conversion of the
calculations into \LaTeX, after entering basic statistics manually.
Let us have S calculate odds ratios and $P$--values to avoid
transcribing them after we print $\hat{\beta}$ and standard errors.
Here is the S program for creating the table that is inserted
into this document as Table \ref{summary.stats}.
\sinput{summary.stats.s}
\input{summary.stats}

There are many other options to the basic \code{latex} function.  Type
\code{?latex} to access the online help.  You may be particularly
interested in the \code{longtable} option, which can be used to easily
break a long table into multiple pages (with repetitions of key header
information).

You can have your S program print hardcopy \LaTeX\ output
```

```
directly using the \code{prlatex} function.  More typically though you
will want the program to create \LaTeX\ files (with suffix \code{.tex})
that will be put together later.  In this way you can add title pages,
running headers or footers, and other text, and refer to tables by
symbolic names.  This document serves as an example of how this is
done, with its \LaTeX\ code listed in Section \ref{latex.code}.

If you like to specify table layouts inside the \LaTeX\ source file
rather than inside S, you can have your S program output
symbolic values to a file that is \verb|\input{}|'d in \LaTeX\ as
shown in the following example.  A restriction is that variable names
defined to \LaTeX\ may contain only letters and they should not
coincide with names of \LaTeX\ commands.
\begin{verbatim}
chisq <- (beta/se)^2
pval  <- 1 - pchisq(chisq, 1)
cat('\\def\\chisq{',round(chisq,2),'}\\n',   # \\ -> \ in parms.tex
    '\\def\\pval{',round(pval,4),'}\n', sep='', file='parms.tex')
\end{verbatim}
If \LaTeX\ variables are named the same as S variables, and
the names contain only letters, code can be simplified using a little
function.  This function can also convert \code{NA}s to blanks.
\begin{verbatim}
lvar <- function(x, digits=2)
  paste('\\def\\',substitute(x),'{',
        ifelse(is.na(x),'',round(x,digits)),'}', sep='')

cat(lvar(chisq), lvar(pval,4), sep='\n', file='parms.tex')
\end{verbatim}

The contents of file \code{parms.tex} will look like the following:
\begin{verbatim}
\def\chisq{3.84}
\def\pval{0.05}
\end{verbatim}
Inside the main \LaTeX\ source file use for example
\begin{verbatim}
\input{parms}
\ctable[caption={Main Results},label=resultsc]{lcc}{}{
Test        & $\chi^2$    & $P$--value \ML
Age effect  & \chisq      & \pval      \LL
}
\end{verbatim}

\section{Using S to Create Graphics for \LaTeX}
```

```
PostScript is a graphics format accepted by all versions of \LaTeX\ as
long as you have a PostScript printer or have GhostScript or Adobe
Acrobat Distiller to convert postscript output to other formats.  The
basic graphics driver in S for creating postscript files is the
\code{postscript} function.  For creating 35mm slides, overhead
transparencies, or $5 \times 7$ glossy figures, the \code{ps.slide}
function in the Hmisc library assists in setting up nice defaults for
postscript images.  For reports and books, the Hmisc \code{setps}
function makes creating of individual postscript graphics easy.
\code{setps} uses reasonable defaults and sets up for a minimally sized
bounding box.  It tries not to waste space between axes and axis
labels.  In the following example, a file called \code{test.ps} is
created in the current working directory.  Note the absence of quote
marks around the word \code{test} below.
\bex
setps(test)     \scom{use setps(test, trellis=T) if using Trellis (R Lattice)}
plot(....)
dev.off()       \scom{close file, creating test.ps}
\eex
As you will see later, we can symbolically label this figure using the
word \code{test} in \LaTeX.  By default, \code{setps} uses Helvetica
font and makes small book--style figures.  There are many options to
override these and other settings.

If you are using \code{pdflatex}, graphics files must be in Adobe \code{pdf}
format.  You can create \code{pdf} files directly in
\splus\ using the builtin \code{pdf.graph} function, in \R\ using the
\code{pdf} function, or using the Hmisc \code{setpdf} function.  In
older versions of \splus, better results are obtained by creating
postscript and converting the graph to \code{pdf}.  If you have
Ghostscript installed and have used \code{setps} followed by
\code{dev.off}, you can type \code{topdf()} with no arguments to invoke
Ghostscript from S to create, in this case, \code{test.pdf}.  You
can also convert from postscript to \code{pdf} using Adobe Acrobat
Distiller, which produces more compact \code{pdf} files.  In \R, direct
creation of \code{.pdf} files seems to work well.

\subsection{Inserting Graphics Files into \LaTeX\ Documents}
The standard \code{graphics} and \code{graphicx} packages in \LaTeX\
provide all you need to insert postscript and \code{pdf} graphics into a
document in a flexible fashion.  This is not to say that it is as easy
as using Word; frequently some trial and error is required to get
graphics to
have an appropriate scaling (magnification) factor.  Inclusion of \code{pdf}
graphics in older versions of \splus\ and \LaTeX\ frequently resulted
```

```latex
in much wasted spaced before and after the
graph when using \code{pdflatex}, so you often had to use
\verb|\vspace| commands with negative arguments when including \code{pdf}
files.

Here is a \LaTeX\ macro for inserting \code{pdf} graphics files, which are assumed to
have a \texttt{.pdf} suffix.
\begin{verbatim}
% Usage: \fig{label=.pdf prefix}{caption}{short caption for list of
% figures}{scalefactor}
\newcommand{\fig}[4]{\begin{figure}[hbp!]
 \leavevmode\centerline{\includegraphics[scale=#4]{#1.pdf}}
 \caption[#3]{\small #2}
 \label{#1}
 \end{figure}}
\end{verbatim}
For example, \verb|\fig{test}{long caption}{short caption}{.8}|
will insert \code{test.pdf} and reduce its size by $20\%$.  You can
refer to this figure in the text using for example
\verb|see Figure~\ref{test}|.

\section{Making S Compose \LaTeX\ Tables}
In many cases S functions can be used to make all calculations
for the table and then to create the \LaTeX\ table.  Harrell's S\
\code{summary} function for formulas (actually \code{summary.formula})
is one function that will do this when what you need is descriptive
statistics (including statistics computed by functions you create).
\code{summary} is in the Hmisc library available at
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/Hmisc}}.
It has three methods for
computing descriptive statistics on univariate or multivariate
responses, subsetted by categories of other variables.  See {\em An
Introduction to S and the Hmisc and Design Libraries} by CF
Alzola and FE Harrell
(\href{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf}
{\url{biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf}}) for
more information about \code{summary.formula} and S usage in
general, especially information on how to recode and reshape data to
be used in reports.

The output from \code{summary.formula} can be printed (for ordinary
text file printouts), plotted (dot charts or occasionally
box-percentile plots), or typeset using \LaTeX,
as there are several \code{print}, \code{plot}, and \code{latex} methods
```

```
for objects created by \code{summary.formula}.  The \code{latex} methods
create all the needed table elements, then invoke the
\code{latex.default} method in \code{Hmisc} to build the complete set of
\LaTeX\ commands to make each table.

The method of data summarization to be done by \code{summary.formula}
is specified in the parameter \code{method}.  These methods are defined
below.  For the first and third methods, the statistics used to
summarize the data may be specified in a flexible manner by the user
(e.g., the geometric mean, $33^{rd}$ percentile, or Kaplan--Meier
2--year survival estimate, mixtures of several statistics).  The
default summary statistic is the mean, which for a binary response
variable is the proportion of positive responses.

\begin{description}
\item[\code{method='response'}:\ ] The response variable may be
 multivariate, and any number of statistics may be used to summarize
 the responses.  Sometimes dependent variables are multivariate
 because they indicate follow--up time and censoring, and sometimes
 they are multivariate because there are several response variables
 (e.g., systolic and diastolic blood pressure).  The responses are
 summarized separately for each independent variable (independent
 variables are not cross--classified).  Continuous independent
 variables are automatically stratified into quantile groups.  One or
 more of the independent variables may be stratification factors, in
 which all computations are done separately by levels of these
 categorical variables.  The stratification variables form major
 column groupings in tables.  For multivariate responses, subjects are
 considered to be missing if {\em any} response variable is missing.
\item[\code{method='reverse'}:\ ]
 This format is typical of baseline characteristic tables describing
 the usual success of randomization.  Here the single dependent
 variable must be categorical (e.g., treatment assignment), and the
 ``independent'' variables are broken down separately by the dependent
 variable.  Continuous independent variables are described by three
 quantiles (quartiles by default), and categorical ones are described
 by counts and percentages.  There is an option to automatically
 generate test statistics for testing across columns of
 \code{'reverse'} tables.
\item[\code{method='cross'}:\ ]
 The \code{'cross'} method allows allows for multiple dependent
 variables and multiple statistics to summarize each one.  If there is
 more than one independent variable (up to three is allowed),
 statistics are computed separately for all cross--classifications of
 the independent variables, and marginal and overall statistics may
 optionally be computed.  \code{summary.formula} for this method
 outputs a data frame containing the combinations of predictors along
 with the response summaries.  This data frame may be summarized
 graphically in various ways using the \splus\ \code{trellis}
 library or \R\ \code{lattice} package\footnote{For this purpose, the
 Hmisc \code{summarize} function
 may be more useful, if you don't want marginal statistics computed.}.
 A \LaTeX\ printing method, for the case where there is exactly two
 predictors, typesets a two--way table where the first predictor forms
 rows and the second forms columns.  Like \code{method='response'}, continuous
 variables are automatically divided into quantile groups.
\end{description}

The \code{latex} methods in the \code{Hmisc} library
create tables using standard \LaTeX\ commands.  These tables are
inserted into the master document at the desired location using an
\verb|\input{}| command.  \code{latex} methods allow a font \code{size}
argument.  For example, you may specify \code{size='small'} to
\code{latex()}, or you may want to use a generic size that is set at
\LaTeX\ run time in the document preamble.  For example,
specify \verb|\def\tsz{small}| in the master document and
specify \code{size='tsz'} to \code{latex()}.  Then you can define (and
redefine) the size for tables without modifying the individual
\code{.tex} files created by \code{latex()}. Another approach using
\LaTeX's \code{relsize} style is discussed on P.~\pageref{relsize}.

\subsection{Reports Formatted to Describe Responses}
Tables \ref{s1}--\ref{s4b} were produced by the S \code{latex}
function (actually, \\ \code{latex.\-summary.\-formula.\-response}),
which is run on an object created by the \code{summary} function with
\code{method='response'}, the default.

Table \ref{s1} presents Kaplan--Meier 2 and 5 year survival estimates and
mean life length of subjects in the Mayo Clinic primary biliary
cirrhosis dataset available from
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets}}.
The calculations are subsetted on various patient characteristics.  For
estimating mean life length, an exponential survival model was assumed
(the estimate is years per event).  Continuous variables are
categorized into quartiles automatically.  Each quartile group is
identified using the upper and lower endpoints within that quartile.
The code for this example follows.
\sinput{kmsurv.s}
```

```
\rhead{\scriptsize The {\em EXAMPLE} Study \\
    Protocol xyz--001 \\
    \today}

\input{s1}

This table is converted to two dot plots (Figures \ref{f1a} and
\ref{f1b}) using the \code{plot} method for an object created by
\code{summary} with \code{method='response'} (see previous code).
The Hmisc \code{setpdf} function is used to create the \code{pdf}
graphics files.  See Section \ref{latex.code} for the \LaTeX\ code
used to insert these graphics.

\fig{f1a}{Two and five--year Kaplan--Meier survival probability
estimates}{Kaplan--Meier estimates}{1}
\fig{f1b}{Estimated mean life length from an exponential survival
model}{Estimated life length}{1}

Table \ref{s2} is similar to Table \ref{s1} except that the
Kaplan--Meier estimates are not shown, life length estimates are also
stratified by treatment assigned (using the \code{stratify} function),
and continuous variables are grouped into tertiles.
\sinput{s2.s}
{\small\input{s2}}
This table is converted to a dot plot in Figure \ref{f2}.
\fig{f2}{Estimated mean life length from an exponential survival
model}{Estimated life length stratified by treatment}{1}

Table \ref{s3} displays quartiles of cholesterol and bilirubin by
various patient characteristics.  To compute statistics simultaneously
for cholesterol and bilirubin, we must use the S \code{cbind}
function to create a bivariate response variable (a 2--column matrix).
To compute quantiles for this new 2--variable entity we have to use
the \code{apply} function instead of a simple invocation to
\code{quantile}.  For \code{age}, pre--specified intervals are used.
\sinput{s3.s}
\input{s3}
Table \ref{s3} is shown as a graphic in Figure \ref{f3}.
\fig{f3}{Quartiles of cholesterol and bilirubin}{Distribution of
cholesterol and bilirubin}{.8}

Tables \ref{s4a} and \ref{s4b} summarizes only bilirubin, but both the
mean and median are printed.  Separate tables are made for the two
arms of the randomized study.  For the active arm, the data are shown
in Figure \ref{f4}.
```

```
\sinput{s4.s}
\input{s4}
\fig{f4}{Mean (solid circle) and median (open circle) bilirubin for
D--penicillamine patients}{Mean and median bilirubin for treated patients}{1}
\clearpage

\subsection{Baseline Characteristic Tables}
Here the S \code{summary} function is used with the parameter
\code{method='reverse'}, which reverses the role of the dependent variable
and the independent variables.  The dependent variable is assumed to be
categorical; in clinical trials it will be the treatment assignment.

The next example again uses the primary biliary cirrhosis dataset.
The result is in Table \ref{s5}.  It is printed in landscape mode
using the \LaTeX\ \code{lscape} package, and using the \LaTeX\
\code{small} font.  For \code{'reverse'}-type tables, an option
\code{test=TRUE} will cause \code{summary.formula} to compute test
statistics for testing across columns.  Default tests are Wilcoxon or
Kruskal-Wallis for continuous variables and Pearson $\chi^2$ for
categorical ones, but users may specify their own statistical tests\footnote{
In randomized trials, tests for baseline imbalance are unwarranted and
difficult to interpret, in addition to causing multiple comparison
problems (see Stephen Senn, {\em Statistical Issues in Drug
Development}).}.
\sinput{s5.s}
\clearpage
\thispagestyle{empty}  % suppress page number
\begin{landscape}
\input{s5}
\end{landscape}

To convert Table \ref{s5} to graphical form,
\code{plot.\-summary.\-formula.\-reverse} constructs two pages.  The
first page contains statistics for all of the categorical variables,
as all of these statistics are on the same scale (proportion or
percent in each category).  The second page contains a matrix of dot
charts showing (by default) the 3 quartiles of each right--hand--side
variable (on the $x$--axis), stratified by the left--hand variable (on
the $y$--axis of each dot plot).  The second set of plots is scaled to the
most extreme 0.025 to 0.975 quantiles of the variable over all
treatment groups.  \R\ can plot Greek letters, superscripts,
subscripts, and mathematical operators, and Figure \ref{f5a} and
\ref{f5b} take advantage of this capability. \splus\ does not have
this capability, so simpler output would appear.
\fig{f5a}{Proportions of patients in various categories of baseline
variables, stratified by drug.  Pearson $\chi^2$ test results are
given.}{Categorical variables stratified by drug}{.75}
\fig{f5b}{Box-percentile plots for continuous baseline variables in prostate
cancer trial.  0.90, 0.75, 0.50, and 0.25 coverage intervals are
shown.  The solid circle depicts the mean and the vertical line the
median.  Kruskal-Wallis tests are also shown.}{Continuous variables
stratified by drug}{.7}

Table \ref{s6} presents a description of data from a trial for
prostate cancer (from Byar and Green).  The \code{prostate} data frame
is available from
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets}}.
The \code{overall} option is used to add a final column of statistics
for the whole sample.  The following listing contains code that
produced all the tables and figures for the \code{prostate} data.
This is a good application of the \LaTeX\ \code{relsize} style\label{relsize}.
Specifying an overall size of the table of \code{smaller[3]} causes
\code{latex()} to issue the command \verb|\smaller[3]| at the start of
the table and changes the overall table's font size to three levels below
\code{normalsize}, which is \LaTeX's \code{scriptsize}.  Specifying
\code{outer.size} and \code{Nsize} as \code{smaller} means to use one
size smaller than this within the table, for $25^{th}$ and $75^{th}$
percentiles and for the sample sizes above the columns.  One advantage
of \code{relsize} is that if you use for example \verb|{\smaller foo}|
within a footnote, the next smaller size than is used for the overall
footnoted text will be the size for \code{foo}.
\sinput{prostate.s}
\clearpage
\thispagestyle{empty}
\begin{landscape}
\input{s6}
\end{landscape}
\fig{f6a}{Distribution of categorical baseline variables in prostate
cancer trial}{Categorical variables in prostate trial}{.8}
\fig{f6b}{Quartiles of continuous variables in prostate cancer trial.
  $x$--axes are scaled to the lowest 0.025 and highest 0.975 quantiles
  over all groups for each variable.}{Continuous variables in prostate
  trial}{.8}
\clearpage

\subsection{Data Displays from Cross--Classifying Variables}

The final examples use cross--classification on possibly more than one
independent variable.  The summary function with \code{method='cross'}
```

```
produces a data frame containing the cross--classifications.  This
data frame is suitable for multi-panel trellis displays although if
marginal statistics are not needed, the Hmisc \code{summarize} function
is better.  The first example in this series was \LaTeX'ed to create
Table \ref{s7} (the code is listed above).
\input{s7}
There is no \code{plot} method for \code{method='cross'} tables, but you
can use Trellis graphics on the data frame that is created by
\code{summary} (see code above).  For this purpose, the \code{Hmisc}
\code{summarize} function might be better than \code{summary.formula}
for producing the needed aggregated data.
\fig{f7}{Proportion of patients with acid phosphatase exceeding 1.0,
cross--classified by tumor size and bone metastasis}{Proportion of
patients with AP $ > 1.0$}{1}

%Turn off header
\rhead{}

\section{Handling Special Variables}
\subsection{Multiple Choice Variables}
Clinical reports freququently must summarize ``checklist'' or
multiple--choice variables.  Such variables are typically listed on a
case report form using one of two methods:
\begin{enumerate}
\item   Specify up to three primary presenting symptoms: \\
        \verb|_________  ________ ________| \\
        Here the respondent writes in up to three symptom codes from a
        list of perhaps 15 integer codes defined below the question.
\item   Check symptoms that are present: \\
        \verb|headache   __   stomach ache __   hangnail __| \\
        \verb|back pain  __   neck ache    __   wheezing __|
\end{enumerate}
When such data are processed, either a series of three categorical
variables or 6 binary variables is created.  In what follows we assume
that the binary variables are coded as numeric 0/1 or as character
variables with values (ignoring case) of \code{'yes'} and
\code{'present'} denoting a positive response.  In composing a report,
we usually want to consider all of these component variables under the
umbrella of \code{'Presenting Symptoms'}.  If using presenting symptoms
as stratification (independent) variables, we will want to know an
outcome statistic computed separately for those subjects having
headache, those having stomach ache, etc.  These categories will
overlap for some subjects.  When summarizing presenting symptoms
stratified by treatment, we will want to know the proportion of
subjects in each treatment group having headache, the proportion
```

```
having stomach ache, etc., with the proportions summing to $ > 1.0$ if
any subject had more than one symptom.

The Hmisc \code{summary.formula} function can handle multiple choice /
checklist variables after they are combined into a matrix.  The Hmisc
\code{mChoice} function will take as input a series of categorical
vector variables (using the first input format above), and make a
matrix with the number of columns equal to the number of choices that
were actually selected in the data\footnote{There is also an option to
create a column for \code{'none'} for subjects for whom no choices were
selected.  The input variables need not have the same levels.  A
master list of categories is constructed by finding all unique
categories in the levels of all variables combined, preserving the
order of levels for the factor variables.}.  This new matrix consists
of logical \code{T/F} values.  You can also give \code{summary.formula}
a matrix you create, if using input format two above.  The elements of
this matrix need to be numeric with values 0 and 1, logical \code{F/T},
or character with values (ignoring case) of \code{'yes'} or \code{'present'}.

Here is an example of the use of \code{mChoice} from its help file.
\bex
> options(digits=3)
> set.seed(173)
> sex \Gets factor(sample(c("m","f"), 500, rep=T))
> age \Gets rnorm(500, 50, 5)
> treatment \Gets factor(sample(c("Drug","Placebo"), 500, rep=T))

> # Generate a 3-choice variable; each of 3 variables has 5 possible levels
> symp \Gets c('Headache','Stomach Ache','Hangnail',
+           'Muscle Ache','Depressed')
> symptom1 \Gets sample(symp, 500, T)
> symptom2 \Gets sample(symp, 500, T)
> symptom3 \Gets sample(symp, 500, T)
> Symptoms \Gets mChoice(symptom1, symptom2, symptom3, label='Primary Symptoms')

> # Note: In this example, some subjects have the same symptom checked
> # multiple times; in practice these redundant selections would be NAs
> # mChoice will ignore these redundant selections
> # If the multiple choices to a single survey question were already
> # stored as a series of T/F yes/no present/absent questions we could do:
> # Symptoms <- cbind(headache,stomach.ache,hangnail,muscle.ache,depressed)
> # where the 5 input variables are all of the same type: 0/1,logical,char.
> # These variables cannot be factors in this case as cbind would
> # store integer codes instead of character strings.
> # To give better column names can use
> # cbind(Headache=headache, 'Stomach Ache'=stomach.ache, ...)

> # Following 8 commands only for checking mChoice
> data.frame(symptom1,symptom2,symptom3)[1:10,]

       symptom1     symptom2     symptom3
 1     Headache Stomach Ache     Headache
 2    Depressed  Muscle Ache    Depressed
 3 Stomach Ache  Muscle Ache Stomach Ache
 4     Hangnail  Muscle Ache     Headache
 5  Muscle Ache     Headache    Depressed
 6     Headache     Headache     Headache
 7 Stomach Ache Stomach Ache  Muscle Ache
 8  Muscle Ache     Headache    Depressed
 9     Hangnail     Hangnail     Hangnail
10    Depressed  Muscle Ache    Depressed

> Symptoms[1:10,]  # Print first 10 subjects' new binary indicators

Primary Symptoms
      Depressed Hangnail Headache Muscle Ache Stomach Ache
 [1,]         F        F        T           F            T
 [2,]         T        F        F           T            F
 [3,]         F        F        F           T            T
 [4,]         F        T        T           T            F
 [5,]         T        F        T           T            F
 [6,]         F        F        T           F            F
 [7,]         F        F        F           T            T
 [8,]         T        F        T           T            F
 [9,]         F        T        F           F            F
[10,]         T        F        F           T            F

> meanage \Gets single(5)
> for(j in 1:5) meanage[j] \Gets mean(age[Symptoms[,j]])
> names(meanage) \Gets dimnames(Symptoms)[[2]]
> meanage

 Depressed Hangnail Headache Muscle Ache Stomach Ache
      49.9     49.8     49.9        50.3         49.8
> # Manually compute mean age for 2 symptoms
> mean(age[symptom1=='Headache' | symptom2=='Headache' | symptom3=='Headache'])
[1] 49.9
> mean(age[symptom1=='Hangnail' | symptom2=='Hangnail' | symptom3=='Hangnail'])
[1] 49.8
```

```
> #Frequency table sex*treatment, sex*Symptoms
> summary(sex {\Twiddle} treatment + Symptoms, fun=table)
> # could also do summary(sex {\Twiddle} treatment + mChoice(symptom1,...),...)

sex    N=500

----------------+------------+---+---+---+
                |            |N  |f  |m  |
----------------+------------+---+---+---+
treatment       |Drug        |246|123|123|
                |Placebo     |254|129|125|
----------------+------------+---+---+---+
Primary Symptoms|Depressed   |242|130|112|
                |Hangnail    |238|125|113|
                |Headache    |236|110|126|
                |Muscle Ache |255|127|128|
                |Stomach Ache|252|125|127|
----------------+------------+---+---+---+
Overall         |            |500|252|248|
----------------+------------+---+---+---+

> #Compute mean age, separately by 3 variables
> summary(age {\Twiddle} sex + treatment + Symptoms)

age    N=500

----------------+------------+---+----+
                |            |N  |age |
----------------+------------+---+----+
sex             |f           |252|49.8|
                |m           |248|49.9|
----------------+------------+---+----+
treatment       |Drug        |246|49.7|
                |Placebo     |254|50.0|
----------------+------------+---+----+
Primary Symptoms|Depressed   |242|49.9|
                |Hangnail    |238|49.8|
                |Headache    |236|49.9|
                |Muscle Ache |255|50.3|
                |Stomach Ache|252|49.8|
----------------+------------+---+----+
Overall         |            |500|49.9|
----------------+------------+---+----+

> f \Gets summary(treatment {\Twiddle} age + sex + Symptoms, method="reverse")
```

```
Descriptive Statistics by treatment

----------------------------+--------------+--------------+
                            |Drug          |Placebo       |
                            |(N=246)       |(N=254)       |
----------------------------+--------------+--------------+
                         age|46.5/49.8/52.5|46.4/50.1/53.4|
----------------------------+--------------+--------------+
                     sex : m|   50% (123)  |   49% (125)  |
----------------------------+--------------+--------------+
Primary Symptoms : Depressed|   50% (122)  |   47% (120)  |
----------------------------+--------------+--------------+
                    Hangnail|   47% (116)  |   48% (122)  |
----------------------------+--------------+--------------+
                    Headache|   45% (110)  |   50% (126)  |
----------------------------+--------------+--------------+
                 Muscle Ache|   48% (117)  |   54% (138)  |
----------------------------+--------------+--------------+
                Stomach Ache|   53% (130)  |   48% (122)  |
----------------------------+--------------+--------------+
\eex

\subsection{Conditionally Defined Variables}
Another type of variable that is common in clinical reports is a
variable that is of no interest unless another variable equalled a
certain value.  A common example is cause of death.  We may want our
report to contain the proportion of patients dying on each treatment,
and for the deaths, we may want to know the proportions of deaths due
to each cause.  For the latter calculation, the denominator is not the
number of subjects in a treatment but rather the number of subjects
who died on that treatment.  \code{summary.formula} will handle such
variables correctly as long as they have missing values when they are
not pertinent.  For example, suppose that the variable
\code{death.cause} is \code{NA} if \code{death} is \code{F} (false)
and \code{death.cause} is a categorical (or \code{mChoice}) variable if
\code{death} is \code{T}.  Then a \code{'reverse'} type summary will produce
the needed proportions of \code{death} as well as \code{death.cause}.

\section{Alternate Approaches}
\subsection{Literate Programming}
In \emph{literate programming} as used in reproducible research (see
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatReport}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatReport}}),
a single source document contains analysis code as well as text for
```

```
the report.  This has been found to be easier to maintain and to
result in better documentation.  Under \R, the \code{Sweave} package
provides a concise syntax for mixing S and \LaTeX\ code for producing
reports, as discussed in Section~16.3 of the course notes at
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse}}.
\code{Sweave} will run the S code chunks through \R, include S
printed output in the report, and will generate \LaTeX\ commands to
automatically include graphics generated by the S code.  One
especially nice feature of \code{Sweave} is the ease with which users
can insert variables computed by S into \LaTeX\ text without the need
of the \verb|\def\varname{value}| approach described earlier.

\code{Sweave} is particularly well suited for non-recurring
statistical reports.  Reports that are run after periodic data
updates, for which the time spent polishing the report is well spent,
are sometimes better suited to the customized programming methods
described earlier in this document.

\subsection{\LaTeX\ Server}
The UVa Biostatistics \LaTeX\ server allows the user to upload S
output that contains a mixture of S commands and printed output and to
upload a \code{.zip} file containing all the postscript graphics files
for the report, and will run \LaTeX\ on the server, automatically
including graphics and making it easy for the user to provide legends
for the plots.  The user can then download a \code{.pdf} document
containing the typeset report.  See Chapters~2, 6, and 11 of the
course notes at
\href{http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse}
{\url{biostat.mc.vanderbilt.edu/twiki/bin/view/Main/StatCompCourse}}
for more information.

\section{Data Preparation}
For making nice--looking tables, as well as for having
self--documenting variables, it is important to spend time defining
good variable and value labels.  If you are managing the data in SAS,
for example, specify nice variable labels in a DATA step or using PROC
DATASETS, and specify pretty value labels using PROC FORMAT.  Both
variable and value labels should use letter cases carefully.  Don’t
use all upper case for either kinds of labels.  Variable labels should
often contain units of measurements.  An example of a good label is
\code{’Serum Cholesterol, mg/dl’}.  Better still, separate the
\code{’units’} attribute from the \code{’label’} attribute of a
variable:
\bex
```

```
label(chol) \Gets 'Serum Cholesterol'
units(chol) \Gets 'mg/dl'
# Alternate approach:
mydata \Gets upData(mydata, labels=c(chol='Serum Cholesterol'),
                         units =c(chol='mg/dl'))
\eex
Some of the \code{latex} and \code{plot} methods in the \code{Hmisc} and
\code{Design} libraries make special use of \code{units} attributes by
typesetting them in a different font or by right-justifying units in
cells of \LaTeX\ tables.

Binary variables are often coded 0/1.  Good variable labels for these
are of the form \code{'Nocturnal angina present'}.  Sometimes you may
want printouts to be more self--documenting.  Then consider defining a
SAS format of the form \code{0='Angina absent' 1='Angina present'}.

You can always change labels and value labels after data are imported
into S.  Here are some examples.
\bex
  label(age) \Gets 'Age (y)'
  levels(pain) \Gets c('None','Mild','Moderate','Severe')
  levels(pain) \Gets list('Moderate/Severe'=c('Moderate','Severe'))
  #Combines last two levels for subgroup analyses in which
  #there were two few patients with severe pain

  levels(symptom)[3] \Gets 'Night sweats'   # fix one level

  #Give fuller labels to levels of a binary variable
  nangina \Gets factor(nangina, 0:1, c('Absent','Present'))
\eex
The Hmisc \code{upData} function provides a more general approach for
changing variable attributes.  See Section~4.1.5 of
\href{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/RS/sintro.pdf}{Alzola
  and Harrell}.

The Hmisc \code{sas.get} function is used to translate SAS data to an
S data frame, carrying all data attributes.  There are options to
handle special missing values.  A typical procedure is to make an
S program called \code{create.s} for each project directory.
This program is run only whenever the SAS data changes.  The create
program should run the Hmisc \code{describe} function (and possibly
the \code{hist.data.frame} or \code{datadensity} function) to check each
variable being analyzed for valid values and to make sure that key
data are seldom missing.  Here is a typical \code{create.s}:
\bex
```

```
  rct \Gets sas.get('/my/data/path', 'rct', format.library='/my/formats',
                var=Cs(age,sex,treatment,dtime,death,pressure),
                uncompress=T)  #automatically uncompresses .ssd01 files
  #Cs() quotes all names (doesn't work if SAS names contain underscores)

  describe(rct)
\eex

If you run S interactively to develop and debug your reporting
programs, you will find it handy to make a pop--up window showing
variable names, labels, and value levels.  To do this, issue the
command \code{contents(rct)} after getting access to the \code{Hmisc}
library, where \code{rct} is the name of your randomized trial data
frame.  To pop--up a more detailed window with distributions for each
variable, use for example \code{page(contents(rct), multi=T)} (in
\splus).  There is also an \code{html} method for the results of
\code{contents}, to allow you to view metadata in a browser (with
hyperlinks between variables and value labels).  See
\href{http://biostat.mc.vanderbilt.edu/twiki/pub/Main/DataSets/Cpbc.html}
{\url{biostat.mc.vanderbilt.edu/twiki/pub/Main/DataSets/Cpbc.html}} for
example HTML output from \code{contents()}.

If you want to make variable label or value label changes in S
permanent, one option is to add the following type of statements after
the \code{sas.get} command above.
\bex
  attach(rct, pos=1, use.names=F)
  label(trt) \Gets 'Treatment'
  sex \Gets factor(sex, c('f','m'), c('Female','Male'))
  xx \Gets factor(xx, c('a','b'), c('A label','B label'), exclude='Unknown')
  # Treat 'Unknown' as a missing value instead of a level
  ...
  detach(1, 'rct')
\eex
A safer approach follows.
\bex
rct \Gets upData(rct,
              labels=c(trt='Treatment'),
              sex=factor(sex,c('f','m'),c('Female','Male')),
              xx =factor(xx, c('a','b'), c('A label','B label'),
                         exclude='Unknown'))
\eex

See the Alzola and Harrell online text for much more information about
modifying and recoding variables and reshaping data.
```

```
The Hmisc function \code{Label} will generate S assignment
statements containing all \code{label}s for variables in a specified
data frame.  You can edit the file output by \code{Label} to easily
modify labels you don’t like.  Look at the help file for \code{label}
for more information.

If you run \code{summary} output through \code{latex()}, caret signs in
variable labels and sometimes in value labels will cause the word after
the caret (up to the next space, comma, or end of string) to be superscripted.
Also, the symbols \verb|< <= > >=| will be translated to the proper math--mode
symbols such as $\geq$.  There are other cases in which you may want to
embed \LaTeX\ codes inside labels, e.g.:
\bex
  label(x2) \Gets ’\$X\_{2}$’
\eex
which results in \code{x2} being typeset as $X_{2}$.


\section{Inserting \LaTeX\ Output into non--\LaTeX\ Applications}
You can use \LaTeX\ to create tables and other text or graphics and
convert the output file to encapsulated postscript (EPS) for insertion
into Word or Wordperfect ‘‘pictures’’.  These pictures will not be
viewable on the screen (a blank box with be displayed) but they will
print correctly as long as you remember to set your printer to a
postscript printer before actually printing.  Once you import the
picture you can re--size it (if you use a 300 dpi postscript driver,
making the image larger will result in fuzzy printing).

Use the \code{dvips} program to make an EPS file from a \LaTeX\ dvi
file, using the \code{E} option.  Here is an example for the simple
case in which the document is only one page long (e.g., it consists of
a single table).
\begin{verbatim}
  dvips -E -o doc.eps doc    # creates doc.eps from doc.dvi
\end{verbatim}
If you have a multiple--page \LaTeX\ document, you can tell
\code{dvips} which page to store in a separate EPS file, for example,
page 9:
\begin{verbatim}
  dvips -E -p 9 -l 9 -o nine.eps doc
\end{verbatim}
You can even have \code{dvips} put every page of the document into a
separate file.  The files will be numbered e.g.\ \code{doc.001,
doc.002, doc.003, \ldots}:
```

```
\begin{verbatim}
  dvips -E -S 1 -i -o doc.0 doc
\end{verbatim}

Note that S plots are already in EPS, so you can include them in any
document with no extra steps, as long as you stored only one plot in the
EPS file.  A nice way to pick out individual plots and store them in a
separate \code{.ps} file is to use a postscript utility program called
\code{psselect}, e.g. if you created 3 pages of plots in \code{myplots.ps} use
\begin{verbatim}
  psselect -p1 myplots.ps myplots1.ps
\end{verbatim}
to put the first page of \code{myplots.ps} into \code{myplots1.ps}.
\code{psselect} can also be used to split out desired pages from a
postscripted version of a \LaTeX\ document as an alternative to using
the page number or section splitting options to \code{dvips}.  Michael
Stevens of Duke University has written a program called \code{oneperpg}
which will go through a multiple--page postscript file and
automatically create separate files each containing one page of
output, using \code{psselect}.  For example, typing
\begin{verbatim}
  oneperpg myplots
\end{verbatim}
creates \code{myplots1.ps, myplots2.ps, myplots3.ps}.

Another way, at least for UNIX or Linux users, to use \LaTeX\ output
in other applications, is to run the \code{latex2html} program to
convert the .tex files into an .html file.  This was done with Table
\ref{s3} by running the following code (\code{test.tex}) through
\code{latex2html} using the system command \code{latex2html test}:
\begin{verbatim}
\documentclass{article}
\begin{document}
\input{s3}
\end{document}
\end{verbatim}
The result can be found in
\href{http://hesweb1.med.virginia.edu/biostat/s/doc/s3.html}
{\url{hesweb1.med.virginia.edu/biostat/s/doc/s3.html}}.

You can insert the HTML file into Microsoft Word 97 documents, but if
you save the document as a Word file rather than as HTML, special
formatting such as \LaTeX\ font size changes will be lost.  This is
because Microsoft is not consistent in how enhanced HTML commands are
implemented in Internet Explorer and in Word.  In addition to this
```

```
problem, \code{latex2html} does not convert all table commands
properly; sometimes the program just stops in the middle of the
conversion.  If you have any math commands in the document,
\code{latex2html} has to convert these to GIF images.  See
\href{http://www.tex2html.com/}{\url{www.tex2html.com}} for more
information.

In general, HeVeA (\href{http://pauillac.inria.fr/~maranget/hevea/}
{\url{http://pauillac.inria.fr/~maranget/hevea/}}) does an excellent job in
converting \LaTeX\ code to HTML, without the need for graphics images
for math commands.  For some applications the resulting HTML can
easily be inserted into Word documents.

\section{S Documention}
Use the command \code{?summary.formula} under S to get
detailed document of \code{summary.formula} and its \code{print},
\code{plot}, and \code{latex} methods.

\section{\LaTeX\ Code for This Document} \label{latex.code}
{\small\verbatimtabinput{summary.tex}}

\end{document}
```