MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

A.I. memo No. 585

September, 1980

Primer for R users

Judi Jones

R is a text formatter. The information in this primer is meant to explain, in simple English, the basic commands needed to use R. Input for R is prepared on computer systems using a text editor. Which editor is employed depends on which computer system you use, and your personal preference. Almost every characteristic of a document can be controlled or changed if necessary. The difficulty comes in knowing the proper way to make the change. Most features have a variety of possible options. R is designed with automatic defaults which select choices for these options when they are not explicitly specified. For example, ".list" is designed to bring in the margins on both sides by 800 mils, causing the list to be indented from the rest of the text. Wherever possible both the default and methods for overriding or changing it will be given. We start by explaining the various parameters possible for the most useful and common commands. To thoroughly explain every command and its options would require a substantial document, and it is hoped that after reading the first few sections of this memo the user will have obtained enough information to continue alone using the on-line documentation (called the R Guide (try ":INFO RGUIDE" or "XINFO RGUIDE"), and the R Reference Manual, both currently managed by Eliot Moss.

The formatting process involves several steps. Any document you create is stored within the computer as a FILE, under a name chosen by you. The name generally has two parts, separated by a space (or a dot, depending on the system you use). Such files must be created on the computer using an EDITOR program. With most MIT editors, you type on a keyboard and characters appear on the screen before you. The display terminal's screen shows a number of LINEs, each comprised of CHARACTERs (even the spaces are called characters).

The contents of an on-line document file acceptable to R will begin with the specification of the device to be used for printing, followed by the font set-up, the text, and the formatting commands. After the file (called the INPUT, or SOURCE file) is created, it must be processed by R. A new file (called the OUTPUT file), suitable for printing on the selected device, is produced by R. The second name of the output file varies, depending on the device to be used for printing (e.g.: XGP, PRESS, LPT, V). The output file is then sent to the printing device, using an appropriate computer program, and once printed, the actual output can be physically picked up by the user.

In sum, one first EDITs a SOURCE file, then runs R to make an OUTPUT file, then orders the computer the print the newly created output file.

Let us begin by setting forth the conventions used in this primer. All R commands are set apart in quotation marks of this kind: ". In actual use all commands to R must begin with a "." at the left margin, and each command must stand alone on a line. Control characters are denoted by a "↑" followed by a capital letter. These are created on your terminal by holding down the control key while the letter is typed. For example, after pressing control' and A', ↑A will appear on your screen (you will have to precede the ↑A with another letter, probably ↑Q). We use the character "#" to mean an arbitrary font designator, namely: a numeral (0-9), or one of the letters A-F, or a * character. "↑K" is used to indicate that a comment follows. The "↑K" tells R to ignore the rest of the line.

This document has five major sections. The first describes how a document file is organized. That is, we tell you what things must be included in the file besides the text you want printed. The second section talks about the common simple commands used in R for formatting your text. The third section deals with some very useful special features. These are illustrated with examples and diagrams. The fourth section deals with string and number registers. These can really simplify your life by making formatting easier, but you need not use or understand them before you can use R. The fifth and final section describes some extra helpful macros, which are stored in the R directory available for your use. Again, you are not at all obliged to use or understand these, but they can be very helpful in certain situations.

You should definitely read the first three sections, and try to read the last two sometime, though you may wish to wait until you have used R for a little while.

I. File Set-Up

FORMAT	EXAMPLE
.dv device-used	.dv xgp
.if press	.if press
. fo 0 regular (Dover)	. fo 0 TimesRoman10
. fo 1 **bold** (Dover)	. fo 1 TimesRoman10b
. fo 2 *italic* (Dover)	. fo 2 TimesRoman10i
. fo 3 titles (Dover)	. fo 3 TimesRoman14
.else	.else
. fo 0 regular (XGP, etc.)	. fo 0 times.10Rom
. fo 1 **bold** (XGP, etc.)	. fo 1 times.10Bold
. fo 2 *italic* (XGP, etc.)	. fo 2 times.10Ital
. fo 3 titles (XGP, etc.)	. fo 3 times.14Rom
. end if	. end if
.so r.macros	.so r.macros
.tr @	.tr @

Notes:

The command ".tr @" specifies that @ is translated into a blank space. Therefore R doesn't print the @, and it looks like a space in the output. When there is only an @ between two words, R considers the words joined, as if they were one word, and will not make a line break between them. If you have ".tr @" in your file, and then you want to print a real @, just put a "↑O" before the @. The ".tr" command, and its close relative ".tc", can do more than just make "dummy" characters (see the R Reference Manual for details).

The command ".so r.macros" tells R to read the standard macro package along with your file. Among other things, this enables you to use some of the macros mentioned in section V. You should never omit the ".so r.macros" line unless you are an expert.

The above set-up will work for the Dover (Press) and the XGP, and also some other devices (LPT and Diablo). You may have to change the names in the ".fo" lines to use the Varian.

A. Device-used

1. for XGP (Xerox Graphics Printer)

   This machine is located on the ninth floor. Your file set-up can begin with ".dv xgp". Alternatively, if you have no ".dv" command, or a ".dv" for other than the XGP, you must include -x in the call to R, to tell R that you want an XGP file produced.

2. for press (dover, Tremont)

   Your file set-up must begin with ".dv press", or you can tell R to make a press file by adding "-p" after the file name when you run R on the file, e.g., ":r myfile -p".

3. for line printer (lpt)

   Here again your file set-up must begin with ".dv lpt", or you must tell R to make an lpt file by adding "-l" after the file name when you run R on the file. Only one type style is used in printing this way (because that's all a line printer has). The line printer normally underscores for fonts other than 0. But it is possible to make the line printer do nothing special, underline, capitalize, or overstrike (to make it look bold) for different fonts.

4. for diablo (carbon typewriter)

   There are several steps to follow to link the typewriter through the network to TOPS-20 or ITS. These steps are described in a file called "DIABLO INFO" in the R directory on each of our machines. It is wise to check the width and length switch settings under the cover, and set the form feed at the top of the page. The Diablo's treatment of different fonts is similar to that of the line printer.

5. varian

   The varian can only be used from the UNIX or VAX machines. You can begin your file set-up with ".dv varian", or put a "-v" in the command line to R, e.g., "r myfile -v".

B. Fonts

The variety of fonts available when using the XGP is tremendous, and it is even possible to make up arbitrary new fonts. The easiest way to see what the characters in a particular font look like is to request a sample from the XGP (or look at "The Last Whole XGP Font Catalogue 1980" - AI working paper 197). From an ITS machine a sample is obtained by typing:

":xgp ;sample directory;font name".

From XX you type:

"xgpsend <directory>font.name/sample".

The Dover has a more limited assortment of fonts. There will be a catalog for them available sometime, but it has not been created yet. On UNIX a font list is available by typing ">p /fonts |lpr<cr". This will print the list on the fourth floor line printer. The Varian fonts are generally replicas of the corresponding XGP fonts.

C. Registers (string and number)

Usually the commands for setting up any string or number registers (the ".nr" and ".sr" lines) should be placed just before the ".so r.macros" line. Section V discusses the use of string registers, and various option settings are mentioned throughout.

II. Common Commands

A. .sp Space

This leaves one empty line (a blank line in the file has the same effect). This command may be modified by a number, indicating to R how many lines to leave (e.g., ".sp 4"), or a specification in inches or mils. If you say ".sp 4L", then you will get the space that 4 lines would take up in the current line spacing. For example, if you are using double spacing (see ".ls" below), then ".sp 4L" is equivalent to ".sp 8". In most cases you probably want the form with the "L".

The space between words is a different concept! You should know, though, that the width of such a space varies according to the font being used, whether you are in no-fill mode, etc. The main effect is that the way things line up on your terminal screen is not usually the way they will line up in the output. If you have special alignment requirements, read the section "Alignment" in the R Guide.

B. .br Break

This causes a BREAK to a new line without leaving an empty space. All "break" means is that R stops the current output line, and starts a new one. Spaces, tabs, and ↑P's at the start of a line also cause breaks, as do a number of R commands, such as ".sp", ".in", etc. The Reference Manual describes which commands do and do not break, and how to prevent a command from breaking.

C. .ls Line Space

The default line spacing in R is 1, i.e., single spaced. When you are working on a draft version it is sometimes helpful to double space the file. To do this you need to use the line space command and increase the number to 2 (".ls 2"). Sometimes a fractional line spacing can give a pleasing appearance (this document was prepared using ".ls 1.25").

D. Fill vs. No-Fill

The difference between fill and no-fill has to do with the number of words put on an output line. In fill mode, R combines separate lines from the input, using as many words as necessary to fill a line. In no-fill each line in the input produces one line in the output.

E. .fi B, .fi L, and .fi R Fill Both, Fill Left, or Fill Right

Fill Both

R is instructed to fill, and to adjust the text to touch both margins, that is, to create an even edge on each side of the text. It does this by adding blank space between words. This is the normal mode for running text.

Fill Left or Fill Right

This only controls (by use of "L" or "R") which side R should justify (keep straight and even). Thus "L" means to produce a justified left margin (as these lines show), leaving the right side ragged. This is often useful when producing bibliographies, where there often are not enough words on each line to make both sides justified without leaving large spaces between words. This text was produced using ".fi L".

The command with "R" will produce a justified right margin, and leave the left side ragged. In fact, these lines were produced with ".fi R" command. We are not sure anyone has ever seriously used this.

F. .nf L or .nf R No Fill LEFT or No Fill RIGHT

With this command R is instructed not to fill up the line with text. The "L" or "R" tells R which side to make justified or straight. Below are two examples, the first done with ".nf L", and the second ".nf R". (".nf B" exists, but nobody has ever thought of anything useful for it.)

A wise old owl sat in an oak,
The more he saw the less he spoke,
The less he spoke the more he heard,
Why can't we all be like that wise old bird.

A wise old owl sat in an oak,
The more he saw the less he spoke,
The less he spoke the more he heard,
Why can't we all be like that wise old bird.

G. .nf C No Fill CENTER

R is instructed not to fill, and to center the following text.

This the way a line preceded by this command looks.

".nf C" is often useful in the title page of a document.

H. .top_of_page Begin New Page

R is instructed to put the following text on a new page. This is often useful to keep sections of a paper from running together, or, sometimes, to keep text together (but see also ".keep", described below). It is sometimes necessary to use human judgment and not rely entirely on R's judgment. ".top_of_page" is to be preferred to ".bp" in most cases, because ".bp" may produce "spurious"

blank pages. Of the two, ".top_of_page" is probably what you want if you are not sure.

I. .ne *no.*L Need *no.* lines

This command tells R to continue printing on this page only if the number of lines specified will fit on the page. You can also tell R in inches (mils, etc.) how much space is necessary before it begins to print a section. To do this you use the ".ne" command followed by the amount of space needed in inches or mils (e.g., ".ne 1.5i" or ".ne 1750m").

J. ↑F# To Change Fonts

The character typed after ↑F refers to a font in your file set-up. For example, ↑F3 switches to the font defined with ".fo 3". One must be sure to return to the original font after the required character string is printed by typing ↑F0 or ↑F*. ↑F* is generally to be preferred (it reduces the likelihood of errors, especially in string registers and macros).

K. ↑R Right Justify

There are times when you want a word or words to be printed so that they end at the right margin. The simplest way to do this is to type "↑R" just before the phrase you want on the right side of the page.
(example)

L. ↑C Center

This centers the following phrase (up to the next ↑R, end of line, etc.). For example, "left↑Ccenter↑Rright" produces this:
left center right
(You should surround such a line with ".br" before and after, or else do it in no-fill mode, or you will be surprised by what you get.)

III. Special Features

A. Underlining

To underline in R use "↑B" (for begin) and "↑E" (for end). This underlines the words (not spaces) between the "↑B" and the "↑E". The underlining of spaces is controlled by the built-in number register UL_SPACE -- see the R Guide or the Reference Manual for details. "Dummy" spaces (e.g., ones produced by @), because they are different from "real" spaces, will be underlined.

B. Super- and Sub-Scripts

To superscript a character in R type "↑U" (for up), then the character (or phrase) to be raised and then type "↑D" (for down) to return to the normal text line.

Similarly, to subscript a character type "↑D" (for down), then the text to be lowered and then type "↑U" (for up) to return to the normal text line.

After a trial printing you can see if the character was raised or lowered enough. It is possible to use 2 or more ups (or downs) but these must always be balanced by an equal number of the opposite command to return to the normal line level. For finer control, see the R Manual information on ↑V.

C. Margins and Line Length

The output file is organized in pages. The following figure illustrates various terms related to horizontal and vertical positioning on each output page.

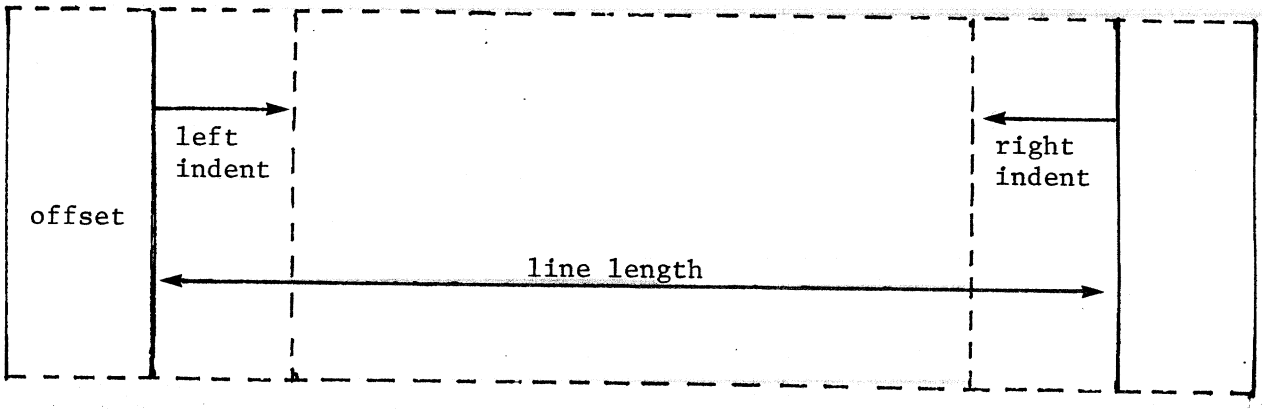
Horizontal Positioning

offset or left margin (default 1 inch)

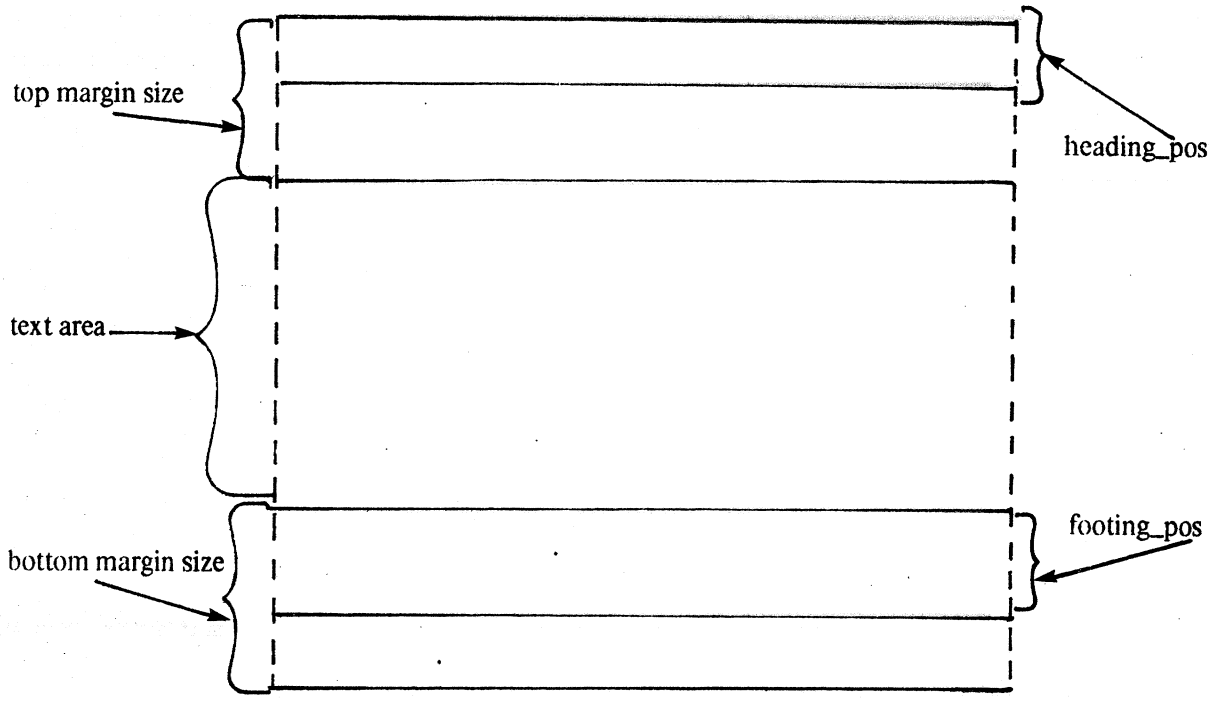
indent (measured positively inwards)

line length (default 6.5 inches), and

indent



Vertical Positioning



C. Footnotes[1]

The simplest way to create footnotes in R is to type the text until the point where the footnote is wanted. Here you type a carriage return, and type ".foot" on a line by itself. After this, type the body of the footnote. You can use as many lines as necessary. Then type ".efoot" on a line by itself. Now you just continue with the text where the footnote interrupted you. R will adjust the text printed to include the footnote on the proper page.

If R does not print the footnote number in the place where you want it in the text you can type ↑Sfn in the text where you want the footnote number to go. Then enter the footnote as described above, except use ".sfoot" in place of ".foot".

R now uses a solid line to separate footnotes from text. This is in the standard macro package. ".line" will cause R to draw a line (at any point) for the entire line length (from margin to margin). (The ".in" and ".ir" commands can be used to vary the left and right margins, respectively. Refer to the Guide or Reference Manual.)

D. Horizontal Positioning

When you wish to position a word or equation at a specific spot on the line this may be accomplished by using the command ↑P(*no.*). The number in parentheses may be in mils, inches, or character widths, etc. The examples below are set at 10, 20 and 30 character widths. Note that the distances are always measured from the margin, not the current indentation. Within a macro one more often uses ".hp" or ".nr hpos" to control horizontal positioning. The R Guide has a much more complete discussion of these issues and methods.

↑P(10)
This is the first setting.
↑P(20)
Here is the second.
↑P(30)
And the third.

---

1. This is an example of a footnote.

E. Vertical Positioning

This is usually accomplished with the ".sp" (space) command. In R the argument (value) may be in number of lines, or in inches, etc. R does not like to leave empty space at the top of a page[1]. Sometimes ".vp" is more appropriate (see the Reference Manual for more information.)

It is sometimes necessary to line up items at a uniform point within the text. See the section titled "Alignment" in the R Guide for a complete discussion. (Read about ↑Xs(.) and ↑Xp(.) (under "Logical Tab Stops") -- these seem to be the most commonly used.)

F. Indenting

The command in R for indenting from the left margin is ".in +*no.*". One must be sure to undo this command when the margin should return to normal by typing ".in -*no.*". Note that ".in *no.*" has a somewhat different effect from that of ".in +*no.*". The "+" (or "-") means to add (subtract) the specified amount from the current indentation. If there is no "+" or "-", the indentation is set to be equal to the specified value. Generally it works out better to always use the "+" and "-" forms.

One can also indent from the right margin in R. The command for this is ".ir +*no.*" (for indent right'). As with the left indent this command must also be undone to return to the normal margin by typing ".ir -*no.*".

As with ↑P, *no.* may be in inches, mils, character widths, etc.

G. Lists

The list macro in R may be invoked by the command ".list". Type the first element of the list on the line after the ".list". Use as many lines for each element as you need. Just before the second element, put ".next" on a line by itself. After the last element put ".end_list" on a line by itself (do not put a ".next" immediately before the ".end_list", or you will get extra blank space and maybe other undesirable junk).

---

1. If this is giving you problems, read up in the R manual on ".ns" and ".rs".

By default, R indents a list by 800 mils (on both right and left). You can change the values by setting the following string registers:

".sr list_left_margin *no.*"

".sr list_right_margin *no.*"

Here again the *no.* can be in mils, inches or character widths.

In addition to the left and right margin indentations, you can control a number of other things about lists. In particular, the indentation of the first line of each element can be different, the space at the beginning and end of the whole list can be varied, as well as the space between list elements, and the line spacing (".ls") to use within items. It is even possible to have the items numbered automatically. For details on adjusting these parameters, and a more complete presentation of lists in general, see the R Guide section on Lists.

EXAMPLE

1. This is the first item in the list. If it extends beyond one line this is the way R wraps the text around.

2. Here is the second item in the list. This one should also extend beyond one line so R will wrap it around as with the first.

In order to create an indented list, the ".ilist *no. no.*" command may be used. This command uses arguments to control the amount indented after the numbers and the space between items. (".list" will also respond to arguments, in a slightly different way; the Guide explains all of this completely.)

EXAMPLE

1. This is the first item to be listed. If we lengthen it so that it extends past one line R will wrap it around in this way.

2. Here is the second item in the list. Should this one also extend beyond one line, R will continue it in this fashion.

NOTE: There is a tab (↑I) after the "1." and "2." in the last example.

IV. Other Features

These features are explained in the R Guide; we mention them here so you will know they exist.

A. .begin_figure .finish_figure

These are used to produce numbered, titled figures, output automatically, without being broken in two. (See section "Figures" in the Guide.)

B. .keep and ↑Xlkeep .end_keep

The keep command tells R to output all the lines within the ".keep" and the ".end_keep" on the same page.

The "lkeep" macro is intended for equations which need to be printed on one line. That is, "lkeep" is used to make sure something stays on one line.

Both "keep" and "lkeep" are described in section "Keeping" of the Guide.

C. Suppressing headers (e.g., page numbers)

To suppress page numbers in the output, put

.nr print_headings 0

in the set-up file before ".so r.macros". Headers are normally turned off for page 1 anyway.

V. String Registers and Number Registers

These can make life easier by simplifying what needs to be typed for a complex pattern of stuff needed. They also make it possible to change this pattern of stuff by only changing the string register at the beginning of the file and thereby getting it changed uniformly throughout the file.

Example: Suppose you need to use an arrow to the left and an arrow to the right. This can be accomplished by creating string registers called "larrow" and "rarrow". Type at the beginning of the file:

".sr larrow ↑F#↑Q↑X↑F*"[1]

".sr rarrow ↑F#↑Q↑Y↑F*"

---

1. This assumes that font # is "math 12pt"

Then in the place in the text where an arrow is needed all you type is "↑Slarrow" or "↑Srarrow", and the proper arrow is printed and you can just go on with the text. If the text following "↑Slarrow" starts with a letter (A-Z, a-z), or underbar ("_"), add an "!" so R can tell where the string register name ends and the rest of the stuff begins. For example, "↑Slarrow," is ok, but "↑Slarrowa" is wrong (use "↑Slarrow!a" instead). You can use the "!" whenever you like; it never hurts.

↑F* should always be used in string register definitions, as opposed to:

".sr abc ↑F2x↑F0".

The reason is that you might put "↑Sabc" in a place where the surrounding text is not in font 0.

VI. Other Helpful Macros

It is suggested that all frequent system users create their own set-up file (stored on their own directory as "usual.rmac" or "usual rmac"). This should include the normal fonts they will want, as well as the R macros and perhaps some of the following macros. This file would be inserted on the first line of any file created as ".so usual" (or ".so usual.rmac").

A. Paragraph

Macro for paragraph set up:

```
.de para      ↑K defines paragraph setup
. sp 1l       ↑K leaves one line space
. ne 2l       ↑K must have at least two lines
. ti +5       ↑K temporary indent five (**)
. em          ↑K end
```

(**): The "+" insures a uniform added indentation of 5 spaces, even in an indented section of text.

B. Column or tab

Insert ".so tab" in the file set-up after the fonts. At the point in the text where columns are desired you type ".tab *no. no.*". The first *no.* regulates the indentation from the left margin to the first column. The second *no.* regulates the blank space between columns. On the next line, begin with ".row", and follow with the items for the first line, separated by spaces. If the item includes spaces it must be surrounded by " marks. A ".row" may be used without any items to create an empty line. You continue with ".row" for each row of items to be included in the column. End the column with ".end_tab".

Example: If you put this in your file:

```
.tab 20 5
.row item1 item2
.row a "a longer item" "a longer line"
.row ↑K a blank row
.row the last row
.end_tab
```

This is what you will get:

item1	item2	
a	a longer item	a longer line
the	last	row

C. Equations

R has a built-in macro called DIV. This takes two text arguments and outputs them one above the other, separated by a horizontal line. For more information on this macro look in the Reference Manual under Equations.

Example: If you type ↑Xdiv("the top thing" bottom), you will get this:

$$\frac{\text{the top thing}}{\text{bottom}}$$

Acknowledgments: Grateful thanks to:

Eliot Moss for the patience to explain things hundreds of times;
Berthold Horn for improving the English (German?) style;
John Guttag for suggesting the title;
Countless others for their encouragement and support.