Compile this document within R as:

```
library(knitr)
knit("testRnw.Rnw")
system("pdflatex testRnw")
```

1 R

Testing a long string:

```
b <- "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lka
```

In my test, the line overflowed the page.
A work-around for long strings:

```
b <- "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf "
tmp <- "aljdkfad kljafda kaljdf afdlkja lkajdfsa lajdfa adlfjaf jkladf afdl"
b <- paste0(b, tmp)
```

Testing long comments:

```
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad
# kljafda kaljdf afdlkja lkajdfsa lajdfa adlfjaf jkladf afdl'
```

That worked.
Testing whether blank lines are preserved in the R code:

```
tmp1 <- 7
# there should be a blank line after this

tmp3 <- 9
tmp4 <- 11
# a work around if you can't insert a blank line easily is just to put an empty
# comment line instead there should be two lines with just comment characters
# next:
#
# that creates some space, but in my Sweave+knitr test, it removed one of the
# fake blank lines while in RStudio, it kept both lines.
```

```
tmp1 = 7
# there should be a blank line after this

tmp3 = 9
tmp4 = 11
# there should be two 'blank' lines next
#
#
# now both lines are 'blank' when I use tidy=FALSE
```

Testing long code lines:

```
vecLongName <- rnorm(100)
a <- length(mean(5 * vecLongName + vecLongName - exp(vecLongName) + vecLongName,
    na.rm = TRUE))
a <- length(mean(5 * vecLongName + vecLongName))  # this is a comment that goes over the line by a good
a <- length(mean(5 * vecLongName + vecLongName - exp(vecLongName) + vecLongName,
    na.rm = TRUE))  # this is a comment that goes over the line by a good long ways long long long long
```

In my Sweave+knitr test, the code line was broken at a reasonable place, but the lines with comments and code ran over the page. In RStudio, the code line was not broken and ran over the page.

Instead, breaking a code line manually with tidy=FALSE works:

```
vecLongName = rnorm(100)
a = length(mean(5 * vecLongName + vecLongName
  - exp(vecLongName) + vecLongName, na.rm = TRUE))
a = length(mean(5 * vecLongName + vecLongName - exp(vecLongName) + vecLongName +
  exp(vecLongName), na.rm = TRUE)) # this is a comment that goes over the
  # line by a good long ways long long long long long long long
```

Testing long output:

```
print(b)

## [1] "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja ll

cat(b, fill = TRUE, width = 50)

## asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfs
## 50

cat(b, fill = 50)

## asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfs


# a kludgey work-around
printShort <- function(str, width = 80) {
    con <- textConnection(str)
    tmp <- read.fwf(con, rep(width, ceiling(nchar(str)/width)))
    tmp <- as.matrix(tmp)
    dimnames(tmp) <- NULL
    tmp[1, ]
}
printShort(b)

## [1] "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda k"
## [2] "aljdf afdlkja lkajdfsa lajdfa adlfjaf jkladf afdl"


rnorm(100)

##   [1] -1.00854  1.21207  0.04042  0.13353  0.18192 -0.80403 -0.04731  0.14821
##   [9] -0.63630 -1.54948  1.42616 -0.98755 -1.07092  0.66828  0.74304  0.18288
##  [17] -0.40092  0.09061 -0.43411  0.72408  1.27664 -1.48632 -1.54399 -0.41934
##  [25]  0.33967  1.41328  1.39256 -0.44976 -0.08819 -1.43182  3.10852  0.06076
##  [33] -0.27664 -0.06719 -1.52950  2.02147  0.21764 -0.61937  0.15134  1.97918
```

```
##  [41] -0.41040  0.86834 -2.26857  0.38543  0.71735  0.37902  0.70982  1.15534
##  [49] -0.10547  0.64079 -1.28508 -0.54305 -0.16376 -0.91509 -0.61342 -0.82638
##  [57]  1.94037 -0.19691  0.73802  0.44185  1.87195  0.19459 -0.24219 -1.45513
##  [65] -1.44949 -0.08811  0.05555 -0.21288  0.90275  0.66896  0.07417  1.83812
##  [73] -1.11572 -0.07967 -1.22591  0.86576 -1.13026 -1.13451 -0.42150  0.47858
##  [81] -2.67315 -1.25693 -0.60152 -0.54863  0.73832  0.27468 -0.96949 -0.90659
##  [89] -1.05970 -0.40141 -1.43651 -0.05133  1.13888 -0.19635  3.14123 -0.96021
##  [97]  1.91570  1.01417  0.96804  1.35965
```

In my test, long strings run off the page, but numerical output does not.

Testing whether there are extra lines inserted when a chunk is indented in an enumerate/list and not read in from a .R file:

1. An item

   ```
   a <- 5
   b <- 7
   c <- 9
   ```

2. Second item

That works fine in straight Sweave+knitr, but when I did it within Lyx, extra blank lines were inserted between all code lines when the chunk was within an enumeration. RStudio had trouble with the chunks.

Having the chunk get its code from an R file (in Sweave+knitr or Lyx+knitr) worked fine though:

1. An item

   ```
   d <- 9
   e <- 11
   f <- 12
   ```

2. Second item

Using the Latex listings package

A note on listings. If we have the

```
\lstset
```

command in the preamble uncommented and we put

```
render_listings()
```

in the setup chunk, then long text strings will break appropriately. However I'm not a big fan of the colors and fonts and assignments get converted to arrows that one cannot cut and paste into R. But as Matt points out on Piazza you can monkey with the fonts in the Sweavel.sty file that is created after you knit the document for the first time. This link includes a list of some of the settings that you can change: `http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings#Using_the_listings_package`.

2 bash

Now in bash, we have similar problems with line endings but bash allows us to use a backslash to break lines of code. The one thing that doesn't help us with is long lines of output. Also, we need to break long comments ourselves. And note that all results are printed at the end of a code chunk instead of interspersed immediately after the command generating the output.

```
echo "my long line blah blah blah blah more blah more blah more blah blah blah blash blah blah blah of s

echo "contents of tmp.txt blah blah blah blah \
more blah more blah more blah \
blah blah blash blah blah blah \
more more more more more more \
more more more more" > tmp.txt

echo "my long line"\
> tmp2.txt

cat tmp.txt

# the following long comment line is not broken in my test:
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfsa

# instead manually break it:
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla
# lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfsa lajdfa
# adlfjaf jkladf afdl
## contents of tmp.txt blah blah blah blah more blah more blah more blah blah blah blash blah blah blah
```

None of the bash code worked in RStudio.

3 Python

In my Sweave+knitr test it only shows Python code evaluation results when you use an explicit print statement. Line breaks are still an issue but manual breaking of long code lines seems to work. As with bash, all output is printed at the end of the chunk.

```
2 + 2
print(2 + 2)
type(2.0)

endLine = "."
name = "sam"
if name == 'samuel':
  print(intro + name.capitalize() + endLine)
elif name == "sam":
  print("My nickname is " + name.capitalize() + endLine)
else:
  print("My name is something else.\n")

# this overflows the page
b = "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkaj
print(b)

# this code overflows the page
zoo = {"lion": "Simba", "panda": None, "whale": "Moby", "numAnimals": 3, "bear": "Yogi", "killer whale"
print(zoo)

# instead manually break the code
zoo = {"lion": "Simba", "panda": None, "whale": "Moby",
```

```
       "numAnimals": 3, "bear": "Yogi", "killer whale": "shamu",
       "bunny:": "bugs"}
print(zoo)
## 4
## My nickname is Sam.
## asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdf
## {'killer whale': 'shamu', 'lion': 'Simba', 'bunny:': 'bugs', 'bear': 'Yogi', 'numAnimals': 3, 'whale
## {'killer whale': 'shamu', 'lion': 'Simba', 'bunny:': 'bugs', 'bear': 'Yogi', 'numAnimals': 3, 'whale
```

None of the Python code worked in RStudio.