

Lucid printing

Kevin Wright

December 15, 2014

1 Abstract

The **lucid** package provides a method for pretty-printing vectors of floating point numbers, with special application to printing of variance components from mixed models.

2 Intro

Numerical output from R is often in scientific notation, which can make it difficult to quickly glance at numbers and understand the relative sizes of the numbers. This not a new phenomenon. Before R had been created, ([Finney, 1988](#), 351-352) had this to say about numerical output:

Certainly, in initiating analyses by standard software or in writing one's own software, the aim should be to have output that is easy to read and easily intelligible to others. ... Especially undesirable is the so-called 'scientific notation' for numbers in which every number is shown as a value between 0.0 and 1.0 with a power of 10 by which it must be multiplied. For example:

```
0.1234E00 is 0.1234
0.1234E02 is 12.34
0.1234E-1 is 0.01234
```

This is an abomination which obscures the comparison of related quantities; tables of means or of analyses of variance become very difficult to read. It is acceptable as a default when a value is unexpectedly very much larger or smaller than its companions, but its appearance as standard output denotes either lazy programming or failure to use good software properly. Like avoidance of 'E', neat arrangement of output values in columns, with decimal points on a vertical line, requires extra effort by a programmer but should be almost mandatory for any software that is to be used often.

One recommendation for improving the display of tables of numbers is to round numbers a lot ([Wainer, 1997](#)). Humans cannot understand more than two digits very easily. It is rare that more than two digits of accuracy can be justified, and even if we could justify more than two digits, we seldom care about more than two digits. In R, using the **round** and **signif** functions can help improve numerical output, but can still print results in scientific notation and leave much to be desired. The **lucid** package provides functions to improve the presentation of floating point numbers in a way that makes interpretation of the numbers **immediately** apparent.

Consider the following vector of coefficients from a fitted model:

```
##          effect
## A      -1.350000e+01
## B       4.500000e+00
## C       2.450000e+01
## C1      6.927792e-14
## C2     -1.750000e+00
## D       1.650000e+01
## (Intercept) 1.135000e+02
```

Which coefficient is basically zero? How large is the intercept?

Both questions can be answered using the output shown above, but it takes too much effort to answer the questions. Now examine the same vector of coefficients with prettier formatting:

```
require("lucid")
options(digits=7) # knitr defaults to 4, R console uses 7
lucid(df1)

##          effect
## A      -13.5
## B       4.5
## C      24.5
## C1       0
## C2     -1.75
## D      16.5
## (Intercept) 114
```

Which coefficient is basically zero? How large is the intercept?

Printing the numbers with the `lucid` function has made the questions much easier to answer.

The sequence of steps used by `lucid` to format and print the output is.

1. Zap to zero
2. Round using significant digits
3. Drop trailing zeros
4. Align numbers at the decimal point

The `lucid` package contains a generic function `lucid` with specific methods for numeric vectors, matrices, lists and data frames. The methods for matrices and data frames apply formatting to each numeric column and leave other columns unchanged.

3 Example: Antibiotic effectiveness

[Wainer and Larsen \(2009\)](#) present data published by Will Burtin in 1951 on the effectiveness of antibiotics against 16 types of bacteria. The data is included in the `lucid` package as a dataframe called `antibiotic`. The default view of this data is:

```
print(antibiotic)
```

	bacteria	penicillin	streptomycin	neomycin	gramstain
## 1	Aerobacter aerogenes	870.000	1.00	1.600	neg
## 2	Brucella abortus	1.000	2.00	0.020	neg
## 3	Brucella antracis	0.001	0.01	0.007	pos
## 4	Diplococcus pneumoniae	0.005	11.00	10.000	pos
## 5	Escherichia coli	100.000	0.40	0.100	neg
## 6	Klebsiella pneumoniae	850.000	1.20	1.000	neg
## 7	Mycobacterium tuberculosis	800.000	5.00	2.000	neg
## 8	Proteus vulgaris	3.000	0.10	0.100	neg
## 9	Pseudomonas aeruginosa	850.000	2.00	0.400	neg
## 10	Salmonella (Eberthella) typhosa	1.000	0.40	0.008	neg
## 11	Salmonella schottmuelleri	10.000	0.80	0.090	neg
## 12	Staphylococcus albus	0.007	0.10	0.001	pos
## 13	Staphylococcus aureus	0.030	0.03	0.001	pos
## 14	Streptococcus fecalis	1.000	1.00	0.100	pos
## 15	Streptococcus hemolyticus	0.001	14.00	10.000	pos
## 16	Streptococcus viridans	0.005	10.00	40.000	pos

Due to the wide range in magnitude of the values, nearly half of the floating-point numbers in the default view contain trailing zeros after the decimal, which adds significant clutter and impedes interpretation. The **lucid** display of the data is:

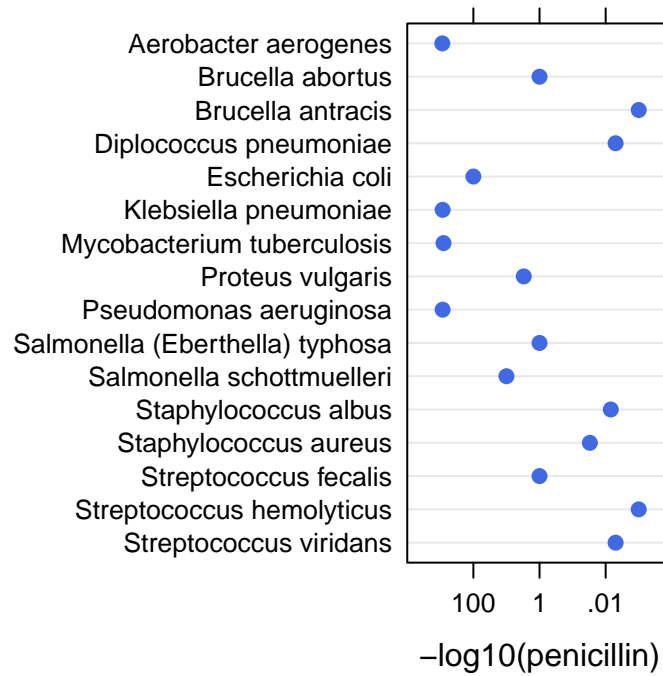
```
lucid(antibiotic)
```

	bacteria	penicillin	streptomycin	neomycin	gramstain
## 1	Aerobacter aerogenes	870	1	1.6	neg
## 2	Brucella abortus	1	2	0.02	neg
## 3	Brucella antracis	0.001	0.01	0.007	pos
## 4	Diplococcus pneumoniae	0.005	11	10	pos
## 5	Escherichia coli	100	0.4	0.1	neg
## 6	Klebsiella pneumoniae	850	1.2	1	neg
## 7	Mycobacterium tuberculosis	800	5	2	neg
## 8	Proteus vulgaris	3	0.1	0.1	neg
## 9	Pseudomonas aeruginosa	850	2	0.4	neg
## 10	Salmonella (Eberthella) typhosa	1	0.4	0.008	neg
## 11	Salmonella schottmuelleri	10	0.8	0.09	neg
## 12	Staphylococcus albus	0.007	0.1	0.001	pos
## 13	Staphylococcus aureus	0.03	0.03	0.001	pos
## 14	Streptococcus fecalis	1	1	0.1	pos
## 15	Streptococcus hemolyticus	0.001	14	10	pos
## 16	Streptococcus viridans	0.005	10	40	pos

The **lucid** display is dramatically simplified, providing a much clearer picture of the effectiveness of the antibiotics against bacteria. This view of the data matches exactly the appearance of Table 1 in [Wainer and Larsen \(2009\)](#).

A stem-and-leaf plot is a semi-graphical display of data, in that the *positions* of the numbers create a display

similar to a histogram. In a similar manner, the **lucid** output is a semi-graphical view of the data. The figure below shows a dotplot of the penicillin values on a reverse log10 scale. Note the similarity in the overall shape of the positions of the left-most significant digit in the penicillin column of the output and the dotplot. Also, as noted by Gelman (2011), the amount of ink in printing the significant digit has a surprisingly large correlation with the value of the digit, increasing the information in the semi-graphical view from **lucid** printing.



4 Application to mixed models

During the process of iterative fitting of mixed models, it is often useful to compare fits of different models to data, for example using loglikelihood or AIC values, or with the help of residual plots. It can also be very informative to inspect the variance components. The **lucid** package provides a function called **vc** that makes it easy to extract the estimated variances and correlations from fitted models and prints them in friendly format using the **lucid** function.

The **vc** function has methods that can be used with the **nlme** (Pinheiro et al., 2014), **lme4** (Bates et al., 2014), and **asreml** (Butler, 2009) packages. The **VarCorr** function can be used to similar effect with the **nlme** and **lme4** packages, but it shows variances for **nlme** models and standard deviations for **lme4** models. The **VarCorr** function is not available for the **asreml** package. The **vc** function provides a unified interface for extracting the variance components from fitted models and prints the results using **lucid**.

Pearce et al. (1988) suggest showing four significant digits for the error mean square and two decimal places digits for F values. The **lucid** function uses a similar philosophy, presenting the variances with four significant digits and **asreml** Z statistics with two significant digits.

The following simple example illustrates use of the **vc** function.

```

require("nlme")
data(Rail)
mn <- lme(travel~1, random=~1|Rail, data=Rail)
vc(mn)

##          effect variance stddev
## (Intercept)   615.3   24.81
##   Residual    16.17   4.021

require("lme4")
m4 <- lmer(travel~1 + (1|Rail), data=Rail)
vc(m4)

##      grp      var1 var2  vcov  sdcor
##   Rail (Intercept) <NA> 615.3  24.81
##  Residual          <NA> <NA> 16.17  4.021

#require("asreml")
#ma <- asreml(travel~1, random=~Rail, data=Rail)
#vc(ma)

##          effect component std.error z.ratio constr
## Rail!Rail.var    615.3      392.6     1.6    pos
##   R!variance    16.17       6.6     2.4    pos

```

A second example is more complex. The example is no longer reproducible due to changes in the mixed-models software, so only the output is shown. A single model was fit to data using two different optimization methods. The goal was to compare the results from the two optimizers. In the output below, the first two columns identify terms in the model, the next two columns are the variance and standard deviation from one optimizer, while the final two columns are from the other optimizer.

The default printing is shown first.

```

cbind(d1,d2[,3:4])

##      Groups      Name  Variance Std.Dev.  Variance  Std.Dev.
## 1 new.gen (Intercept)  2869.45  53.567  3.23e+03  56.818310
## 2      one      r1:c3  5531.60  74.375  7.69e+03  87.675211
## 3   one.1      r1:c2  58225.75 241.300  6.98e+04 264.123506
## 4   one.2      r1:c1 128003.60 357.776  1.07e+05 327.750047
## 5   one.3         c8   6455.77  80.348  6.79e+03  82.381314
## 6   one.4         c6   1399.73  37.413  1.64e+03  40.446339
## 7   one.5         c4   1791.65  42.328  1.23e+04 110.764195
## 8   one.6         c3   2548.89  50.486  2.69e+03  51.831045
## 9   one.7         c2   5941.80  77.083  7.64e+03  87.435345
## 10  one.8         c1      0.00   0.000  9.56e-04   0.030918
## 11  one.9        r10   1132.95  33.659  1.98e+03  44.446766
## 12 one.10         r8   1355.23  36.813  1.24e+03  35.234043
## 13 one.11         r4   2268.73  47.631  2.81e+03  53.020431
## 14 one.12         r2    241.79  15.550  9.28e+02  30.465617
## 15 one.13         r1   9199.94  95.916  1.04e+04 101.802960

```

```
## 16 Residual          4412.11    66.424 4.13e+03   64.240858
```

Do the two optimization methods give similar results? It is difficult to compare the results due to the clutter of extra digits, and even more importantly, because of a quirk in the way R formats the output. The variances in column 3 are shown in non-scientific format, while the variances in column 5 are shown in scientific format!

The `lucid` function is now used to show the results.

```
lucid(cbind(d1,d2[,3:4]))
```

##	Groups	Name	Variance	Std.Dev.	Variance	Std.Dev.
## 1	new.gen	(Intercept)	2870	53.6	3230	56.8
## 2	one	r1:c3	5530	74.4	7690	87.7
## 3	one.1	r1:c2	58200	241	69800	264
## 4	one.2	r1:c1	128000	358	107000	328
## 5	one.3	c8	6460	80.3	6790	82.4
## 6	one.4	c6	1400	37.4	1640	40.4
## 7	one.5	c4	1790	42.3	12300	111
## 8	one.6	c3	2550	50.5	2690	51.8
## 9	one.7	c2	5940	77.1	7640	87.4
## 10	one.8	c1	0	0	0	0.0309
## 11	one.9	r10	1130	33.7	1980	44.4
## 12	one.10	r8	1360	36.8	1240	35.2
## 13	one.11	r4	2270	47.6	2810	53
## 14	one.12	r2	242	15.6	928	30.5
## 15	one.13	r1	9200	95.9	10400	102
## 16	Residual		4410	66.4	4130	64.2

The formatting of the variance columns is now consistent and simplified with fewer digits shown. It is easy to compare the columns and see that the two optimizers are giving quite different answers.

5 Appendix

Session information:

- R version 3.1.2 (2014-10-31), x86_64-w64-mingw32
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: knitr 1.8, lattice 0.20-29, lme4 1.1-7, lucid 1.1, Matrix 1.1-4, nlme 3.1-118, Rcpp 0.11.3
- Loaded via a namespace (and not attached): evaluate 0.5.5, formatR 1.0, grid 3.1.2, highr 0.4, MASS 7.3-35, minqa 1.2.4, nloptr 1.0.4, splines 3.1.2, stringr 0.6.2, tools 3.1.2

References

D Bates, M Maechler, B Bolker, and S Walker. *lme4: Linear mixed-effects models using Eigen and S4*, 2014. URL <http://CRAN.R-project.org/package=lme4>. R package version 1.1-7.

- David Butler. *asreml: asreml() fits the linear mixed model*, 2009. URL www.vsni.co.uk. R package version 3.0.
- D. J. Finney. Was this in your statistics textbook? II. Data handling. *Experimental agriculture*, 24:343–353, 1988.
- Andrew Gelman. Tables as graphs: the Ramanujan principle. *Significance*, 8:183–183, 2011.
- S. C. Pearce, G. M. Clarke, G. V. Dyke, and R. E. Kempson. *A Manual of Crop Experimentation*. Charles Griffin and Company, 1988.
- Jose Pinheiro, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2014. URL <http://CRAN.R-project.org/package=nlme>. R package version 3.1-118.
- Howard Wainer. Improving tabular displays, with NAEP tables as examples and inspirations. *Journal of Educational and Behavioral Statistics*, 22(1):1–30, 1997. doi: 10.3102/10769986022001001.
- Howard Wainer and Mike Larsen. Pictures at an exhibition. *Chance*, 22(2):46–54, 2009. doi: 10.1080/09332480.2009.10722958. URL <http://chance.amstat.org/2009/04/visrev222/>.