Common Numerical Issues in Statistical Computing.

Robin J. Evans
www.stats.ox.ac.uk/∼evans

Michaelmas 2017

Algorithmic Considerations

Some methods of computing things are easier than others.

Multiplying $n \times m$ matrix by $m \times k$ matrix is $O(nmk)$ calculations.

Let A, B be $n \times n$ matrices and c be an $n \times 1$ vector.

$$ABc = (AB)c = A(Bc)$$

But $(AB)c$ takes $O(n^3 + n^2) = O(n^3)$, and $A(Bc)$ takes $O(n^2 + n^2) = O(n^2)$.

Not all Code is Created Equal

```
A = matrix(rnorm(1e4), 100, 100); B = matrix(rnorm(1e4), 100, 100)
c = rnorm(100)

library(microbenchmark)
microbenchmark(A %*% B %*% c,  A %*% (B %*% c),  times=100)

## Unit: microseconds
##              expr     min       lq      mean    median
##     A %*% B %*% c 656.153 722.571 915.37622 814.2185
##   A %*% (B %*% c)  22.532  23.476  31.00104  26.8315
##         uq      max neval
##   953.1395 3084.564   100
##    30.1225   82.958   100
```

R evaluates left to right in this case.

Numerical Stability Considerations

Floating point numbers have a limited accuracy (usually around 10^{-16} for an $O(1)$ number).

```
0.3 - 0.2 - 0.1

## [1] -2.775558e-17
```

```
summary(A %*% B %*% c - A %*% (B %*% c))

##        V1
##  Min.   :-1.421e-13
##  1st Qu.:-4.408e-14
##  Median : 0.000e+00
##  Mean   :-2.442e-17
##  3rd Qu.: 4.263e-14
##  Max.   : 1.279e-13
```

Problems of numerical accuracy can be solved with long doubles in languages like C.

Arithmetic Precision

R may hide some of these rounding issues, so don't forget that they exist!

```
1+1e-15

## [1] 1

print(1+1e-15, digits=22)

## [1] 1.000000000000001110223
```

R also has an integer type (but it's a bit tricky)

```
1 == 1L

## [1] TRUE

identical(1, 1L)

## [1] FALSE
```

Arithmetic Precision

Equality testing may be problematic with floating point numbers:

```
x = 1+1e-15
x == 1

## [1] FALSE


all.equal(x, 1)

## [1] TRUE
```

`all.equal()` ignores small differences in numbers, and also their type.

```
all.equal(1L, 1)

## [1] TRUE
```

Numerical Stability Considerations

Floats also have an upper and lower limits on the numbers they can hold

```
c(2^-1074, 2^-1075)

## [1] 4.940656e-324  0.000000e+00


c(2^1023, 2^1024)

## [1] 8.988466e+307           Inf
```

You may need to think carefully about the way in which you compute things

```
c(2^(2000-1993), 2^2000/2^1993)

## [1] 128 NaN
```

Numerical Stability Considerations

Additive computations are generally much more stable than multiplicative ones. Suppose you want to calculate the geometric mean of some numbers

```
set.seed(324)
geomean = function(x) prod(x)^(1/length(x))
x = rlnorm(1e3, meanlog=-1)  # log-normals
geomean(x)

## [1] 0


range(x)

## [1] 0.01134218 9.73558109
```

If we do everything on a log-scale, there's no problem

```
geomean2 = function(x) exp(mean(log(x)))
geomean2(x)   # approximately exp(-1)

## [1] 0.3597272
```