Power and precision with simulation

SDC
January 2013
Version 1

Compiled Tuesday 22nd January, 2013, 18:30
from: `C:/stat/R/BxC/Examples/pwrprc.tex`

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
`bxc@steno.dk`
http://BendixCarstensen.com

Contents

1 Simple simulation set-up 1

2 Power or precision? 4

References 4

1 Simple simulation set-up

In the following we set up a simulation of 2 by 2 tables, computing the RR, the error factor (that is the multiplier for the RR that yields the 95 % c.c.) and the p-value for testing the null of RR=1:

```
> simp <- function( p0, p1, n1, fac=2, N=1000, alpha=0.05 )
+   {
+     n0 = fac*n1
+     s1 <- rbinom( N, n1, p1 )
+     s0 <- rbinom( N, n0, p0 )
+     rr <- (s1/n1)/(s0/n0)
+     vn <- 1/s1 - 1/n1 + 1/s0 - 1/n0
+     pval <- 1 - pchisq( log(rr)^2/vn, 1 )
+     c( mean( pval < alpha ),
+        exp( 1.96*sqrt(mean(vn)) ) )
+   }
```

In the function `fac` is ratio of unexposed (`n0`) to exposed (`n1`) persons, `p0` and `p1` the disease probabilities in the two groups. The function returns the power and the average precision, based on `N` simulations

Once we have that function we just make loops over the relevant scenarios, the RR associated with exposure, the disease probabilities (`pr`), and the number of exposed cases `nc`:

```
> RR <- c(1.1,1.2,1.5,2)
> p0 <- (1:25)/ 100
> nc <- c(500,1000,2000,5000)
```

Once these values are decided on, we set up an array to hold the results; note that the array is defined entirely from `RR`, `p0` and `nc`, so the whole thing can be fine-tuned by just tampering with the lines above, most likely to cover more relevant scenarios.

```
> dnam = list( RR=RR, p0=p0, N.cases=nc, What=c("power","erf") )
> pwpr <- array( NA, dim=sapply( dnam, length ), dimnames=dnam )
> str( pwpr )
```

```
 logi [1:4, 1:25, 1:4, 1:2] NA NA NA NA NA NA ...
 - attr(*, "dimnames")=List of 4
  ..$ RR     : chr [1:4] "1.1" "1.2" "1.5" "2"
  ..$ p0     : chr [1:25] "0.01" "0.02" "0.03" "0.04" ...
  ..$ N.cases: chr [1:4] "500" "1000" "2000" "5000"
  ..$ What   : chr [1:2] "power" "erf"
```

Note that inside the loop, we just use the function that generates `N` data sets and extracts the power and precision. Note the `system.time` command that gives you a handle on how long it takes — and allows you to tune the sizes of your array to fit an over-night run.

```
> system.time(
+ for( r in 1:length(RR) )
+ for( p in 1:length(p0) )
+ for( n in 1:length(nc) )
+ pwpr[r,p,n,] <- simp( p0=p0[p], p1=p0[p]*RR[r], fac=1, n1=nc[n], N=10000 )
+             )
```

```
   user  system elapsed
   4.88    0.02    4.96
```

Once we have collected the output, show a table of the results:

```
> round( ftable( pwpr[,1:8*3,,], row.vars=1:2, col.vars=3:4 ), 2 )
```

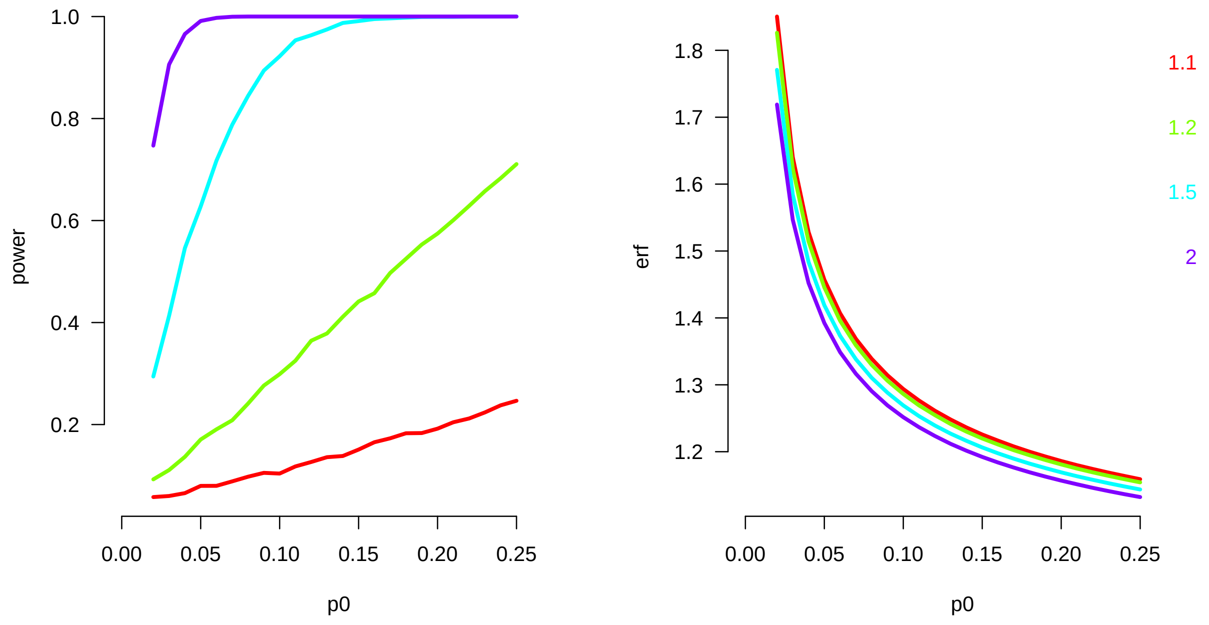
```
         N.cases   500        1000        2000        5000
         What     power  erf power  erf power  erf power  erf
RR  p0
1.1 0.03           0.05 2.04  0.06 1.64  0.08 1.41  0.14 1.24
    0.06           0.06 1.63  0.08 1.41  0.12 1.27  0.23 1.16
    0.09           0.07 1.47  0.11 1.31  0.16 1.21  0.33 1.13
    0.12           0.09 1.39  0.13 1.26  0.21 1.18  0.44 1.11
    0.15           0.10 1.34  0.15 1.23  0.25 1.15  0.53 1.10
    0.18           0.11 1.30  0.18 1.20  0.30 1.14  0.63 1.08
    0.21           0.12 1.26  0.20 1.18  0.36 1.12  0.73 1.08
    0.24           0.15 1.24  0.24 1.16  0.42 1.11  0.79 1.07
1.2 0.03           0.07 2.01  0.11 1.62  0.18 1.40  0.39 1.24
    0.06           0.12 1.61  0.19 1.40  0.33 1.27  0.68 1.16
    0.09           0.16 1.46  0.28 1.31  0.48 1.21  0.85 1.13
    0.12           0.20 1.38  0.36 1.25  0.61 1.17  0.94 1.11
    0.15           0.24 1.33  0.44 1.22  0.72 1.15  0.98 1.09
    0.18           0.30 1.29  0.52 1.19  0.82 1.13  0.99 1.08
    0.21           0.36 1.26  0.60 1.17  0.88 1.12  1.00 1.07
    0.24           0.40 1.23  0.68 1.16  0.93 1.11  1.00 1.07
1.5 0.03           0.22 1.94  0.41 1.58  0.70 1.38  0.98 1.23
    0.06           0.42 1.57  0.72 1.37  0.95 1.25  1.00 1.15
    0.09           0.61 1.43  0.89 1.29  0.99 1.20  1.00 1.12
    0.12           0.75 1.36  0.96 1.24  1.00 1.16  1.00 1.10
    0.15           0.86 1.30  0.99 1.21  1.00 1.14  1.00 1.09
    0.18           0.93 1.27  1.00 1.18  1.00 1.13  1.00 1.08
    0.21           0.97 1.24  1.00 1.16  1.00 1.11  1.00 1.07
    0.24           0.99 1.22  1.00 1.15  1.00 1.10  1.00 1.06
2   0.03           0.61 1.87  0.91 1.55  1.00 1.36  1.00 1.21
    0.06           0.92 1.53  1.00 1.35  1.00 1.23  1.00 1.14
    0.09           0.99 1.40  1.00 1.27  1.00 1.18  1.00 1.11
    0.12           1.00 1.33  1.00 1.22  1.00 1.15  1.00 1.09
    0.15           1.00 1.28  1.00 1.19  1.00 1.13  1.00 1.08
    0.18           1.00 1.25  1.00 1.17  1.00 1.12  1.00 1.07
    0.21           1.00 1.22  1.00 1.15  1.00 1.10  1.00 1.06
    0.24           1.00 1.20  1.00 1.14  1.00 1.09  1.00 1.06
```

If you are interested in how precision depends on disease probability and RR, you can expand these and graph select parts of the resulting array, for example for `nc=1000`, showing how the power resp. precision depends on p_0 for different RRs:

```
> lR <- length(RR)
> clr <- rainbow( lR )
> ( wR <- (2*lR-1:lR)/(2*lR) )
```

```
[1] 0.875 0.750 0.625 0.500
```

```
> par(mfrow=c(1,2))
> for( wh in dimnames(pwpr)[[4]] )
+ matplot( p0, t(pwpr[,,"1000",wh]),
+          type="l", lty=1, lwd=3, col=clr,
+          ylab=wh, xlim=c(0,max(p0)*1.1), bty="n", las=1 )
> text( par("usr")[rep(2,lR)],
+       par("usr")[4]*wR + par("usr")[3]*(1-wR),
+       dimnames(pwpr)[[1]], adj=1, col=clr )
```

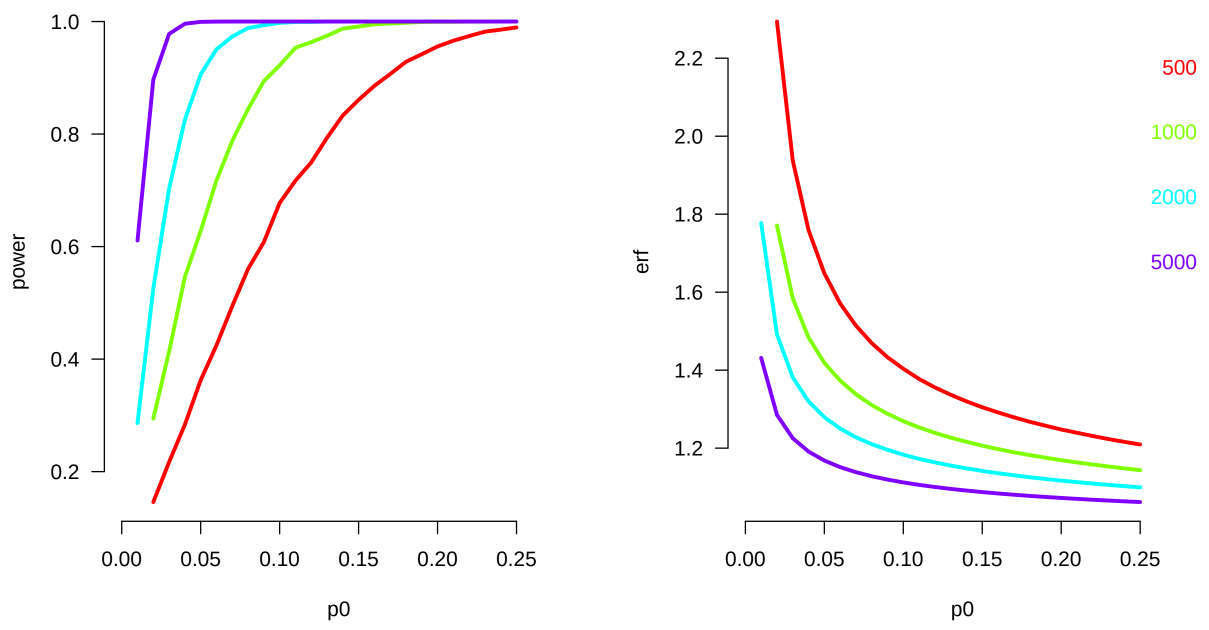


Likewise we can show how (for RR=1.5) power and precisoion depend on N:

```
> lN <- length(nc)
> clr <- rainbow( lN )
> ( wN <- (2*lN-1:lN)/(2*lN) )
```

```
[1] 0.875 0.750 0.625 0.500
```

```
> par(mfrow=c(1,2))
> for( wh in dimnames(pwpr)[[4]] )
+ matplot( p0, pwpr["1.5",,,wh],
+          type="l", lty=1, lwd=3, col=clr,
+          ylab=wh, xlim=c(0,max(p0)*1.1), bty="n", las=1 )
> text( par("usr")[rep(2,lR)],
+       par("usr")[4]*wR + par("usr")[3]*(1-wR),
+       dimnames(pwpr)[[3]], adj=1, col=clr )
```



Note the power of R here: the plotting works even if you change the range of RRs or probabilities. This can also be achieved by Stata or SAS, but hardly in finite programming time.

2 Power or precision?

Note from the figure above, that the power depends both on the sample size and on the probability of event and from a certain level of probability is virtually 1, whereas the precision from a certain level is pretty flat, and only depends on the sample size.

This is because the power mixes in the actual effect size (or more precisely the distance to the null), which is not very sensible; the interesting question in planning science is normally how precisely you determine you quantities, not how close to the null they are.

If you have a good precision (small s.e.) then a non-significant result is informative: There is evidence that there is no effect. And a significant result might even tell you the same: there is a detectable effect, but too small to be of any relevance.

If you have a bad precision (large s.e.) a non-significant result tells you nothing. And a significant result might tell you very little too: either the effct is too small to bother about or it is huge.

Hence the precision (as for example the width of a confidence interval) should be the target quantity in sample size calculations.

So maybe grants should be denied researchers that put forward power calculations; by doing so they demonstrate that they have no clue of quantitative science.