Bayes Factors & the BIC Statistic

John Maindonald

2024-06-29

Kahneman (2011) makes the point that humans are poor intuitive statisticians. This is an especially serious issue for understanding and using p-values, for the choice of priors for Bayesian analyses, and for the use of information statistics. Observing how these various statistics compare for simulated data, and how they relate to one another, is helpful for the development of intuition.

Here, the focus will be on the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC), and on the connection that can be made, more directly for the BIC than for the AIC, to a form of Bayes Factor.

For purposes of making a connection to the BIC and AIC, the focus will be on Bayes Factors as returned by functions in the *Bayesfactor* package and, by `BPpack::BF()`.

All summary statistics are random variables

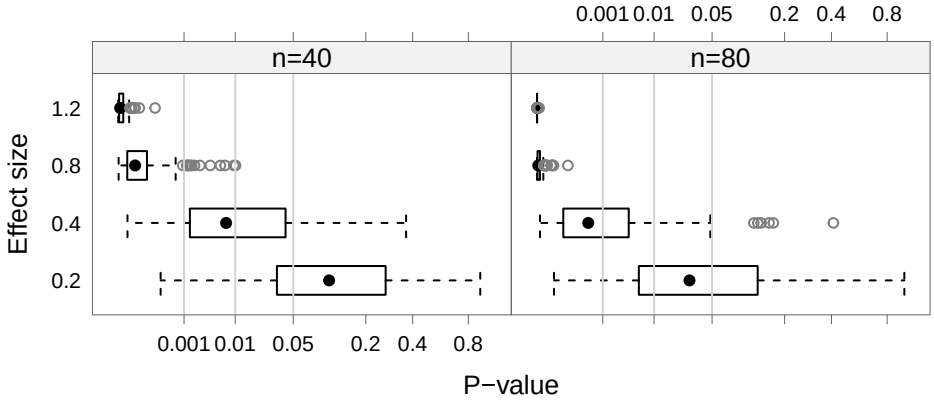

Figure 1: Boxplots are for 200 simulated p-values for a one-sided - one-sample t-test, for the specified effect sizes `eff` and - sample sizes `n`. The $p^{0.25}$ scale on the x-axis is used - to reduce the extreme asymmetry in the distributions.

Note first that all these statistics, as well as p-values, are random variables. The randomness is a particular issue when sample sizes are small and/or effect sizes are small. Figure 1 highlights this point. Code is:

```
eff2stat <- function(eff=c(.2,.4,.8,1.2), n=c(40,160), numreps=200,
                     FUN=function(x,N)pt(sqrt(N)*mean(x)/sd(x), df=N-1,
                                         lower.tail=FALSE)){
  simStat <- function(eff=c(.2,.4,.8,1.2), N=10, nrep=200, FUN){
    num <- N*nrep*length(eff)
    array(rnorm(num, mean=eff), dim=c(length(eff),nrep,N)) |>
      apply(2:1, FUN, N=N)
  }
  mat <- matrix(nrow=numreps*length(eff),ncol=length(n))
  for(j in 1:length(n)) mat[,j] <-
    as.vector(simStat(eff, N=n[j], numreps, FUN=FUN))  ## length(eff)*numep
  data.frame(effsize=rep(rep(eff, each=numreps), length(n)),
             N=rep(n, each=numreps*length(eff)), stat=as.vector(mat))
}
```

```
library(lattice)
set.seed(31)
n <- c (40,80)
df200 <- eff2stat(eff=c(.2,.4,.8,1.2), n=n, numreps=200)
labx <- c(0.001,0.01,0.05,0.2,0.4,0.8)
gph <- bwplot(factor(effsize) ~ I(stat^0.25) | factor(N), data=df200,
              layout=c(2,1), xlab="P-value", ylab="Effect size",
              scales=list(x=list(at=labx^0.25, labels =labx)))
update(gph+latticeExtra::layer(panel.abline(v=labx[1:3]^0.25, col='lightgray')),
       strip=strip.custom(factor.levels=paste0("n=", n)),
       par.settings=DAAG::DAAGtheme(color=F, col.points="gray50"))
```

The BIC and AIC connection to Bayes Factors

As a starting point, comparisons will be for a one-sample t-statistic.

Given two models with the same outcome variable, with respective BIC (Bayesian Information Criterion) statistics m_1 and m_2, the quantity

$$b_{12} = exp((m_1 - m_2)/2)$$

can be used as a relative preference statistic for m_2 as against m_1. If model 1 is nested in model 2 this becomes, under the prior that has the name Jeffreys Unit Information (JUI) prior,

with the prior centered on the maximum likelihood of the difference under the alternative, a Bayes Factor giving the probability of model 2 relative to model 1. In the case of a one-sample t-statistic, the BIC-derived Bayes Factor is

$$\exp(N * \log(1 + \frac{t^2}{N-1}) - \log(N))/2), \text{ where } N \text{ is the sample size}$$

How does this compare with Bayes Factors that are obtained with other choices of prior? Specifically, because the calculations can then be handled without Markov Chain Monte Carlo simulation, we look at results from functions in the *Bayesfactor* and *BFpack* packages.

Comparison with results from `BayesFactor::ttestBF()`

Functions in the *BayesFactor* package assume a Jeffreys-Zellner-Siow (JSZ) prior, which has a reasonable claim to be used as a default prior. Numerical quadrature is used to calculate the Bayes Factor, avoiding the need for Markov Chain Monte Carlo simulation. A Cauchy prior is assumed for the effect size, with the argument `rscale` giving the scale factor. The Jeffreys distribution has a similar role for the variance of the normal distributions that are assumed both under the null and under the alternative.

```
# Functions that calculate Bayes Factors or relative preferences
t2BF <- function(p=0.05, N, xrs=1/sqrt(2)){
  t <- qt(p/2, df=N-1, lower.tail=FALSE)
  BayesFactor::ttest.tstat(t=t, n1=N, rscale=xrs, simple=TRUE)}
t2BFbic <- function(p,N){t <- qt(p/2, df=N-1, lower.tail=FALSE)
  exp((N*log(1+t^2/(N-1))-log(N))/2)}
t2AIC <- function(p,N){t <- qt(p/2, df=N-1, lower.tail=FALSE)
  exp((N*log(1+t^2/(N-1))-2)/2)}
t2AICc <- function(p,N){t <- qt(p/2, df=N-1, lower.tail=FALSE)
  exp((N*log(1+t^2/(N-1))-12/(N-3)+4/(N-2)-2)/2)}  ## Requires N > 6
t2eff <- function(p=0.05, N)
  eff <- qt(p/2, df=N-1, lower.tail=FALSE)/sqrt(N)
```

```
pval <- c(.05,.01,.001); np <- length(pval)
Nval <- c(3,4,5,10,20,40,80,160,360); nlen <- length(Nval)
## Bayes Factors, using BayesFactor::ttest.tstat()
rs <- c(1/sqrt(2), sqrt(2))
bf <- matrix(nrow=length(rs)+2,ncol=length(Nval))
dimnames(bf) <-
  list(c('rscale=1/sqrt(2)', '       sqrt(2)', 'BIC','Effect size'),
       paste0(c("n=",rep("",length(Nval)-1)),Nval))
bfVal <- setNames(rep(list(bf),length(pval)),
```

```
                  paste0('p', substring(paste(pval),2)))
for(k in 1:length(pval)){
  p <- pval[k]
for(i in 1:length(rs))for(j in 1:nlen)
  bfVal[[k]][i,j] <- t2BF(p=p, N=Nval[j], xrs=rs[i])
bfVal[[k]][length(rs)+1,] <- outer(p, Nval, t2BFbic)
bfVal[[k]][length(rs)+2,] <- outer(p, Nval, t2eff)
}
lapply(bfVal, function(x)signif(x,2))
```

```
$p.05
                  n=3   4   5   10   20   40   80  160  360
rscale=1/sqrt(2)  2.6 2.4 2.2 1.80 1.40 1.10 0.80 0.59 0.40
       sqrt(2)    3.0 2.5 2.1 1.30 0.89 0.62 0.43 0.31 0.20
BIC              19.0 9.6 6.6 3.00 1.80 1.20 0.79 0.55 0.36
Effect size       2.5 1.6 1.2 0.72 0.47 0.32 0.22 0.16 0.10

$p.01
                   n=3    4    5   10   20   40  80  160  360
rscale=1/sqrt(2)   6.4  7.0  7.0  6.2 5.10 4.10 3.1 2.30 1.60
       sqrt(2)     9.8  9.6  8.7  5.6 3.70 2.50 1.8 1.20 0.82
BIC              210.0 77.0 45.0 15.0 8.00 5.00 3.3 2.30 1.50
Effect size        5.7  2.9  2.1  1.0 0.64 0.43 0.3 0.21 0.14

$p.001
                  n=3      4     5    10    20    40    80   160   360
rscale=1/sqrt(2)   21   32.0  38.0  41.0 37.00 31.00 24.00 18.00 13.00
       sqrt(2)     39   56.0  60.0  47.0 31.00 21.00 15.00 10.00  6.70
BIC              6500 1600.0 750.0 180.0 77.00 44.00 28.00 19.00 12.00
Effect size        18    6.5   3.9   1.5  0.87  0.56  0.38  0.27  0.17
```

Note several points:
- The BIC-based 'Bayes Factor' gives unreasonably large factors for small values of n. Not until n=80 is the value in much the same ballpark as the Bayes Factor generated by the *BayesFactor* function. The BIC statistic really is designed for use in a "large sample" context.
- For large enough values of n, the BIC-based values lie between the Bayes Factor with for an `rscale` of $1/\sqrt{2}$ and that for an `rscale` of $\sqrt{2}$.
- As n increases, the estimated effect size to which the Bayes Factor corresponds becomes ever smaller.

Note then that `BayesFactor::ttestBF()` with the default setting of `rscale`, and the BIC-based Bayes Factor, are both using a prior whose scale is large relative to an ever smaller effect

size.

Matching the setting of `rscale` to the effect size

Observe then the result from matching the scale for the prior to the effect size. The following checks this for p=0.05, making at the same time a comparison with AIC-based and BIC-based relative 'preferences'.

```
rs <- c(0.5,1,4,16)
pval <- 0.05
BFrs <- matrix(nrow=length(rs)+3, ncol=nlen)
dimnames(BFrs) <-
  list(c(paste0(c("rscale=",rep("       ",3)),rs,"/sqrt(n)"),
         "rscale=1/sqrt(2)","BIC-based","AIC-based"),
       paste0(c("n=",rep("",nlen-1)),Nval))
for(j in 1:nlen){
  for(i in 1:length(rs))
     BFrs[i,j] <- t2BF(p=pval, N=Nval[j], xrs=rs[i]/sqrt(Nval[j]))
  BFrs[length(rs)+1, j] <- t2BF(p=pval, N=Nval[j], xrs=1/sqrt(2))
  BFrs[length(rs)+2, j] <- t2BFbic(p=pval, N=Nval[j])
  BFrs[length(rs)+3, j] <- t2AIC(p=pval, N=Nval[j])
}
print(setNames("p=0.05",""), quote=F)
```

```
p=0.05
```

```
round(BFrs,2)
```

```
                     n=3    4    5   10   20   40   80  160  360
rscale=0.5/sqrt(n)  1.84 1.77 1.71 1.59 1.53 1.50 1.48 1.48 1.47
       1/sqrt(n)    2.38 2.19 2.06 1.81 1.70 1.65 1.62 1.61 1.60
       4/sqrt(n)    3.02 2.28 1.92 1.41 1.22 1.15 1.11 1.10 1.09
       16/sqrt(n)   1.48 0.91 0.70 0.46 0.39 0.36 0.35 0.34 0.34
rscale=1/sqrt(2)    2.56 2.38 2.22 1.76 1.39 1.07 0.80 0.59 0.40
BIC-based          18.96 9.57 6.56 3.00 1.78 1.16 0.79 0.55 0.36
AIC-based          12.08 7.04 5.39 3.49 2.93 2.71 2.60 2.56 2.53
```

Thus, the BIC is designed to look for effect sizes that are around one. If a small effect size is expected in a large sample context, use of `ttestBF()` or `ttest.tstat()` with a setting of `rscale` that matches the expected effect size, makes better sense than use of `BIC()`.

There is a choice of prior that allows the AIC-based preference measure to be interpreted as a Bayes Factor. See Burnham & Anderson (2004). Relative preference values that are larger than from the *BayesFactor* functions at all settings of `rscale` suggests a tendency to choose an overly complex model.

For p=0.01 we find:

```
rs <- c(0.5,1,4,16)
pval <- 0.01
BFrs <- matrix(nrow=length(rs)+3, ncol=nlen)
dimnames(BFrs) <-
  list(c(paste0(c("rscale=",rep("       ",3)),rs,"/sqrt(n)"),"rscale=1/sqrt(2)","BIC-based","
for(j in 1:nlen){
  for(i in 1:length(rs))
     BFrs[i,j] <- t2BF(p=pval, N=Nval[j], xrs=rs[i]/sqrt(Nval[j]))
  BFrs[length(rs)+1, j] <- t2BF(p=pval, N=Nval[j], xrs=1/sqrt(2))
  BFrs[length(rs)+2, j] <- t2BFbic(p=pval, N=Nval[j])
  BFrs[length(rs)+3, j] <- t2AIC(p=pval, N=Nval[j])
}
```

```
print(setNames("p=0.01",""), quote=F)
```

```
p=0.01
```

```
round(BFrs,2)
```

```
                     n=3     4     5    10    20    40    80   160   360
rscale=0.5/sqrt(n)  3.49  3.65  3.62  3.38  3.22  3.13  3.09  3.07  3.06
       1/sqrt(n)    5.56  5.71  5.54  4.87  4.48  4.28  4.19  4.14  4.12
       4/sqrt(n)   12.32 10.39  8.77  5.85  4.75  4.30  4.09  3.99  3.94
       16/sqrt(n)  12.09  6.54  4.51  2.31  1.73  1.51  1.42  1.38  1.35
rscale=1/sqrt(2)    6.38  7.03  7.05  6.20  5.12  4.08  3.12  2.32  1.60
BIC-based         205.66 76.53 44.54 15.34  8.04  4.96  3.29  2.25  1.47
AIC-based         131.05 56.31 36.64 17.84 13.23 11.54 10.81 10.47 10.29
```

Use of functions from the *BFpack* package

We investigate the Bayes Factors that are calculated using the Fractional Bayes Factor Approach. The details are not easy to describe simply. However the effect is that allowance must

be made for the use of a fraction of the information in the data to determine the null. See Mulder et al (2021).

We compare

- Bayes Factor with Jeffreys-Zellner-Siow prior centered on NULL
- Fractional Bayes Factor from *BFpack* (`BF.type=1`), i.e., the prior is centered on the NULL
- Fractional Bayes Factor from *BFpack* (`BF.type=2`), i.e., the prior is centered on the maximum likelihood estimate under the alternative.
- Alternative versus NULL, based on Bayesian Information Criterion (BIC)

```
suppressPackageStartupMessages(library(BayesFactor))
suppressPackageStartupMessages(library(BFpack))
suppressPackageStartupMessages(library(metRology))
pval <- c(.05,.01,.001); np <- length(pval)
Nval <- c(3:5,10,20,40,80,160,320); nlen <- length(Nval)
bicVal <- outer(pval, Nval, t2BFbic)
# t2BF <- function(p, N){t <- qt(p/2, df=N-1, lower.tail=FALSE)
#                        BayesFactor::ttest.tstat(t=t, n1=N, simple=TRUE)}
BFval <- packValNull <- packValAlt <- matrix(nrow=np,ncol=nlen)
dimnames(packValNull) <- dimnames(packValAlt) <- dimnames(bicVal) <-
  dimnames(BFval) <-
  list(paste(pval), paste0(c("n=",rep("",nlen-1)),Nval))
for(i in 1:np)for(j in 1:nlen){
  t <- qt(pval[i]/2,Nval[j]-1,lower.tail=F)
  d <- rnorm(Nval[j])
  d <- d-mean(d)+t*sd(d)/sqrt(Nval[j])
  tt <- bain::t_test(d)
  packValNull[i,j] <- BF(tt,  hypothesis='mu=0',
    BF.type=1)[['BFmatrix_confirmatory']]['complement', 'mu=0']
  packValAlt[i,j] <- BF(tt,  hypothesis='mu=0',
    BF.type=2)[['BFmatrix_confirmatory']]['complement', 'mu=0']
  BFval[i,j] <- t2BF(pval[i], Nval[j])}
```

```
## Fractional Bayes factor, center on point null
print(setNames("Fractional Bayes Factor, center prior on null",""), quote=F)
```

```
Fractional Bayes Factor, center prior on null
```

```
print(packValNull, digits=3)
```

```
        n=3     4     5    10    20     40     80   160   320
0.05   2.04  2.19  2.13  1.66  1.20  0.856  0.607  0.43 0.304
0.01   4.51  6.19  6.71  6.10  4.66  3.395  2.432  1.73 1.227
0.001 14.24 28.34 36.64 42.93 35.65 26.849 19.520 13.98 9.951
```

```
## Bayes Factor (Cauchy prior, `rscale="medium")`
print(setNames("From `BayesFactor::ttestBF()`, center prior on null",""), quote=F)
```

```
From `BayesFactor::ttestBF()`, center prior on null
```

```
print(BFval, digits=3)
```

```
        n=3     4     5    10    20    40    80    160    320
0.05   2.56  2.38  2.22  1.76  1.39  1.07  0.80  0.585  0.422
0.01   6.38  7.03  7.05  6.20  5.12  4.08  3.12  2.321  1.688
0.001 21.44 32.35 37.65 40.86 36.57 30.53 24.19 18.364 13.527
```

```
## BIC-based to BFpack::BF() ratio
print(setNames("FBF, center prior on null: Ratio to BayesFactor result",""), quote=F)
```

```
FBF, center prior on null: Ratio to BayesFactor result
```

```
print(packValNull/BFval, digits=3)
```

```
        n=3     4     5    10    20    40    80   160   320
0.05  0.797 0.919 0.958 0.939 0.864 0.800 0.759 0.734 0.721
0.01  0.708 0.880 0.952 0.984 0.910 0.833 0.778 0.745 0.727
0.001 0.664 0.876 0.973 1.050 0.975 0.879 0.807 0.761 0.736
```

`BFpack::BF()` with BF.type=2 vs derived from BIC

```r
# Fractional Bayes factor, center on estimate under alternative
print(setNames("FBF, center on estimate under alternative",""), quote=F)
```

FBF, center on estimate under alternative

```r
print(packValAlt, digits=3)
```

```
         n=3       4      5    10    20     40     80    160    320
0.05    20.9    9.57   6.22   2.6  1.48  0.946  0.638  0.441  0.308
0.01   226.8   76.53  42.27  13.3  6.67  4.033  2.647  1.804  1.253
0.001 7123.0 1606.11 715.79 151.9 63.95 35.565 22.407 14.973 10.295
```

```r
## From BIC
print(setNames("Derived from BIC",""), quote=F)
```

Derived from BIC

```r
print(bicVal, digits=3)
```

```
       n=3       4      5    10    20    40     80   160    320
0.05    19    9.57   6.56   3.0  1.78  1.16  0.792  0.55  0.385
0.01   206   76.53  44.54  15.3  8.04  4.96  3.286  2.25  1.567
0.001 6460 1606.11 754.24 175.7 77.10 43.73 27.819 18.68 12.872
```

```r
## BIC-based to BFpack::BF() ratio
print(setNames("FBF, center prior on alternative: Ratio to BIC",""), quote=F)
```

FBF, center prior on alternative: Ratio to BIC

```r
print(packValAlt/bicVal, digits=3)
```

```
      n=3 4     5    10    20    40    80   160 320
0.05  1.1 1 0.949 0.865 0.829 0.813 0.805 0.802 0.8
0.01  1.1 1 0.949 0.865 0.829 0.813 0.805 0.802 0.8
0.001 1.1 1 0.949 0.865 0.829 0.813 0.805 0.802 0.8
```

The function `BFpack::BF()` is making allowance for the use of a fraction of the information in the data used to specify the prior distribution. The BIC based calculations do not make such an adjustment.

As for the use of the BIC to choose between a simpler and a more complex model, the calculated Bayes Factors are unreasonably large for small samples, while in larger samples the prior is tuned to detect effect sizes that are of similar (or larger) magnitude than the standard deviation.

Figure 2 summarizes the comparisons made

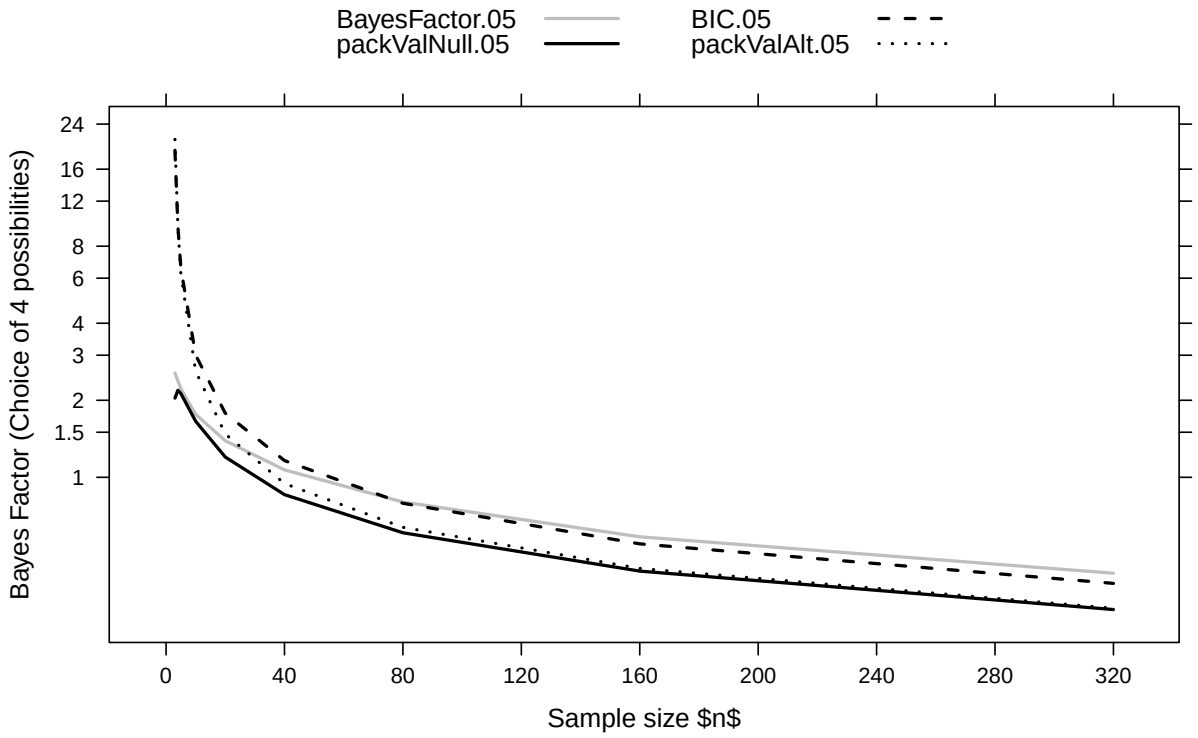

Figure 2: Results from different ways to calculate the Bayes Factor - for a result from a one-sample two-sided t-test where p=0.05

Code is:

```
library(lattice)
allVal <- rbind(BFval, packValNull, bicVal, packValAlt)
rownames(allVal) <- paste0(
  rep(c('BayesFactor', 'packValNull', 'BIC', 'packValAlt'), c(3,3,3,3)),
  rep(c(".05",".01",".001"), 4))
tdf <- as.data.frame(t(allVal))
tdf$n <- Nval
```

```
labs <- sort(c(2^(0:6),2^(0:6)*1.5))
xyplot(BayesFactor.05+packValNull.05+BIC.05+packValAlt.05 ~ n,
       data=tdf, type='l', auto.key=list(columns=2),
       xlab="Sample size $n$",
       ylab="Bayes Factor (Choice of 4 possibilities)",
       scales=list(x=list(at=(0:8)*40),
         y=list(log=T, at=labs, labels=paste(labs))),
 par.settings=simpleTheme(lty=c(1,1:3), lwd=2, col=rep(c('gray','black'), c(1,3))))
```

Bayes Factors for regression coefficients.

We will use the following function to simulate data, with two explanatory variables, for use in regression calculations:

```
simDat <- function(x1=rep(1:20,4)/5, x2=sample(rep(1:20,4)/5),
                   b1=1.2, b2=1.5, sd=8){
  n <- length(x1)
  data.frame(x1=x1, x2=x2, y=b1*x1+b2*x2+rnorm(n,sd=sd))
}
```

One data generating mechanism – large dataset to dataset variation

Note first that with the default settings, the simulated data and p-values in the fitted model show large variation from one simulation to the next. The following shows relatively extreme differences:

```
set.seed(17)
dat <- simDat()
y.lm <- lm(y~x1+x2, data=dat); bf12 <- lmBF(y ~ x1+x2, data=dat)
## Repeat simulation of data
dat <- simDat()
yy.lm <- lm(y~x1+x2, data=dat); bf12 <- lmBF(y ~ x1+x2, data=dat)
cbind(coef(summary(y.lm))[,-2], coef(summary(yy.lm))[,-2]) |> signif(2)
```

```
            Estimate t value Pr(>|t|) Estimate t value Pr(>|t|)
(Intercept)     -2.4    -1.0   0.3200     1.40    0.63    0.530
x1               1.7     2.3   0.0250     1.80    2.40    0.018
x2               2.0     2.7   0.0076     0.74    1.00    0.310
```

P-values from nine simulations are:

```
multSims <- function(sd, nsims=9, rnam=c("(Intercept)","x1","x2")){
pvals <- matrix(nrow=3, ncol=nsims, dimnames=list(rnam,
                paste0('pval', 1:nsims)))
for(i in 1:nsims){dat <- simDat(sd=sd)
  dat.lm <- lm(y~x1+x2, data=dat); bf12 <- lmBF(y ~ x1+x2, data=dat)
  pvals[,i] <- coef(summary(dat.lm))[,4]
}
pvals
}
multSims(sd=8) |> signif(2)
```

```
            pval1 pval2 pval3  pval4  pval5 pval6  pval7 pval8 pval9
(Intercept)  0.40 0.690 0.590 0.2000 0.3500 0.850 0.7800 0.790 0.910
x1           0.49 0.081 0.250 0.0460 0.0380 0.089 0.2500 0.390 0.025
x2           0.44 0.020 0.037 0.0012 0.0064 0.190 0.0024 0.022 0.380
```

Now try with `sd=5`:

```
multSims(sd=5) |> signif(2)
```

```
              pval1  pval2  pval3   pval4   pval5   pval6  pval7   pval8  pval9
(Intercept) 5.7e-01 0.8600 0.6000 0.77000 0.73000 2.2e-01 0.1800 0.09900 0.4200
x1          1.6e-01 0.0510 0.0015 0.16000 0.01700 7.4e-05 0.4800 0.00089 0.0042
x2          3.5e-06 0.0023 0.0150 0.00033 0.00025 6.2e-03 0.0001 0.00004 0.0460
```

The p-values are more consistently small.

Bayes Factors and BIC statistics

Now create a simulated dataset, and calculate Bayes Factors for the coefficients (1) using *Bayesfactor* functions, and (2) derived from BIC statistics:

```
set.seed(31)
dat31 <- simDat()
y.lm <- lm(y~x1+x2, data=dat31); bf12 <- lmBF(y ~ x1+x2, data=dat31)
y2.lm <- lm(y~x2, data=dat31); bf2 <- lmBF(y ~ x2, data=dat31)
y1.lm <- lm(y~x1, data=dat31); bf1 <- lmBF(y ~ x1, data=dat31)
## Regression summary
coef(summary(y.lm)) |> signif(2)
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.20        2.2    0.55    0.590
x1              0.99        0.7    1.40    0.160
x2              1.80        0.7    2.50    0.014
```

```
## Bayes Factors for x1 and x2, using functions from _Bayesfactor_
c(extractBF(bf12/bf2)$bf, extractBF(bf12/bf1)$bf) |> round(2)
```

```
[1] 0.70 4.45
```

```
## Bayes Factors for x1 and x2, derived from BIC statistics
c(exp((BIC(y2.lm)-BIC(y.lm))/2), exp((BIC(y1.lm)-BIC(y.lm))/2)) |> round(2)
```

```
[1] 0.31 2.65
```

Both are substantially smaller than those derived from calculations using `BayesFactor::lmBF()`.

Matching the Bayes Factor to the SEs of the coefficients

Check also the difference made to the Bayes Factors by setting the scale parameter (here `rscaleCont`) to a value that is matched to the standard error of the coefficient estimates. A standard error that is just under 0.7 is the same for both coefficients. We standardise this by dividing by a standard deviation of just under 7.2, multiply by the default `rscale` that equals $\sqrt{(2)}/4$, and use this as the value for `rscaleCont`.

```
rs <- 2*0.75/7.2
  bf12 <- lmBF(y ~ x1+x2, data=dat31, rscaleCont=rs)
  bf2 <- lmBF(y ~ x2, data=dat31, rscaleCont=rs)
  bf1 <- lmBF(y ~ x1, data=dat31, rscaleCont=rs)
  c(extractBF(bf12/bf2)$bf, extractBF(bf12/bf1)$bf) |> round(2)
```

```
[1] 0.84 4.33
```

In this context, the smaller Bayes Factor is modestly increased, with the larger Bayes Factor slightly reduced.

Different statistics offer different perspectives

In addition to the choice between different prior families, one has to choose a scale for the prior, if this is not done automatically. Different choices can lead to quite different Bayes Factors. Be aware that Bayes Factors are at best a rough measure of model preference. Use them along with other measures of model preference. Keep in mind that when samples are small, different samples from the same population, if available, would give widely varying results. Refer back to Figure 1, which showed what could be expected for p-values.

The comparisons could usefully be extended to consider other choices of prior.

References

Burnham, K. P., & Anderson, D. R. (2004). Multimodel inference: understanding AIC and BIC in model selection. Sociological methods & research, 33(2), 261-304.

Kahneman, D. (2011). Thinking, fast and slow. Macmillan.

Mulder, J., Williams, D. R., Gu, X., Tomarken, A., Böing-Messing, F., Olsson-Collentine, A., Meijerink-Bosman, M., Menke, J., van Aert, R., Fox, J.-P., Hoijtink, H., Rosseel, Y., Wagenmakers, E.-J., & van Lissa, C. (2021). BFpack: Flexible Bayes Factor Testing of Scientific Theories in R. Journal of Statistical Software, 100(18), 1–63. https://doi.org/10.18637/jss.v100.i18