SUPPLEMENTARY MATERIALS
Posterior-Hacking: Selective Reporting Invalidates Bayesian Results Also

Uri Simonsohn
The Wharton School
University of Pennsylvania
uws@wharton.upenn.edu

Outline
Supplement 1. Details on the 10 studies depicted on Figure 1 (Pages 2-3)
Supplement 2. R-Code behind simulations depicted on Figure 3 (Pages 4-7)
**Posterior-Hacking**
Supplementary Materials

**Supplement 1. Ten examples in Figure 1.**

Below I provide some details for the results reported in Figure 1. The figure reports ten differences of means analyzed with a t-test and with Bayesian estimation as proposed by Kruschke (2013). The data for these were obtained from 9 papers published in the *Journal of Decision Making* in 2012. The journal was selected for being one of the few requiring data posting. I browsed articles in chronological order of publication, arbitrarily selecting variables to compare with the single objective of obtaining a broad range of types of variables.

Sometimes I also selected an arbitrary subset of conditions (e.g., if an experiment included 3 conditions, I only compared a subset of 2) or an arbitrary comparison that was not of interest to the authors (see Examples 9 and 10).

All decision were made before seeing the comparison of t with Bayesian estimation and hence are effectively ex-ante decisions. All datasets are publicly available from the journal’s website, below I very briefly describe which variables I selected.

**Example 1. Money (Dhingra, Gorn, Kener, & Dana, 2012)**
The single experiment in the paper has four conditions. On the journal’s website the dataset is split into two files (apparently the study was run in 2 waves with slight differences). They were merged vertically for the analysis. Figure 1 compares $ kept between participants given a $0 and a $10 default, relevant columns are “Round 1 kept” indicated the amount of $ kept, and “rd 1 default” (I used the rows where it equals 0 or 10). The difference of means is 2.0591.

**Example 2. Heart rate (Starcke, Ludwig, & Brand, 2012)**
The paper induces stress by telling subjects they will perform a speech. They measured several variables, I selected heart rate. For example 8 in that Figure 1 also used reaction time from this same paper. Figure 1 reports the difference of means for the heart rate measure after the manipulation “labeled EKG_manipulation” in their file, across groups 1 and 2. The difference is 15.77 which is reported divided by 10 in the figure as 1.577.

**Example 3. (Svenson, Eriksson, Slovic, Mertz, & Fuglestad, 2012)**
The study is the 2nd experiment (data for 1st is not posted), I compared Case 1 across the two condition where the driver went over a child or a line. The difference of means is 2.93.

**Example 4. Travel time (Peer & Solomon, 2012)**
The study compares taxi drivers with student drivers. The experimenter entered a cab and asked the driver a series of questions, including how long they thought the trip from the experimenter’s home to the university, or vice versa, would take. Student drivers were asked in a survey form for the same questions. Figure 1 compared estimated journey
duration in minutes. The difference of means is 3.166 (comparing column “time_usually” across “group” 1 vs 2. (see row 1 in their Table 1).

**Example 5. Regret (Huang & Zeelenberg, 2012)**
For experiment 1 I compare the *above expectations* condition with the *below expectation* condition, using the “R1” measure of regret (there are two and their average, I analyze the first). The difference of means is 1.04

**Example 6. Count data (Verrier, 2012)**
Voting in the Eurovision contest is compared as a function of the voter being in the same vs. different round of the competition. Figure 1 compares the column “SF1_total” based on “semifinal_appearance” being 1 vs. 2.
The difference of means is 5.8750. The Figure reports it divided by 2 so as not to affect the y-scale for all other comparisons.

**Example 7. 1-100 rating (Shahrabani, Benzion, Rosenboim, & Shavit, 2012)**
The paper asked people who stayed and left an area that was experiencing attacks a series of questions. Figure 1 compares the average answer to the question “Terrorism Risk” across both groups, as indexed by the variable “Location during the was (1-in rocket range”). The dependent variable is the participants’ responses to the question “how likely is it that…” with answers anchored at 0% and 100%.
The difference of means is 2.62

**Example 8. Reaction time (Starcke et al., 2012)**
This is the same paper used for Example 2. For reaction time I used the column “reaction_times”. It is measured in milliseconds, I divide by 1000 to arrive at a measure in seconds. The difference of means is 1.857

**Example 9. Age (Roets, Schwartz, & Guan, 2012)**
The paper has three samples I used the one collected in the United States. Seeking a variable different in nature from those already included, I -somewhat absurdly- compare average age for men and women in that sample. Note that females are indexed with an F or with a V, while males only with an M. The difference of means is 4.978 years.

**Example 10. # Answers (Szrek, Chao, Ramlagan, & Peltzer, 2012)**
This papers asked respondents a wide range of questions and forecasts. Five questions in the sets were used to measure numeracy. The respondents’ answers were coded as correct or incorrect and the total number of correct answers used as a measure of numeracy. The variable name in the file is “numeracy”, it takes values [0,1,2,3,4]. I compared it across males and females. The difference of means is .28.
Posterior-Hacking
Supplementary Materials

Supplement 2. R-code behind Figure 3

http://opim.wharton.upenn.edu/~uws/papers/R_posterior_hacking_supplement_2.R

#CONCEPTUAL OUTLINE

#THIS PROGRAM RUNS A SIMULATIONS WHERE
#(1) DATA
# Each iteration generates a dataset with:
# - Three samples with n=30 observations each
# - I refer to them as "low" "medium" and "high", index them with the "cond" variable, values=(-1, 0, +1)
# - Three dependent variables, y1,y2,y3 correlated .75 to a latent variable, ~.5 with each other
#
#(2) t-tests
# After generating the dataset the script runs 48 t-tests, that combine the following four forms of p-hacking
# - Data peeking: 2 (Using all n=30 observations or the first n=20) x
# - Choose d.f. 4 (Using each of three d.v. or their average) x
# - Drop outliers 2 (Excluding "outliers" or not (operationalized with highest/lowest two obs. across the compared two samples))
# - Drop condition 3 (Comparing 2 out of the 3 conditions run (H vs M) or (H vs L) or (M vs L))
# ------
# 2x4x2x3=48 combinations
#
#(3) Reported p-value
# I simulate p-hacking by computing the minimum p-value among subsets of the 48 tests
# -For example, to assess the impact of data-peeking alone i compute, for the H vs M contrast, the p-values with n=20
# and n=30 and assess if the lowest of the two is smaller than .01
#
#(4) Bayes factor-hacking
# I then compute the Bayes factor associated with the lowest p-value and count then number that are BF>3. Note that this slightly benefits BF because the lowest p-value need not be the highest BF and hence it is possible that the lowest p-value is not BF>3 but a higher p-value is.

#OUTLINE OF CODE

#(1) Function that computes 6 p-values for a give d.v. and sample size (3 condition pairs, with/without exclusions)
#(2) Loop that generates data, runs the function 8 times (for each of for d.v.s, y1,y2,y3 and their average, and for n=20,n=30
#    Arriving at 48 p-values
#(3) Compute minimum p-value in each of the subsets
#(4) Compute Bayes factor for each of the significant t-values

#(1) I CREATE A FUNCTION THAT COMPUTES 6 P-VALUES FOR GIVEN DV AND SAMPLE SIZE

f=function(y,ntot) {
  #create subsets to find highest and lowest two values of each vector more easily
  z1=subset(y,cond!=1 & n<=ntot)  #DROP LOW
  z2=subset(y,cond!=0 & n<=ntot)   #DROP MED
  z3=subset(y,cond!=1 & n<=ntot)   #DROP HIGH

  #t-tests including all ntot observations
  p1=t.test(y~cond,subset=(cond!=1 & n<=ntot),var.equal=TRUE, data=data.all)$p.value  #Drop low
  p2=t.test(y~cond,subset=(cond!=0 & n<=ntot),var.equal=TRUE, data=data.all)$p.value  #Drop med
  p3=t.test(y~cond,subset=(cond!=1 & n<=ntot),var.equal=TRUE, data=data.all)$p.value  #Drop high

  #t-tests dropping lowest and highest two
  p4=t.test(y~cond,subset=(cond!=1 & n<=ntot & y<sort(z1)[2*ntot-1] & y>sort(z1)[2]),var.equal=TRUE, data=data.all)$p.value  #Drop low
  p5=t.test(y~cond,subset=(cond!=0 & n<=ntot & y<sort(z2)[2*ntot-1] & y>sort(z2)[2]),var.equal=TRUE, data=data.all)$p.value  #Drop med
  p6=t.test(y~cond,subset=(cond!=1 & n<=ntot & y<sort(z3)[2*ntot-1] & y>sort(z3)[2]),var.equal=TRUE, data=data.all)$p.value  #Drop high
Supplementary Materials

Posterior-Hacking

return(c(p1,p2,p3,p4,p5,p6))
}

#total number of simulations
simtot=15000  #paper reports 15,000; that can take nearly an hour. Set to 1000 for ~2 minutes
set.seed(1975)  #seeded so that exactly the same results every time

#STORING RESULTS ACROSS SIMULATIONS
#Creating empty vectors where results from the simulations are stored
#weird anachronistic requirement by R language seems to me.
ptot=c()
p_A=c()  #n=20 or 30
p_B=c()  #choose dv
p_C=c()  #drop or collapse condition
p_D=c()  #drop outliers
#combinations
p_AB=c()
p_ABC=c()
p_ABCD=c()

#Store d.f. associated with p-hacked test (for converting t-value into Bayes factor)
df_A=c()
df_AB=c()
df_ABC=c()
df_ABCD=c()

#These indicate the d.f. associated with the each of the p-values computed in order, that way I find which kth
#p-value I am reporting, and look-up the kth d.f. for the subset
df_A.codebook =c(38,58)  #e.g., if n=20 per each of two cell, df=38, if raise to n=30 then df=58
df_AB.codebook =c(38,38,38,58,58,58)  #y1,y2,avg(y1) has n=20 per cell, if add to n=30, then 58

#to make df_ABC.codebook easier to write down i create the repeated subset
df_half=c(38,38,38,58,58,58)
df_ABC.codebook =c(df_half,df_half,df_half,df_half)

#to make df_ABCD.codebook easier to write down i create the repeated subset
df20=c(38,38,38,58,58,58)
df30=c(58,58,58,54,54,54)
df_ABCD.codebook=c(df20,df30,df20,df30,df20,df30,df20,df30)

#Simulations start here
time0=Sys.time()  #to compute computing time
for (simk in 1:simtot)
{

#DEPENDENT VARIABLES
yl=norm(n=90)  #Latent dependent variable
c1=norm(n=90)  #random error to generate y1 correlated .75 with latent
c2=norm(n=90)  #random error to generate y2 correlated .75 with latent
c3=norm(n=90)  #random error to generate y3 correlated .75 with latent
y1=yl*0.75+(1-.75**2)**.5*c1  #Dependent variable 1
y2=yl*0.75+(1-.75**2)**.5*c2  #Dependent variable 2
y3=yl*0.75+(1-.75**2)**.5*c3  #Dependent variable 3
y4=(y1+y2+y3)/3  #Average dependent variable

#CONDITION
cond=sort(rep(c(-1,0,1),each=30))  #Creates -1,0,1 as condition, first 30 are -1, then 0, then 1

#Observation#
n1=seq(from=1,to=30,by=1)  #counter 1-30
n=c(n1,n1,n1)  #copied for each condition
Posterior-Hacking
Supplementary Materials

#Create the dataset
data.all=data.frame(y1,y2,ya,cond,n)

#RUN THE TESTS USING FUNCTION DEFINED ABOVE, EACH FUNCTION RETURNS 6 P-VALUES

#P-VALUE NUMBERS:
ps1=f(y=y1,ntot=20) #1-6
ps2=f(y=y2,ntot=20) #7-12
ps3=f(y=y3,ntot=20) #13-18
ps4=f(y=ya,ntot=20) #19-24
ps5=f(y=y1,ntot=30) #25-30
ps6=f(y=y2,ntot=30) #31-36
ps7=f(y=y3,ntot=30) #37-42
ps8=f(y=ya,ntot=30) #43-48

pk=c(ps1,ps2,ps3,ps4,ps5,ps6,ps7,ps8) #all 48 p-values into a single vector

#COMPUTE SUBSETS OF TESTS FOR P-HACKING
p_A.this =c(pk[1],pk[25])  #data peeking
p_B.this =c(pk[1],pk[7],pk[13],pk[19])  #choose d.v.
p_C.this =c(pk[1],pk[2],pk[3])  #drop condition
p_D.this =c(pk[1],pk[4])  #drop outlier

#COMBINE A & b
#Comparing the High vs Mid condition, for n=20 or n=30, for each of y1,y2,y3,ya

p_AB.this =c(pk[1],pk[7],pk[13],pk[19],pk[25],pk[31],pk[37],pk[43])

#COMBINE A & B & c

#Store lowest p-value of each subset into the corresopnding vector of results
p_A =c(p_A,min(p_A.this))
p_B =c(p_B.min(p_B.this))
p_C =c(p_C.min(p_C.this))
p_D =c(p_D.min(p_D.this))
p_AB =c(p_AB.min(p_AB.this))
p_ABC =c(p_ABC.min(p_ABC.this))
p_ABCD =c(p_ABCD.min(pk))

#FOR BAYESIAN FACTOR CALCULATIONS IDENTIFY D.F. OF EACH P-HACKED TEST HAS
df_A.this =df_A.codebook[match(min(p_A.this), p_A.this)]  #The match() function finds the order of the selected
df_AB.this =df_AB.codebook[match(min(p_AB.this), p_AB.this)]  # p-value, and uses that same order from the d.f.
df_ABCD.this=df_ABCD.codebook[match(min(p_ABCD.this), p_ABCD.this)]

#STORE THE D.F. FOR THIS TEST
df_A=df_A.this
df_AB=df_AB.this
df_ABC=df_ABCD.this
Posterior-Hacking
Supplementary Materials

df_ABCD=c(df_ABCD,df_ABCD.this)
}

#LOOP ENDS

#REPORT COMPUTING TIME FOR ALL SIMULATIONS
time1=Sys.time()
print(time1-time0)

#COMPUTE T-VALUES FOR COMPUTING BAYESIAN FACTORS
t_A =qt(p=p_A/2 ,df=df_A)
t_AB =qt(p=p_AB/2 ,df=df_AB)
t_ABC =qt(p=p_ABC/2 ,df=df_ABC)
t_ABCD=qt(p=p_ABCD/2,df=df_ABCD)

##############################
#RESULTS
#how many p<.01
p1=sum(p_A<.01)/simtot   #p<.01
p2=sum(p_AB<.01)/simtot   #p<.01
p3=sum(p_ABC<.01)/simtot   #p<.01
p4=sum(p_ABCD<.01)/simtot   #p<.01

#how many BF>3
BF1=sum(BF_A>3)/simtot   #BF>3
BF2=sum(BF_AB>3)/simtot   #BF>3
BF3=sum(BF_ABC>3)/simtot   #BF>3
BF4=sum(BF_ABCD>3)/simtot   #BF>3

#OUTPUT FOR FIGURE 2
OUTPUT=matrix(c(p1,p2,p3,p4,BF1,BF2,BF3,BF4),nrow=2,ncol=4,byrow=TRUE))

References for the supplement.
Posterior-Hacking
Supplementary Materials


