Supplementary Information: Schad, Betancourt, & Vasishth
Toward a principled Bayesian workflow in cognitive science

Code S1

```
priors <- c(set_prior("normal(0, 10)", class = "Intercept"),
            set_prior("normal(0,  1)", class = "b", coef="so"),
            set_prior("normal(0,  1)", class = "sd"),
            set_prior("normal(0,  1)", class = "sigma"),
            set_prior("lkj(2)", class = "cor"))
```

Code S2

```
# prepare data
gw    <- read.table("data/gibsonwu2012data.txt", header=TRUE) # load data
gw$so <- ifelse(gw$type%in%c("subj-ext"),-1,1) # sum-coding for predictor
gw1   <- subset(gw,region=="headnoun") # subset critical region
expdesign <- gw1[,c("subj","item","so")] # extract experimental design
source("scripts/SimFromPrior.R") # load simulation functions
source("scripts/genfake.R")

# prepare simulation
nsim <- 1000
Nsj  <- max(unique(expdesign$subj)) # determine no. of subjects
Nit  <- max(unique(expdesign$item)) # determine no. of items
beta0 <- beta1 <- sigma_u0 <- sigma_u1 <- sigma_w0 <- sigma_w1 <-
  rho_u <- rho_w <- sigma <- NA
rtfakemat <- matrix(NA,nrow(expdesign),nsim)
set.seed(123)

# simulations
for (i in 1:nsim) {
  # simulate parameters
  beta0[i] <- -1; while (beta0[i]<0) # restrict to positive values
     beta0[i] <- SimFromPrior(priors,class="Intercept")
  beta1[i]    <- SimFromPrior(priors,class="b", coef="so")
  sigma_u0[i] <- SimFromPrior(priors,class="sd")
  sigma_u1[i] <- SimFromPrior(priors,class="sd")
  sigma_w0[i] <- SimFromPrior(priors,class="sd")
  sigma_w1[i] <- SimFromPrior(priors,class="sd")
  rho_u[i]    <- SimFromPrior(priors,class="cor")
  rho_w[i]    <- SimFromPrior(priors,class="cor")
  sigma[i]    <- SimFromPrior(priors,class="sigma")
  # simulate data
  rtfakemat[,i] <- genfake(expdesign, Nsj, Nit,
          beta0    = beta0[i],    beta1    = beta1[i],
          sigma_u0 = sigma_u0[i], sigma_u1 = sigma_u1[i],
          sigma_w0 = sigma_w0[i], sigma_w1 = sigma_w1[i],
          rho_u    = rho_u[i],    rho_w    = rho_w[i],
          sigma    = sigma[i])
}
truePars <- data.frame(beta0,beta1,sigma_u0,sigma_u1,sigma_w0,sigma_w1,
                       rho_u,rho_w,sigma)
```

Code S3

```
# Colour codes to use
c_light <- "#DCBCBC"; c_light_highlight <- "#C79999"
c_mid   <- "#B97C7C"; c_mid_highlight   <- "#A25050"
c_dark  <- "#8F2727"; c_dark_highlight  <- "#7C0000"
# set very large data points to a value of 2000
rtfakematH <- rtfakemat; rtfakematH[rtfakematH>2000] <- 2000
# Compute one histogram per simulated data-set
binwidth <- 20
breaks <- seq(0,max(rtfakematH,na.rm=TRUE)+binwidth,binwidth)
histmat <- matrix(NA,ncol=nsim,nrow=length(breaks)-1)
for (i in 1:nsim)
  histmat[,i] <- hist(rtfakematH[,i],breaks=breaks,plot=FALSE)$counts
# For each bin, compute quantiles across histograms
probs <- seq(0.1,0.9,0.1)
quantmat<- as.data.frame(matrix(NA,nrow=dim(histmat)[1],ncol=length(probs)))
names(quantmat) <- paste0("p",probs)
```

```
for (i in 1:dim(histmat)[1])
  quantmat[i,] <- quantile(histmat[i,],p=probs)
quantmat$x <- breaks[2:length(breaks)] - binwidth/2 # add bin mean
# Plot
FigPri1a <- ggplot(data=quantmat, aes(x=x))+
  geom_ribbon(aes(ymax=p0.9, ymin=p0.1), fill=c_light) +
  geom_ribbon(aes(ymax=p0.8, ymin=p0.2), fill=c_light_highlight) +
  geom_ribbon(aes(ymax=p0.7, ymin=p0.3), fill=c_mid) +
  geom_ribbon(aes(ymax=p0.6, ymin=p0.4), fill=c_mid_highlight) +
  geom_line(aes(y=p0.5), colour=c_dark, size=1) +
  labs(title="Prior Predictive Distribution", y="", x="Reading Time [ms]")
```

Code S4

```
tmpM  <- apply(log(rtfakemat),2,mean) # Mean
tmpSD <- apply(rtfakemat,2,sd)
tmpSD[tmpSD>2000] <- 2000 # SD
FigPri1b<-ggplot()+stat_bin(aes(x=tmpM), fill=c_dark)+
  labs(x="Mean [log RT]")
FigPri1d<-ggplot()+stat_bin(aes(x=tmpSD), fill=c_dark)+
  labs(x="Standard Deviation [RT]")
```

Code S5

```
effectSize <- NA
for (i in 1:nsim)
  effectSize[i] <- mean(rtfakemat[expdesign$so==+1,i])-
                   mean(rtfakemat[expdesign$so==-1,i])
effectSize[effectSize> +2000] <- +2000
effectSize[effectSize< -2000] <- -2000
FigPri1c <- ggplot()+stat_bin(aes(x=effectSize), fill=c_dark)+
         labs(x="Object - Subject [S-O RT]")
```

Code S6

```
# effect size per subject
EffectSizeMax <- EffectSizeSD <- NA
for (i in 1:nsim) { tmp <- NA
  expdesign$rtfake <- rtfakemat[,i]
  tmp <- expdesign%>%group_by(subj,so)%>%summarize(rtfake=mean(rtfake))%>%
    spread(key="so",value="rtfake") %>% mutate(dif = `1` - `-1`)
  EffectSizeSD[ i] <- sd(     tmp$dif ,na.rm=TRUE)
  EffectSizeMax[i] <- max(abs(tmp$dif),na.rm=TRUE)
}
EffectSizeMax[EffectSizeMax>2000] <- 2000
FigPri1e <- ggplot()+stat_bin(aes(x=EffectSizeMax), fill=c_dark)+
         labs(x="Max Effect Size [S-O RT]")
EffectSizeSD[EffectSizeSD>2000] <- 2000
FigPri1f <- ggplot()+stat_bin(aes(x=EffectSizeSD), fill=c_dark)+
         labs(x="SD Effect Size [S-O RT]")
```

Code S7

```
priors2 <- c(set_prior("normal(6, 0.6 )", class = "Intercept"),
             set_prior("normal(0, 0.05)", class = "b", coef = "so"),
             set_prior("normal(0, 0.1 )", class = "sd"),
             set_prior("normal(0, 0.5 )", class = "sigma"),
             set_prior("lkj(2)", class = "cor"))
```

Code S8

```
expdesign$fakert <- rtfakemat2[,1]
brm1 <- brm(fakert ~ so + (1+so|subj) + (1+so|item), expdesign,
                    family=lognormal(), prior=priors2,
                    control=list(adapt_delta=0.99, max_treedepth=15))
mods_gw <- list(brm1)
for (i in 2:nsim) {
  expdesign$fakert <- rtfakemat2[,i]
  mods_gw[[i]] <- update(brm1, newdata=expdesign, recompile=FALSE)
}
```

Code S9

```
logPost <- div_trans <- rhat <- neffRatio <- NA
for (i in 1:nsim) { # Extract convergence issues
  logPost[i]   <- log_posterior(mods_gw[[i]])
  np           <- nuts_params(mods_gw[[i]])
  div_trans[i] <- sum(subset(np, Parameter == "divergent__")$Value)
  rhat[i]      <- rhat(mods_gw[[i]])
  neffRatio[i] <- neff_ratio(mods_gw[[i]])
}
table(div_trans)
```

Code S10

```
sbc_rank <- NA
for (i in 1:nsim) { # Compute SBC rank
  postgw      <- posterior_samples(mods_gw[[i]],"^b")
  idx_so      <- seq(1,nrow(postgw),8) # thin to remove autocorrelation
  sbc_rank[i] <- sum( truePars2$beta1[i] < postgw$b_so[idx_so] )
}
est_pars <- data.frame(sbc_rank)
```

Code S11

```
hbins <- seq(0,500,25)-0.5
binom <- qbinom(c(0.005,0.5,0.995), length(sbc_rank), 1/(length(hbins)-1))
pdat  <- data.frame(min=binom[1],med=binom[2],max=binom[3],x=c(-20,520))
ggplot()+
  geom_ribbon(data=pdat, aes(x=x, ymin=min, ymax=max), fill="grey80")+
  geom_line(  data=pdat, aes(x=x,y=med)) +
  stat_bin(data=est_pars, aes(x=sbc_rank), breaks=hbins,
           fill=c_dark, colour=c_dark_highlight) +
  scale_x_continuous(breaks=seq(0,500,100))+
  labs(x="Prior Rank",y="", title="Weakly informative priors")
```

Code S12

```
z_score <- contraction <- NA
for (i in 1:nsim) {
  prior_sd_beta1  <- 0.05 # prior standard deviation
  post_mean_beta1 <- fixef(mods_gw[[i]])[2,1] # posterior mean
  post_sd_beta1   <- fixef(mods_gw[[i]])[2,2] # posterior sd
  z_score[i]      <- (post_mean_beta1 - truePars2$beta1[i]) / post_sd_beta1
  contraction[i]    <- 1 - (post_sd_beta1^2 / prior_sd_beta1^2)
}
estPars <- data.frame(contraction, z_score)
ggplot(data=estPars, aes(x=contraction, y=z_score))+
  geom_hline(yintercept=0)+
  geom_point()+ xlim(c(0,1))+
  labs(x="Posterior Contraction",y="Posterior z-Score")
```

Code S13

```
m_gw <- brm(rt ~ so + (1+so|subj) + (1+so|item), gw1,
            family=lognormal(), prior=priors2, cores=4)
round(fixef(m_gw),3)
```

Code S14

```
postgw <- posterior_samples(m_gw)
ggplot(data=postgw)+stat_bin(aes(x=b_so,y=..density..))+
  labs(x="Object - subject relatives", title="Posterior distribution")
```

Code S15

```
nsamp     <- nrow(postgw)
rtpostmat <- matrix(NA,nrow(expdesign),nsamp)
for (i in 1:nsamp)
  rtpostmat[,i] <- genfake(expdesign, Nsj, Nit,
          beta0    = postgw$b_Intercept[i],
          beta1    = postgw$b_so[i],
          sigma_u0 = postgw$sd_subj__Intercept[i],
          sigma_u1 = postgw$sd_subj__so[i],
          sigma_w0 = postgw$sd_item__Intercept[i],
          sigma_w1 = postgw$sd_item__so[i],
          rho_u    = postgw$cor_subj__Intercept__so[i],
          rho_w    = postgw$cor_item__Intercept__so[i],
          sigma    = postgw$sigma[i])
```

Code S16

```
m_gw1 <- brm(rt ~ so + (1+so|subj) + (1+so|item), gw1,
             family=lognormal(), prior=priors2, cores=4,
             save_all_pars=TRUE, iter=10000, warmup=2000)
t0 <- proc.time()
m_gw0 <- brm(rt ~ 1 + (1+so|subj) + (1+so|item), gw1,
             family=lognormal(), prior=priors2[-2,], cores=4,
             save_all_pars=TRUE, iter=10000, warmup=2000)
BF_informative <- bayes_factor(m_gw1, m_gw0)
time.brm <- proc.time() - t0
BF_informative
```

Code S17

```
c(diffuse=time.brm_diffusePrior[3], weakly.informative=time.brm[3])
```