Data analysis of Keshtiari and Vasishth data

Shravan Vasishth
vasishth@uni-potsdam.de

January 27, 2014

1 Standard analysis as in the paper

Please see the file KeshtiariVasishthDataAnalysis.R for the full code listing. This file only shows the main results, and the code for the bayesian analysis.

```
> ## make results exactly replicable
> set.seed(987654321)
```

Mean reading times for the critical and post-critical regions:

	rel.cl	pronoun	region	M	SE	N
1	no.rel.cl	gap	crit	-1.51	0.03	109.00
3	no.rel.cl	pron	crit	-1.53	0.03	110.00
5	rel.cl	gap	crit	-1.52	0.03	108.00
7	rel.cl	pron	crit	-1.58	0.03	110.00

	rel.cl	pronoun	region	M	SE	N
2	no.rel.cl	gap	post.crit	-1.76	0.03	109.00
4	no.rel.cl	pron	post.crit	-1.81	0.02	110.00
6	rel.cl	gap	post.crit	-1.73	0.03	108.00
8	rel.cl	pron	post.crit	-1.83	0.03	110.00

Here is the code that generates the plots in the paper:

```
> pd <- position_dodge(width = 0.05)
> k<-1
> rel.cl<-ifelse(M.id.w$rel.cl=="no.rel.cl","-RC","+RC")
> pron<-ifelse(M.id.w$pronoun=="gap","gap","pronoun")
> M.id.w2<-data.frame(rel.cl=rel.cl,pron=pron,M.id.w[,c(3:6)])
> M.id.w<-M.id.w2
> p1<-ggplot(subset(M.id.w,region=="crit"), aes(x=pron, y=M,
+ group=rel.cl)) +
+     geom_point(position=pd,shape=21,fill="white",size=k*3) +
```

```
+     geom_line(position=pd,aes(linetype=rel.cl),size=k) +
+     geom_errorbar(aes(ymin=M-SE, ymax=M+SE),
+                   width=.1,position=pd,size=k)+
+     xlab("pron")+
+     ylab("negative inverse reading time (-1000/rt ms^-1)")+
+     scale_colour_hue(name="relative clause status",
+                      # Legend label, use darker colors
+                      breaks=c("-RC", "+RC"),
+                      labels=c("-RC", "+RC"),
+                      l=40)+
+     opts(title="Critical region") +
+ #    scale_y_continuous(limits=c(600,900)) +
+     theme_bw() +
+     opts(legend.position=c(.87, .6)) # Position legend inside
>                                     # This must go after theme_bw
>
> #p1
>
> p1a<-ggplot(subset(M.id.w,region=="crit"), aes(x=pron, y=M,
+ group=rel.cl)) +
+     geom_point(shape=21,fill="white",size=k*3) +
+     geom_line(aes(linetype=rel.cl),size=k) +
+     geom_errorbar(aes(ymin=M-2*SE, ymax=M+2*SE),
+                   width=.1,size=k)+
+     xlab("pron")+
+     ylab("negative reciprocal reading time")+
+     scale_colour_hue(name="relative clause status",
+                      # Legend label, use darker colors
+                      breaks=c("-RC", "+RC"),
+                      labels=c("-RC", "+RC"),
+                      l=40)+
+     opts(title="Critical region") +
+ #    scale_y_continuous(limits=c(600,900)) +
+     theme_bw() +
+     opts(legend.position=c(.87, .6)) # Position legend inside
>                                     # This must go after theme_bw
>
> #p1a
>
> p2<-ggplot(subset(M.id.w,region=="post.crit"), aes(x=pron, y=M,
+ group=rel.cl)) +
+     geom_point(position=pd,shape=21,fill="white",size=k*3) +
+     geom_line(position=pd,aes(linetype=rel.cl),size=k) +
+     geom_errorbar(aes(ymin=M-SE, ymax=M+SE),
+                   width=.1,position=pd,size=k)+
+     xlab("pron")+
```

```
+     ylab("reading time (ms)")+
+     scale_colour_hue(name="relative clause status",
+                      # Legend label, use darker colors
+                      breaks=c("-RC", "+RC"),
+                      labels=c("-RC", "+RC"),
+                      l=40)+
+     opts(title="Reading times at the post-critical region") +
+ #    scale_y_continuous(limits=c(600,900)) +
+     theme_bw() +
+     opts(legend.position=c(.87, .6)) # Position legend inside
>                                     # This must go after theme_bw
>
> #p2
>
> p2a<-ggplot(subset(M.id.w,region=="post.crit"),
+             aes(x=pron, y=M,
+ group=rel.cl)) +
+     geom_point(shape=21,fill="white",size=k*3) +
+     geom_line(aes(linetype=rel.cl),size=k) +
+     geom_errorbar(aes(ymin=M-2*SE, ymax=M+2*SE),
+                   width=.1,size=k)+
+     xlab("pronoun")+
+     ylab("negative reciprocal reading time")+
+     scale_colour_hue(name="relative clause status",
+                      # Legend label, use darker colors
+                      breaks=c("-RC", "+RC"),
+                      labels=c("-RC", "+RC"),
+                      l=40)+
+     opts(title="Post-critical region") +
+ #    scale_y_continuous(limits=c(600,900)) +
+     theme_bw() +
+     opts(legend.position=c(.87, .6)) # Position legend inside
>                                     # This must go after theme_bw
>
> #p2a
> #library(gridExtra)
>
> #grid.arrange(p1a,p2a,nrow=2)
```

2 Bayesian analysis

The dependent variable (negative reciprocal reading times) are assumed to be generated by the model:

$$Y_i = \beta_0 + b_i + c_i + \beta_1 Pronoun_i + \beta_2 RC_i + \epsilon_i \quad (1)$$

where $b_i \sim N(0, \sigma_b), c_i \sim N(0, \sigma_c), \epsilon_i \sim (0, \sigma)$. In the model, we define priors for each of the β parameters, and for b_i, c_i, ϵ. Each of the β have priors $N(0, \sqrt{1e+05})$; the priors for the varying intercepts for subject and item standard deviations are $Unif(0, 10)$; and the prior for the residual standard deviation is $Unif(0, 100)$. The code accompanying this paper provides more details regarding the fitting procedure.

The end-product of the analysis is (among other things) the posterior distribution of the parameters. For example, we can examine the posterior distribution of the parameter β_1, and calculate the probability that this is less than or greater than zero. This is the probability of the parameter given the data, we can abbreviate this as P(parameter| data). This seems like a much more useful probability than that provided by a frequentist analysis, which is the p-value, P(data| parameter). We use the posterior distribution for drawing inferences.

We have the data, and the lmer output from the varying intercepts models:

```
> #head(data.final)
> m.crit0

Linear mixed model fit by REML ['lmerMod']
Formula: -1000/rt ~ pron + rc + inter + (1 | subj) + (1 | item)
   Data: subset(data.final, region == "crit")
REML criterion at convergence: 3216.5
Random effects:
 Groups   Name        Std.Dev.
 subj     (Intercept) 0.416
 item     (Intercept) 0.318
 Residual             0.592
Number of obs: 1631, groups: subj, 110; item, 15
Fixed Effects:
(Intercept)         pron           rc        inter
    -1.5490      -0.0410      -0.0337      -0.0259

> m.post.crit0

Linear mixed model fit by REML ['lmerMod']
Formula: -1000/rt ~ pron + rc + inter + (1 | subj) + (1 | item)
   Data: subset(data.final, region == "post.crit")
REML criterion at convergence: 2935.4
Random effects:
 Groups   Name        Std.Dev.
 subj     (Intercept) 0.403
 item     (Intercept) 0.207
 Residual             0.543
Number of obs: 1631, groups: subj, 110; item, 15
Fixed Effects:
(Intercept)         pron           rc        inter
    -1.7938      -0.0609       0.0060      -0.0325
```

I will fit linear mixed models and then the JAGS models.

3 Critical region

We fit the model with varying intercepts for item and subject:

```
> m1<-m.crit0
> ## estimated sd of varying intercept:
> sigma.u<-attr(VarCorr(m1)$subj,"stddev")
> sigma.w<-attr(VarCorr(m1)$item,"stddev")
> ## estimated residual sd:
> sigma.e<-attr(VarCorr(m1),"sc")
> ## fixed effects:
> beta<-fixef(m1)
```

```
> crit.data<-subset(data.final,region=="crit" & rt>250)
> crit.data$region<-factor(crit.data$region)
> crit.data$rrt<- -1000/crit.data$rt
> crit.dat <- list( subj = sort(as.integer( factor(crit.data$subj) )),
+                       item = sort(as.integer( factor(crit.data$item) )),
+                     rrt = crit.data$rrt,
+                      pron = crit.data$pron,
+                      rc = crit.data$rc,
+                      inter = crit.data$inter,
+                      N = nrow(crit.data),
+                      I = length( unique(crit.data$subj) ),
+                      K = length( unique(crit.data$item) )
+                        )
```

Set up four chains:

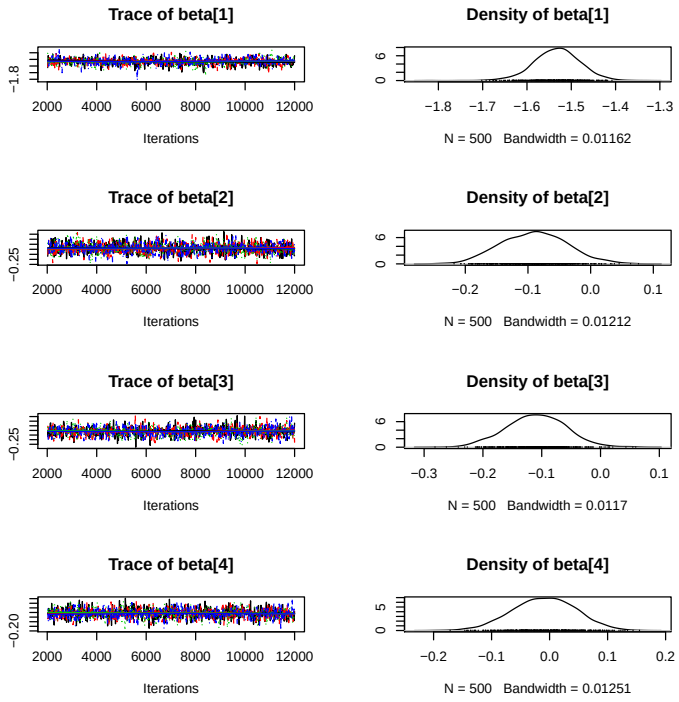
```
> crit.ini <- list( list( sigma.e = sigma.e/3,
+                           sigma.u = sigma.u/3,
+                           sigma.w = sigma.w/3,
+                              beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                           sigma.u = sigma.u*3,
+                           sigma.w = sigma.w*3,
+                              beta = beta   *3 ),
+                     list( sigma.e = sigma.e/3,
+                           sigma.u = sigma.u*3,
+                           sigma.w = sigma.w*3,
+                              beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                           sigma.u = sigma.u/3,
+                           sigma.w = sigma.w/3,
+                              beta = beta   *3 ) )
```

The JAGS model below assumes that we have negative reciprocal rts.

```
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( pron[j] )  +
+               beta[3] * ( rc[j] )  + beta[4] * ( inter[j] )  +
+               u[subj[j]] + w[item[j]]
+      rrt[j] ~ dnorm( mu[j], tau.e )
+      }
+
+    # Random effects for each person
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+      beta[3] ~ dnorm(0.0,1.0E-5)
+      beta[4] ~ dnorm(0.0,1.0E-5)
+
+
+    # Residual (within-person) variance
+      tau.e <- pow(sigma.e,-2)
+      sigma.e  ~ dunif(0,100)
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,10)
+
+      # Between-item variation
```

```
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,10)
+    }",
+      file="critcrossedrandom.jag" )
> track.variables<-c("beta","sigma.e","sigma.u","sigma.w")
> library(rjags)
> system.time(
+ crit.mod <- jags.model( file = "critcrossedrandom.jag",
+                           data = crit.dat,
+                       n.chains = 4,
+                          inits = crit.ini,
+                        n.adapt =2000 ))

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 10044

Initializing model

   user  system elapsed
 12.206   0.030  12.732

> system.time(
+ crit.res <- coda.samples(crit.mod,
+                           var = track.variables,
+                           n.iter = 10000,
+                           thin = 20))

   user  system elapsed
 56.887   0.078  57.455

> summary( crit.res )$statistics[,c(1,2)]

              Mean       SD
beta[1] -1.5321824 0.052039
beta[2] -0.0899325 0.052271
beta[3] -0.1096034 0.050474
beta[4] -0.0094805 0.053988
sigma.e  0.6850404 0.012589
sigma.u  0.2421809 0.028334
sigma.w  0.1620149 0.052051

>

> post<-jags.samples(crit.mod,
+                    var=track.variables,
```

```
+                     n.iter=10000)
> ## pronoun:
> ## chain 1:
> counts<-table(post$beta[2,,][,1]<0)
> 100*counts[2]/(sum(counts))

 TRUE
95.95

> ## checks out: P(theta<0):
> pnorm(0,mean=-0.0892,sd=0.0517)

[1] 0.95777

> ## rc:
> #hist(post$beta[3,,][,4])
> median(post$beta[3,,][,4])

[1] -0.10944

> counts<-table(post$beta[3,,][,4]<0)
> 100*counts[2]/(sum(counts))

 TRUE
98.15

> ## interaction:
> hist(post$beta[4,,][,4])
> median(post$beta[4,,][,4])

[1] -0.010872

> counts<-table(post$beta[4,,][,4]<0)
> 100*counts[2]/(sum(counts))

 TRUE
58.03

>

> par( mfrow=c(3,3) )
> plot(crit.res)
```



4 Post-critical region

We fit the model with varying intercepts for item and subject:

```
> m1<-m.post.crit0
> ## estimated sd of varying intercept:
> (sigma.u<-attr(VarCorr(m1)$subj,"stddev"))

(Intercept)
     0.4033

> (sigma.w<-attr(VarCorr(m1)$item,"stddev"))

(Intercept)
    0.20736

> ## estimated residual sd:
> (sigma.e<-attr(VarCorr(m1),"sc"))

[1] 0.54287

> (beta<-fixef(m1))
```

```
(Intercept)        pron          rc       inter
 -1.7937935  -0.0608707   0.0059969  -0.0325102
```

We will just call the post-critical region data crit.data.

```
> crit.data<-subset(data.final,region=="post.crit" & rt>250)
> crit.data$region<-factor(crit.data$region)
> crit.data$rrt<- -1000/crit.data$rt
> crit.dat <- list( subj = sort(as.integer( factor(crit.data$subj) )),
+                       item = sort(as.integer(factor(crit.data$item))),
+                     rrt = crit.data$rrt,
+                      pron = crit.data$pron,
+                      rc = crit.data$rc,
+                      inter = crit.data$inter,
+                      N = nrow(crit.data),
+                      I = length( unique(crit.data$subj) ),
+                      K = length( unique(crit.data$item) )
+                        )
```

Set up four chains:

```
> crit.ini <- list( list( sigma.e = sigma.e/3,
+                           sigma.u = sigma.u/3,
+                           sigma.w = sigma.w/3,
+                              beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                           sigma.u = sigma.u*3,
+                           sigma.w = sigma.w*3,
+                              beta = beta   *3 ),
+                     list( sigma.e = sigma.e/3,
+                           sigma.u = sigma.u*3,
+                           sigma.w = sigma.w*3,
+                              beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                           sigma.u = sigma.u/3,
+                           sigma.w = sigma.w/3,
+                              beta = beta   *3 ) )
```

The JAGS model below assumes that we have negative reciprocal rts.

```
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
```

```
+      {
+      mu[j] <- beta[1] + beta[2] * ( pron[j] )  +
+               beta[3] * ( rc[j] )  + beta[4] * ( inter[j] )  +
+               u[subj[j]] + w[item[j]]
+      rrt[j] ~ dnorm( mu[j], tau.e )
+      }
+
+    # Random effects for each person
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+      beta[3] ~ dnorm(0.0,1.0E-5)
+      beta[4] ~ dnorm(0.0,1.0E-5)
+
+
+    # Residual (within-person) variance
+      tau.e <- pow(sigma.e,-2)
+      sigma.e  ~ dunif(0,100)
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,10)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,10)
+    }",
+      file="postcritcrossedrandom.jag" )
> track.variables<-c("beta","sigma.e","sigma.u","sigma.w")
> library(rjags)
> system.time(
+ crit.mod <- jags.model( file = "postcritcrossedrandom.jag",
+                         data = crit.dat,
```

```
+                       n.chains = 4,
+                          inits = crit.ini,
+                        n.adapt =2000 ))

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 10068

Initializing model

   user  system elapsed
 13.486   0.076  17.225

> system.time(
+ crit.res <- coda.samples(crit.mod,
+                           var = track.variables,
+                           n.iter = 10000,
+                           thin = 20))

   user  system elapsed
 56.817   0.124  58.724

> summary( crit.res )$statistics[,c(1,2)]

              Mean       SD
beta[1] -1.7810606 0.044752
beta[2] -0.0698208 0.040151
beta[3] -0.0091506 0.041538
beta[4] -0.0171780 0.043191
sigma.e  0.6595543 0.012015
sigma.u  0.1363709 0.029054
sigma.w  0.1415816 0.040077
```

Compute probabilities:

```
> post<-jags.samples(crit.mod,
+                    var=track.variables,
+                    n.iter=10000)
> ## pronoun:
> ## chain 1:
> #hist(post$beta[2,,][,1])
> counts<-table(post$beta[2,,][,1]<0)
> 100*counts[2]/(sum(counts))

 TRUE
96.45
```

```
> ## rc:
> #hist(post$beta[3,,][,1])
> #median(post$beta[3,,][,  ])
> counts<-table(post$beta[3,,][,1]<0)
> 100*counts[2]/(sum(counts))

 TRUE
57.69

> ## interaction:
> #hist(post$beta[4,,][,4])
> #median(post$beta[4,,][,4])
> counts<-table(post$beta[4,,][,1]<0)
> 100*counts[2]/(sum(counts))

 TRUE
65.37

>

> par( mfrow=c(3,3) )
> plot(crit.res)
```