SVA Simulation

Christopher Lo

9/6/2022

Simulation Set-Up

Simulation studies recreated to my best abilities from "A general framework for multiple testing dependence" (Leek et al. 2008)

We generate X from the following model: $X = BS + \Gamma G + U$.

We have 1000 genes, 20 samples, and 2 latent variables. The design matrix S is a case control set-up of 10 cases and 10 controls.

$$\sigma_i^2 \sim InvGamma(10, 9)$$

$$U_{ij} \sim N(0, \sigma_i^2)$$
$$b_{m,1} \sim N(0, 1), m = 1, ..., 1000$$
$$b_{m,2} \sim N(0, 3.1), m = 1, ..., 300, b_{m,2} \sim N(2.5, 1), m = 301, ..., 1000$$

$$\Gamma_{m,1} \sim N(0, 2.5), m = 301, ..., 700$$
$$\Gamma_{m,2} \sim N(0, 2.5), m = 501, ..., 900$$

$$G_{1:10} \sim Bernoulli(.7)$$
$$G_{11:20} \sim Bernoulli(.2)$$

Run SVA and regression to estimate parameters and SVs

We look at the number of SVs estimated, whether the latent variables are spanned by the estimated SVs, the recovered regression coefficients, the null p-value distribution, and the ranks of top genes.

Todo: ranks of top genes code unsure right now.

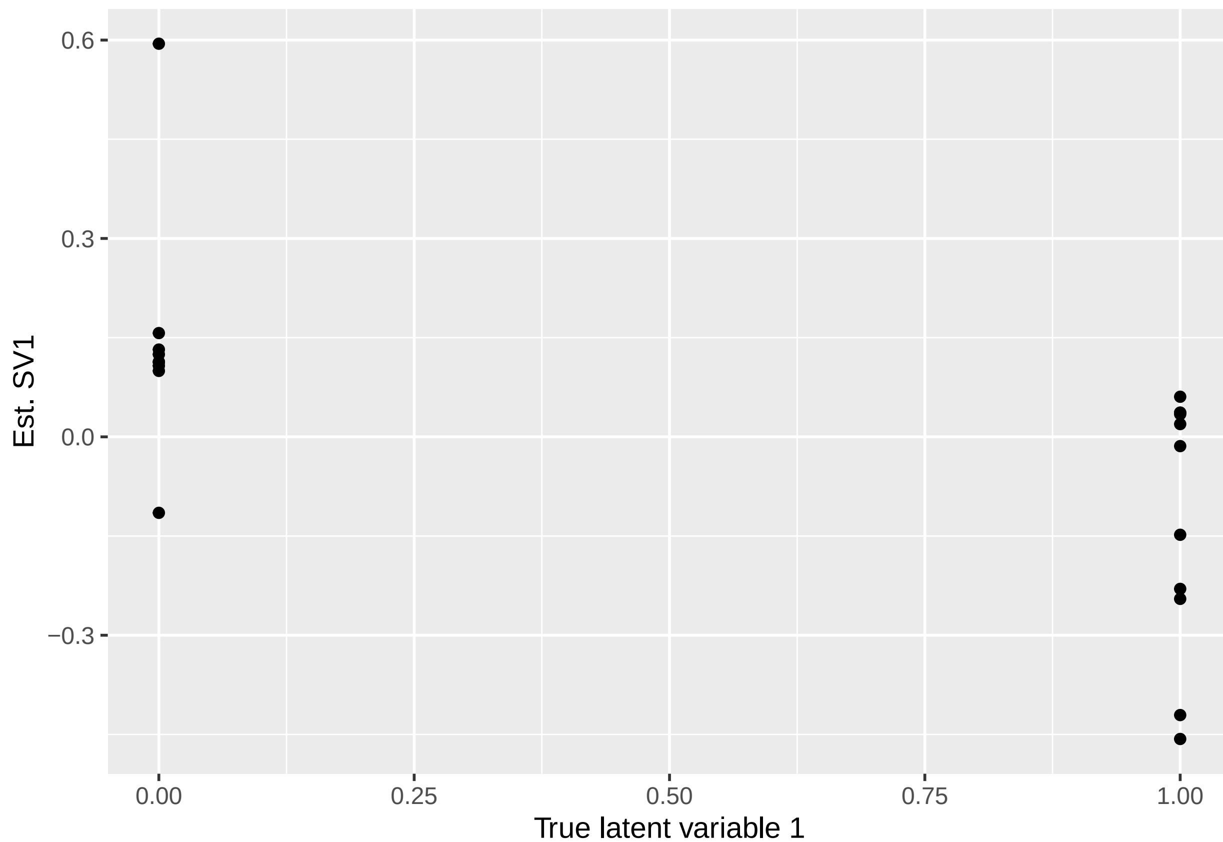
```
nullMod = t(S)[, 1]
n.sv = num.sv(X, t(S), method = "be")
cat("Number of SVs: ", n.sv, "\n")
```

```
## Number of SVs:  2
```

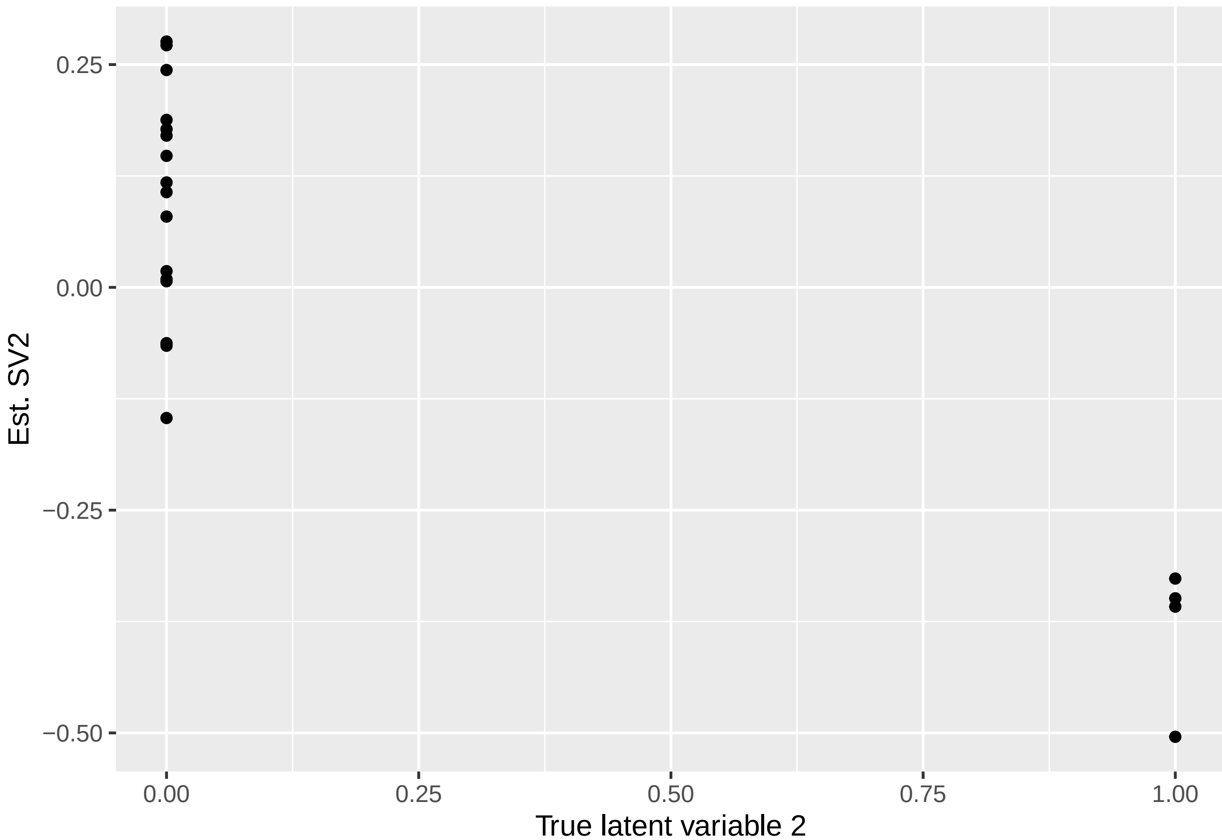
```r
svobj = sva(X, t(S), nullMod, n.sv = n.sv)
```

```
## Number of significant surrogate variables is:  2
## Iteration (out of 5 ):1  2  3  4  5
```

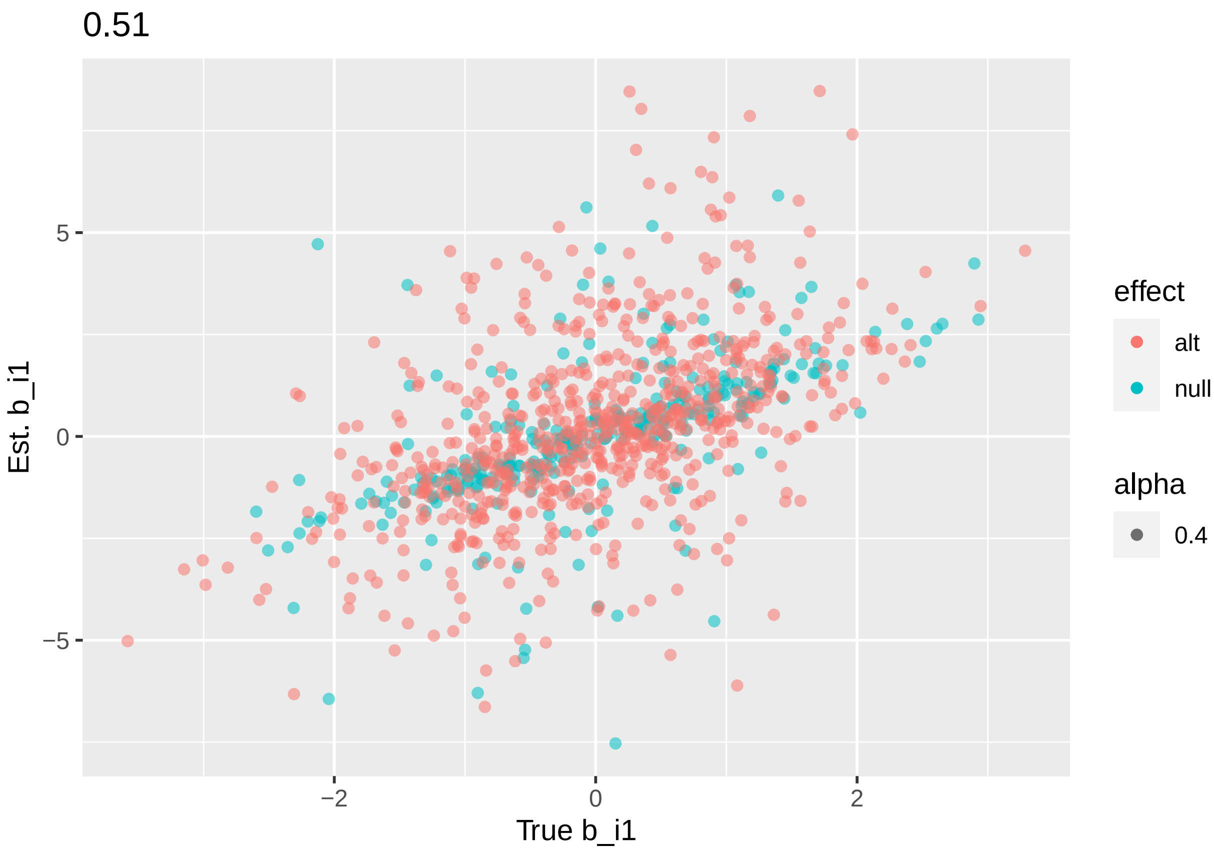
```r
#visually look at predicted SVs.
qplot(as.numeric(G[1 ,]), svobj$sv[, 1], xlab = "True latent variable 1", ylab = "Est. SV1")
```



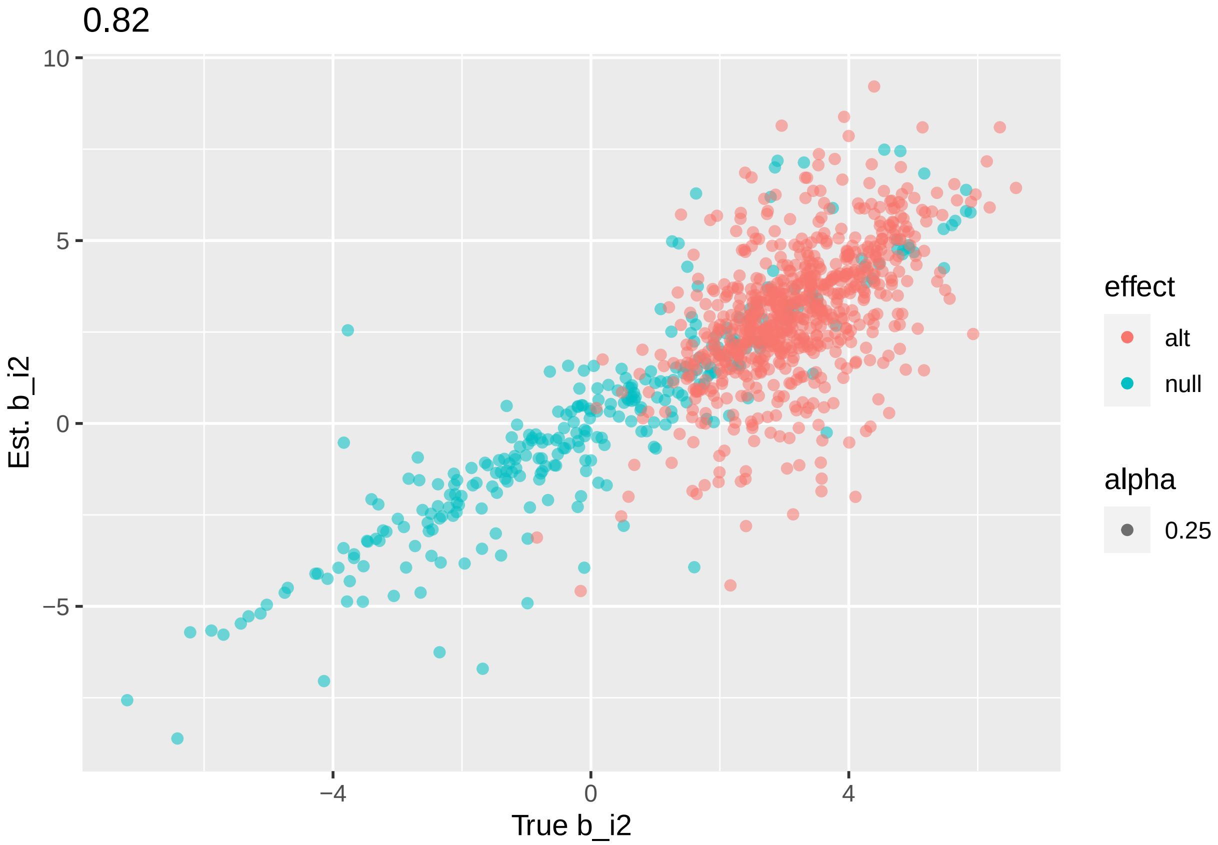

```r
qplot(as.numeric(G[2 ,]), svobj$sv[, 2], xlab = "True latent variable 2", ylab = "Est. SV2")
```



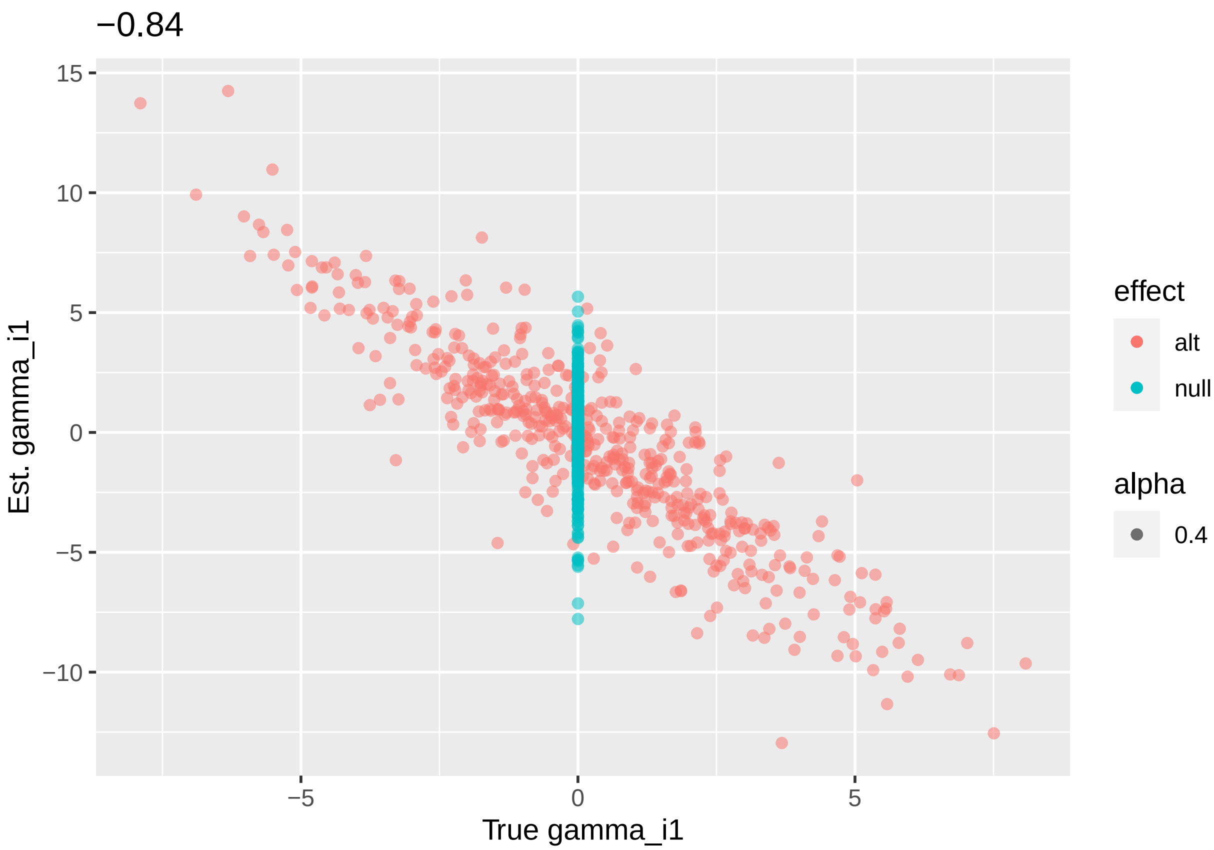

```
nullmodsv = cbind(nullMod, svobj$sv)
modsv = cbind(t(S), svobj$sv)
#run full regression.
fitsv = lm.fit(modsv, t(X))

#visually look at predicted coefficients
effect = c(rep("null", 300), rep("alt", m - 300))
qplot(B[, 1], fitsv$coefficients[1 ,], color = effect, main = round(cor(fitsv$coefficients[1 ,], B[, 1])
```



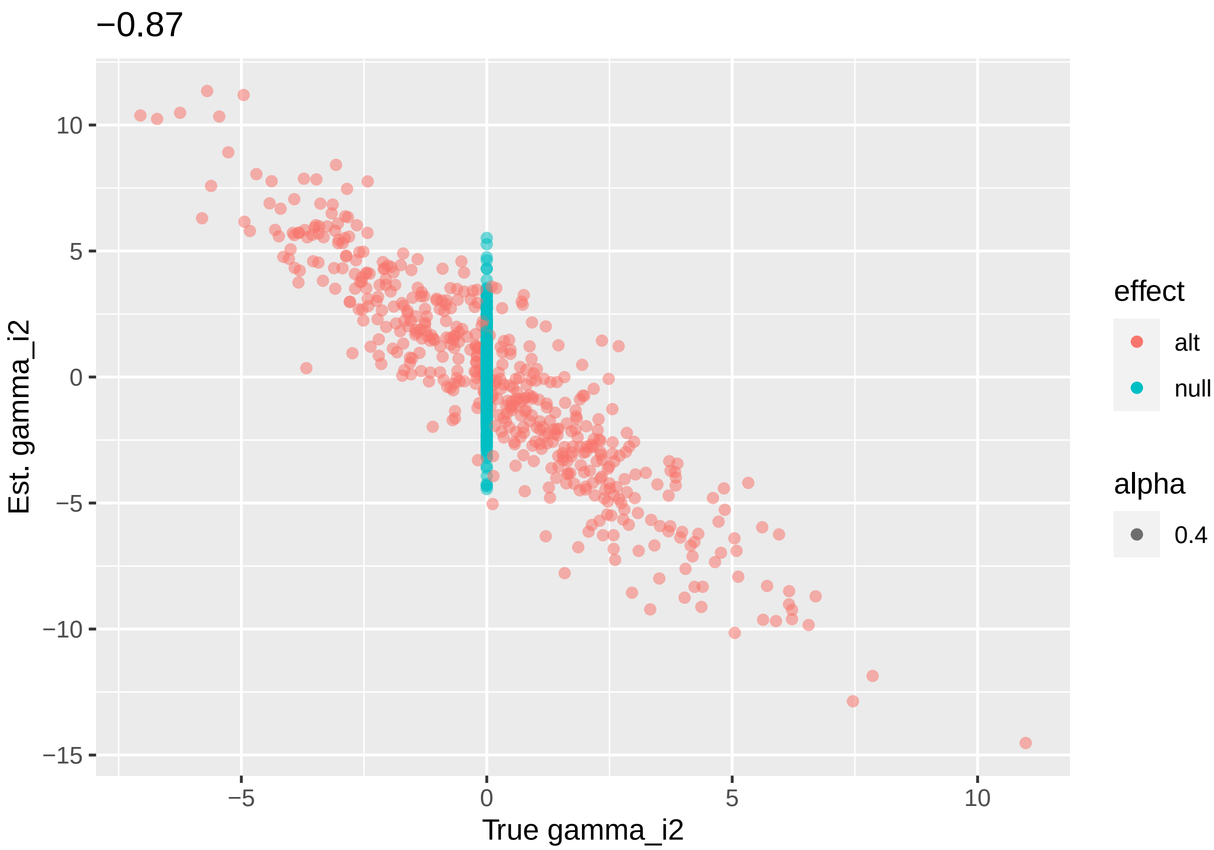

```
qplot(B[, 2], fitsv$coefficients[2 ,], color = effect, main = round(cor(B[, 2], fitsv$coefficients[2 ,]
```



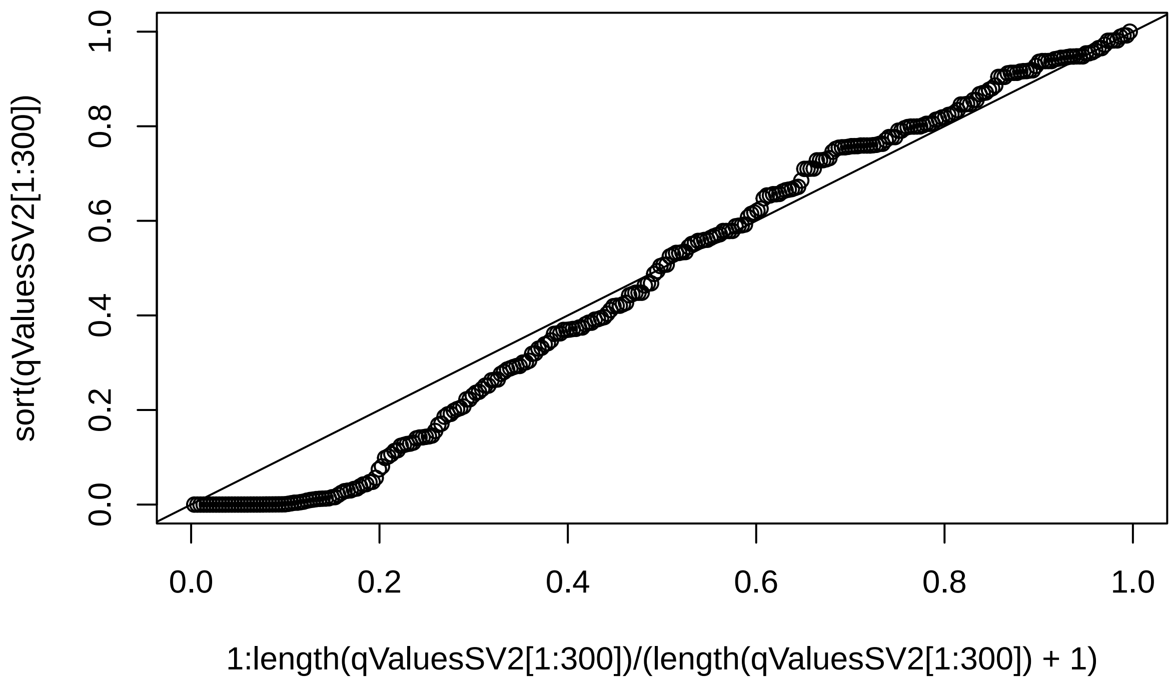

```
effect = c(rep("null", 200), rep("alt", 500), rep("null", 300))
qplot(Gamma[, 1], fitsv$coefficients[3 ,], color = effect, main = round(cor(Gamma[, 1], fitsv$coefficien
```




```
effect = c(rep("null", 400), rep("alt", 500), rep("null", 100))
qplot(Gamma[, 2], fitsv$coefficients[4 ,], color = effect, main = round(cor(Gamma[, 2], fitsv$coefficie
```




```
#just compute p-value of b_12 = 0 using F statistics.
pValuesSv = f.pvalue(X, modsv, nullmodsv)
#double check with existing function:
pValuesSV2 = rep(NA, 1000)
fstat = rep(NA, 1000)
for(i in 1:1000) {
  dat = data.frame(x = X[i ,], pv = S[2 ,], sv1 = svobj$sv[, 1], sv2 = svobj$sv[, 2])
  dat_nullmod = lm(x ~ pv, dat)
  dat_mod = lm(x ~ pv + sv1 + sv2, dat)
  an = anova(dat_nullmod, dat_mod)
  pValuesSV2[i] = an$`Pr(>F)`[2]
  fstat[i] = an$F[2]
}


qValuesSv = p.adjust(pValuesSv, method = "BH")
qValuesSV2 = p.adjust(pValuesSV2, method = "BH")


plot(1:length(qValuesSV2[1:300])/(length(qValuesSV2[1:300])+1),sort(qValuesSV2[1:300]))
abline(a = 0, b = 1)
```




```
ks.test(qValuesSV2[1:300],"punif",0,1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  qValuesSV2[1:300]
## D = 0.14522, p-value = 6.396e-06
## alternative hypothesis: two-sided
```

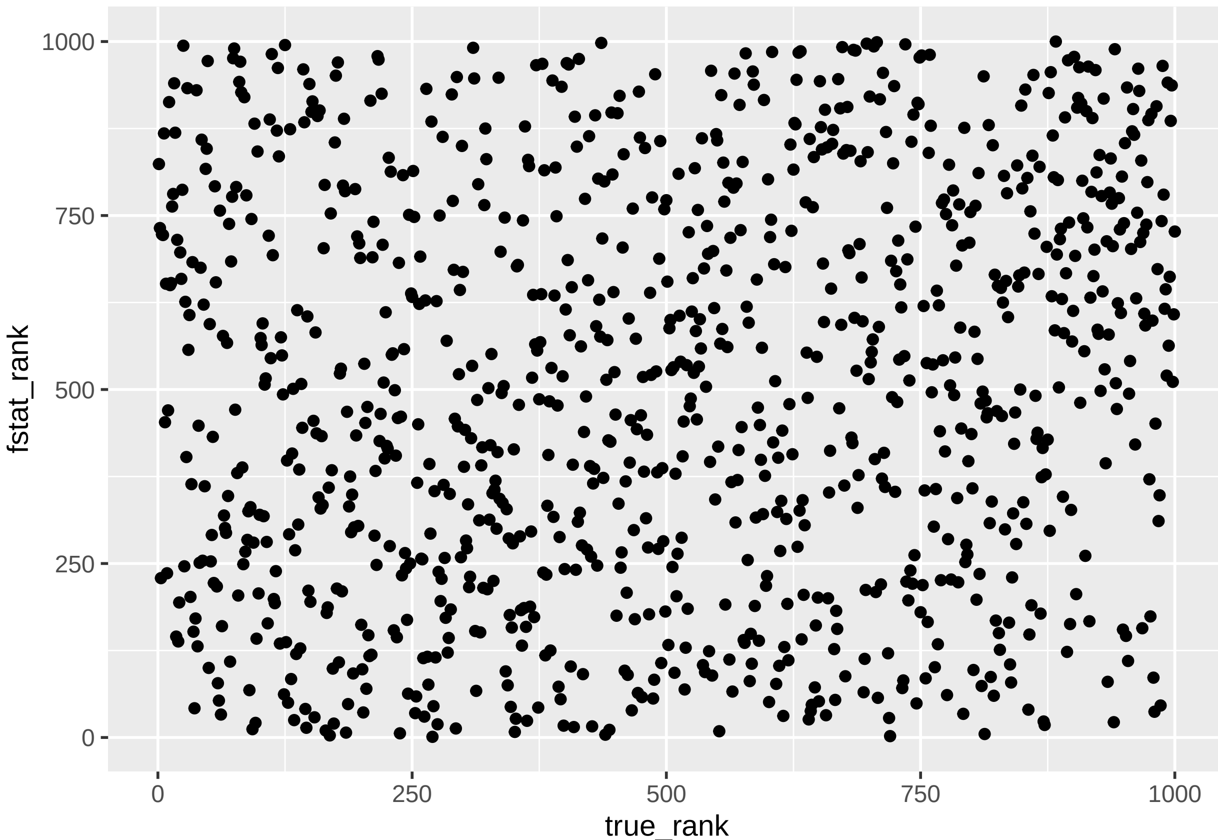
```
table(qValuesSV2[1:300] < .05)
```

```
##
## FALSE  TRUE
##   242    58
```

```
table(qValuesSV2[301:1000] < .05)
```

```
##
## FALSE  TRUE
##   315   385
```

```
fstat_rank = rank(-fstat)
true_rank = rank(-abs(b2))

qplot(true_rank, fstat_rank)
```


1000
750
500
250
0
fstat_rank
0
250
500
750
1000
true_rank