SVA Simulation

Christopher Lo

9/6/2022

Simulation studies inspired from *"A general framework for multiple testing dependence"* (Leek et al. 2008)

Simulation Set-Up, one single experiment

We generate X from the following model:

$$X = BS + \Gamma G + U$$

We have $m = 1000$ genes (tests), $n = 20$ samples, and $r = 2$ latent variables.

Sampling noise: $U_{m,n} \sim N(0, 1)$.

The design matrix S is 10 cases and 10 controls: $S_{1,n} = 1$ for $n = 1 : 20$. Then, $S_{2,n} = 0$ for $n = 1 : 10$, $S_{2,n} = 1$ for $n = 11 : 20$.

Control effect for all genes: $b_{m,1} \sim N(0, 1), m = 1 : 1000$

Case effect for DE genes $m = 1 : 300$: $b_{m,2} \sim N(3, 1)$

Case effect for Non-DE genes $m = 301 : 1000$: $b_{m,2} \sim N(0, 2)$

Latent design matrix (kernel) G: $G_{r,n} \sim Bernoulli(.2), n = 1 : 10$. $G_{r,n} \sim Bernoulli(.8), n = 11 : 20$, where $r = 1, 2$. (This ensures correlation between the two design matrices.)

Latent effect 1: $\Gamma_{m,1} \sim N(0, 1), m = 1 : 300$, $\Gamma_{m,1} \sim N(1, 2), m = 301 : 1000$. (Positive signal overlaps with Non-DE genes, will lead to FPs if not corrected)

Latent effect 2: $\Gamma_{m,2} \sim N(-1, 2), m = 1 : 300$, $\Gamma_{m,2} \sim N(0, 1), m = 301 : 1000$. (Negative signal overlaps with DE genes, will lead to FNs if not corrected)

Therefore, for every gene, whether it is DE or not, it will be affected by one of the two latent variables

To ask/consider:

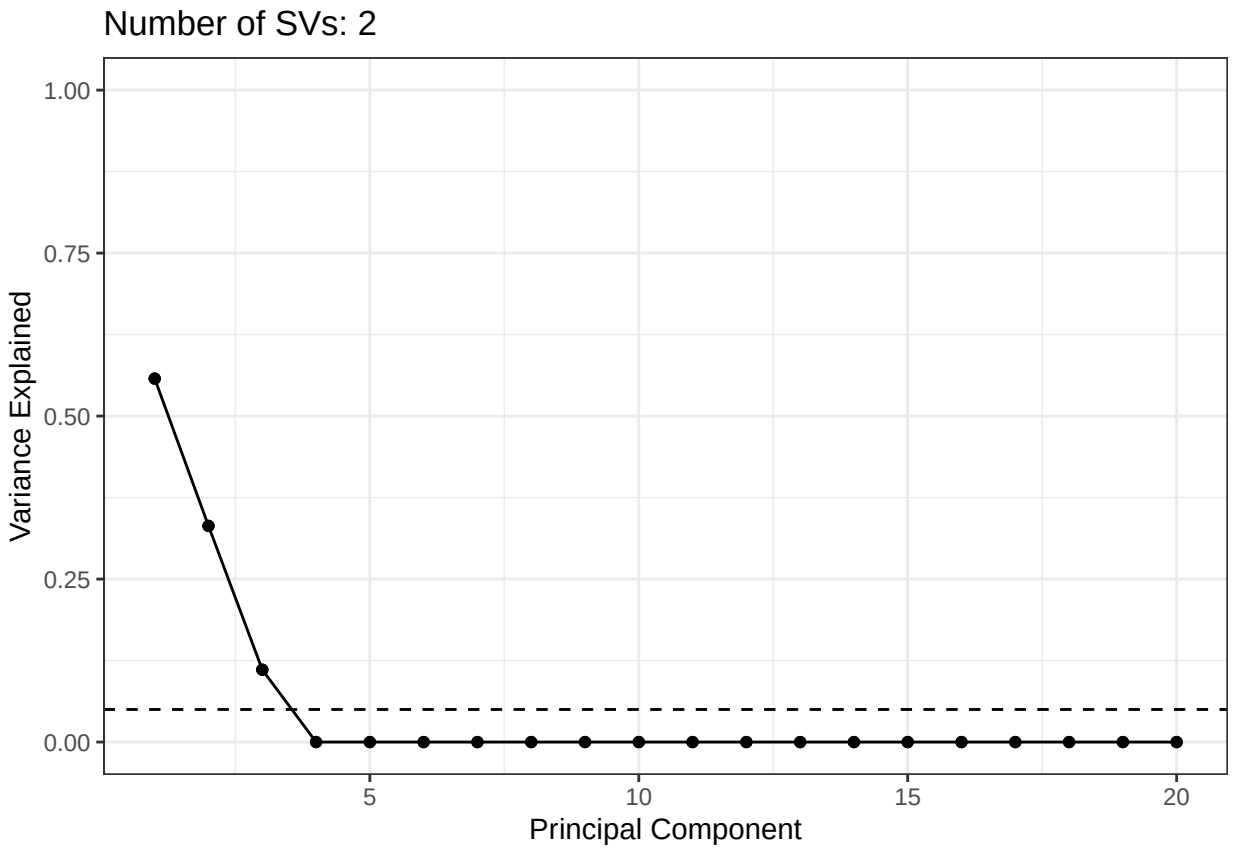
- When there's no effect, should we use $N(0, 1)$, or should we just use 0?
- Currently we have negative expression due to Latent effect 2.
- Should we normalize before running analysis?

Estimate the number of SVs:

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

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

```
pca = prcomp(t(X))
variance = pca$sdev^2 / sum(pca$sdev^2)
qplot(c(1:length(variance)), variance) + geom_line() + geom_point() +
  geom_hline(yintercept=1/ncol(X), linetype = "dashed") +
  xlab("Principal Component") + ylab("Variance Explained") + ggtitle(paste0("Number of SVs: ", n.sv)) +
```



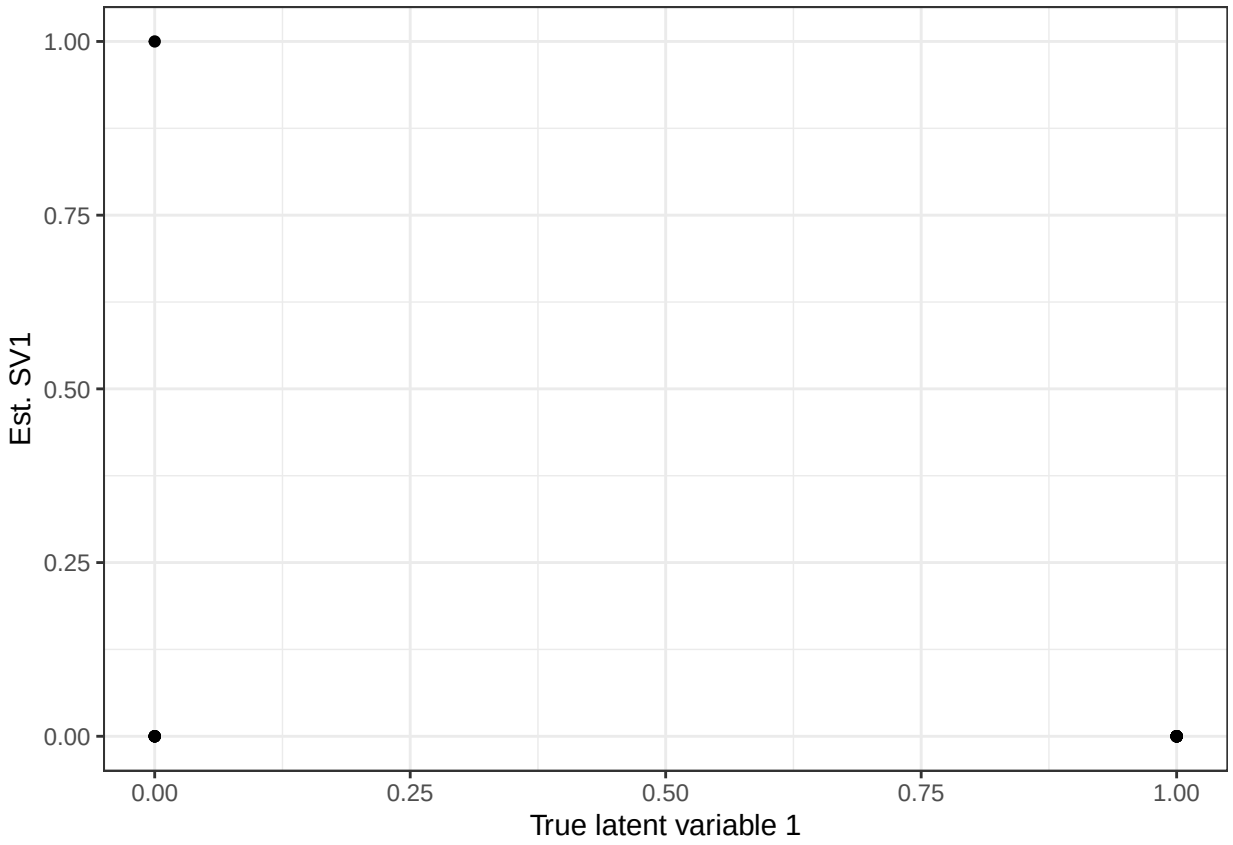
Estimate SVs, primary variable coefficients, and SV coefficients

- Are latent variables are spanned by the estimated SVs?
- Are the estimated coefficients similar to true coefficients?
- Is the null p-value distribution uniform?
- Do the ranks of top genes match?

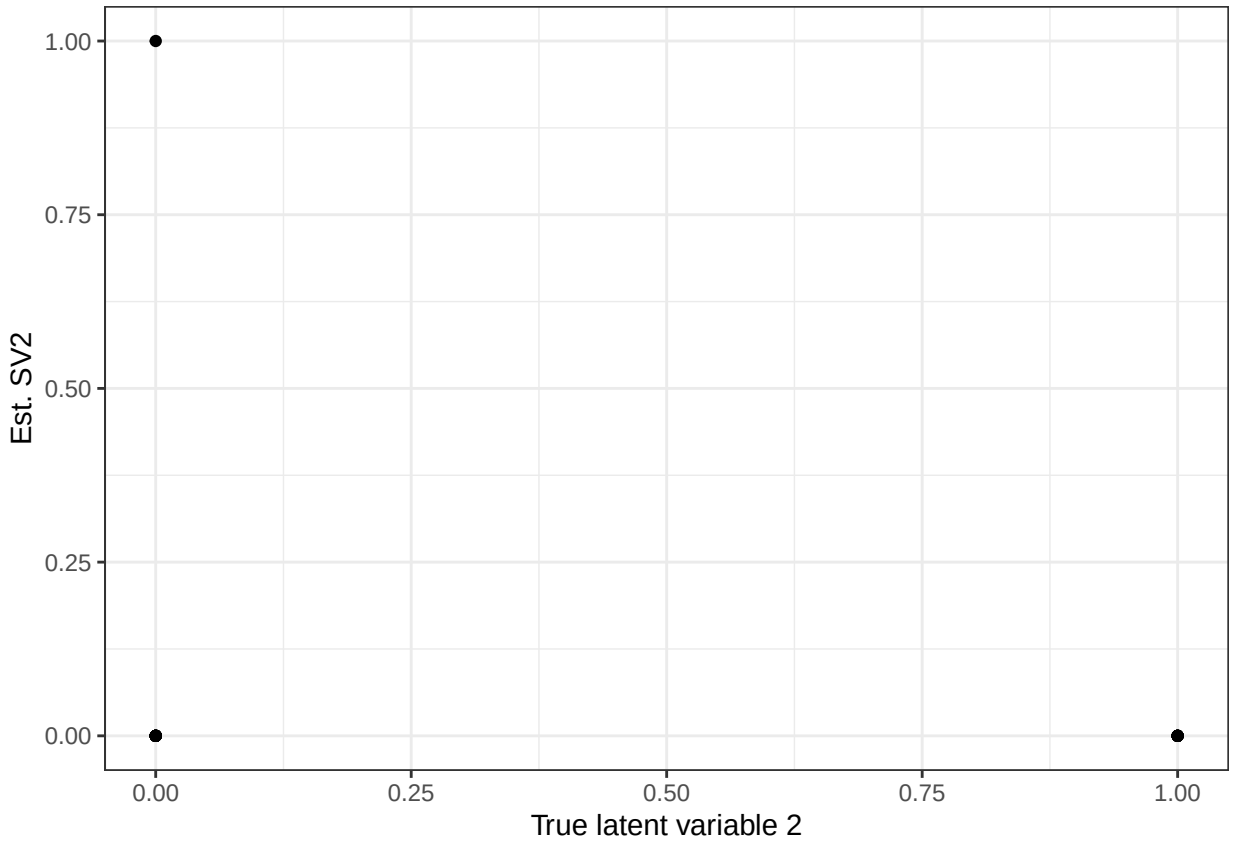
```
nullMod = t(S)[, 1]
svobj = sva(X, t(S), nullMod, n.sv = n.sv)
```

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

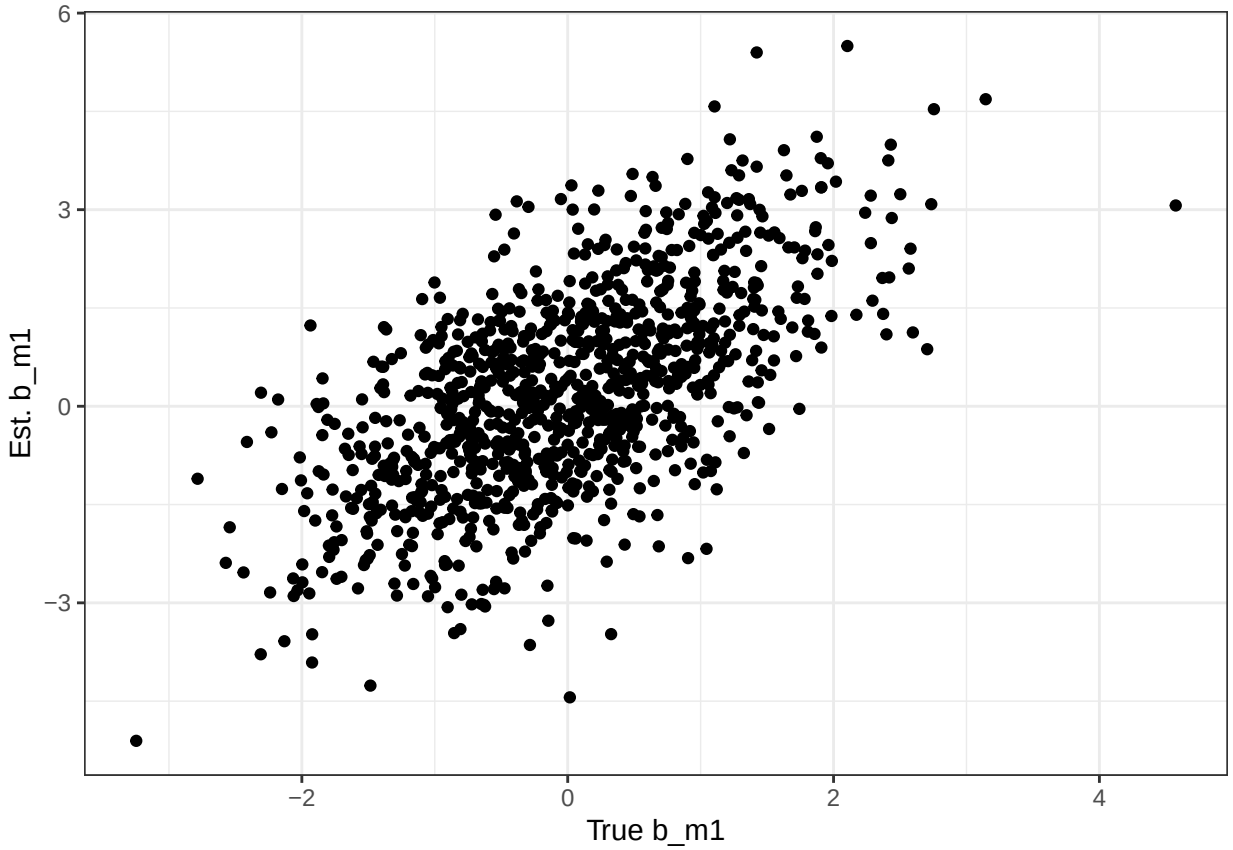
```
#visually look at predicted SVs.
qplot(as.numeric(G[1 ,]), svobj$sv[, 1], xlab = "True latent variable 1", ylab = "Est. SV1")
```



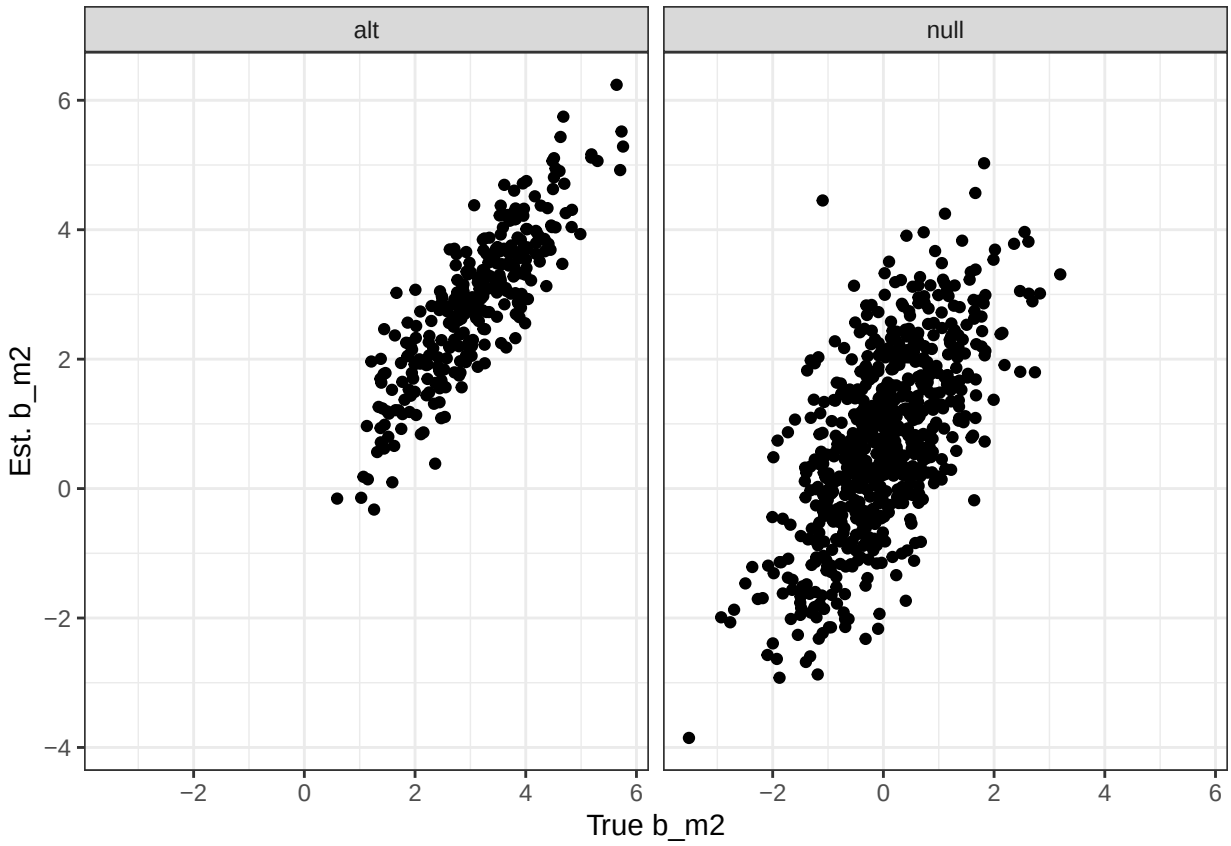

```
qplot(as.numeric(G[2 ,]), svobj$sv[, 2], xlab = "True latent variable 2", ylab = "Est. SV2")
```



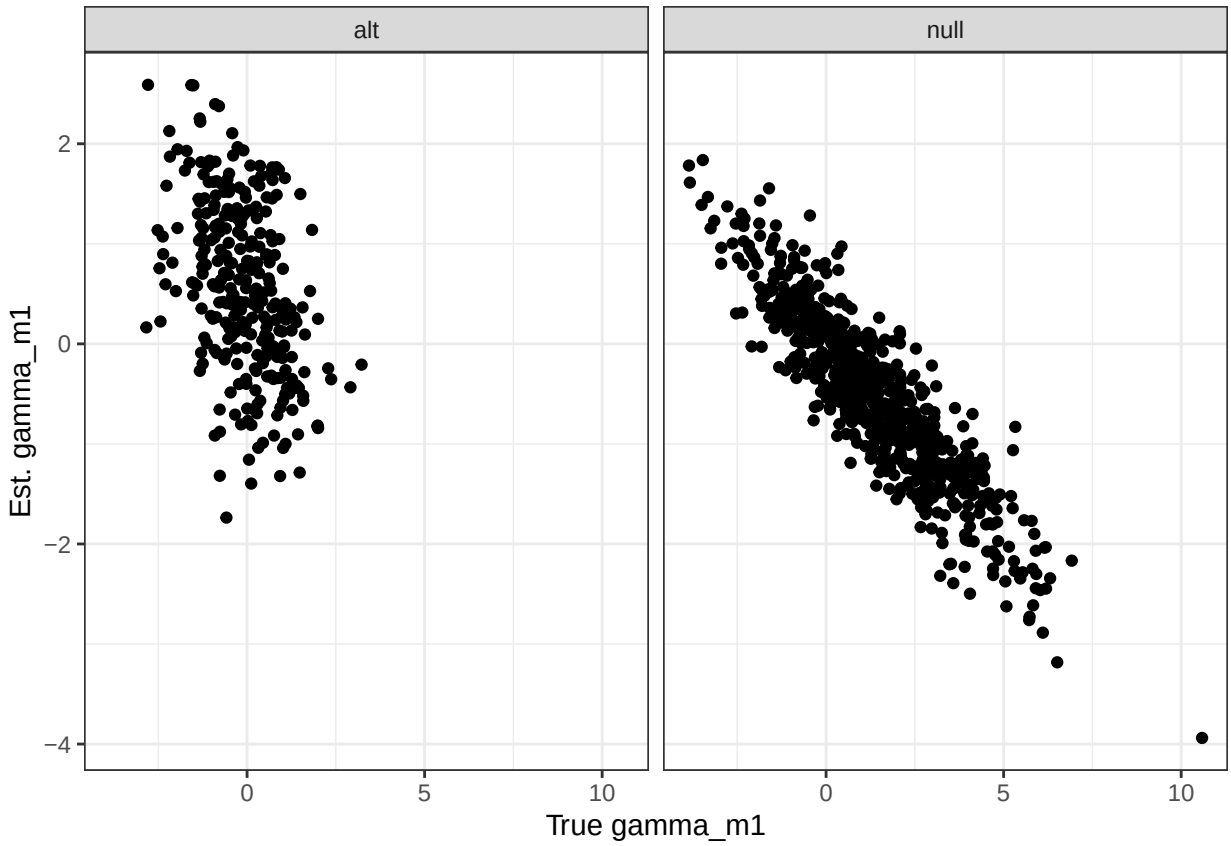

```
nullmodsv = cbind(nullMod, svobj$sv)
modsv = cbind(t(S), svobj$sv)
#run full regression.
fitsv = lm.fit(modsv, t(X))

#visually look at predicted coefficients
plot_df = data.frame(b1 = B[, 1],
                     b2 = B[, 2],
                     b1_hat = fitsv$coefficients[1 ,],
                     b2_hat = fitsv$coefficients[2 ,],
                     b2_labels = c(rep("alt", 300), rep("null", m - 300)),
                     gamma1 = Gamma[, 1],
                     gamma1_hat = fitsv$coefficients[3 ,],
                     gamma1_labels = c(rep("alt", 300), rep("null", m - 300)),
                     gamma2 = Gamma[, 2],
                     gamma2_hat = fitsv$coefficients[4 ,],
                     gamma2_labels = c(rep("alt", 300), rep("null", m - 300)))

ggplot(plot_df, aes(b1, b1_hat)) + geom_point() + labs(x = "True b_m1", y = "Est. b_m1")
```



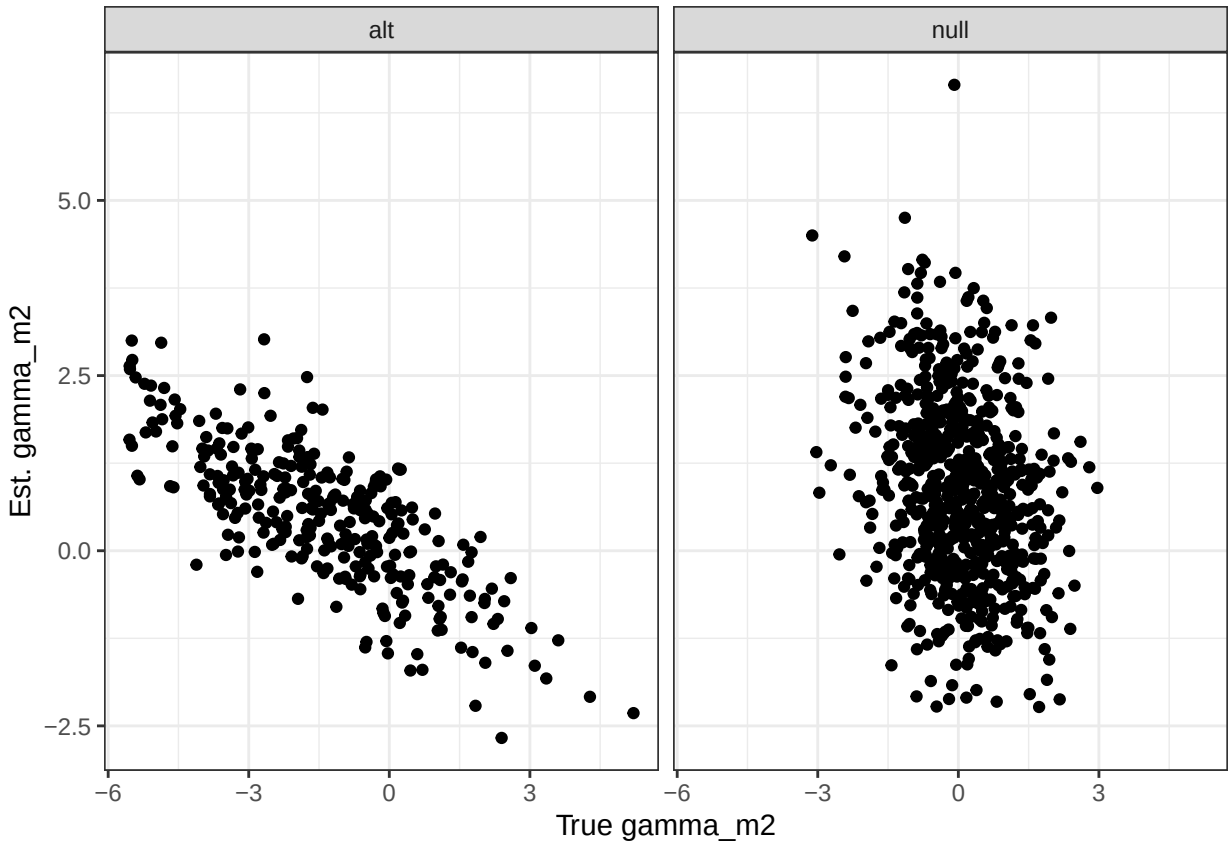

```
ggplot(plot_df, aes(b2, b2_hat)) + geom_point() + facet_wrap(~b2_labels) + labs(x = "True b_m2", y = "Es
```




```
ggplot(plot_df, aes(gamma1, gamma1_hat)) + geom_point() + facet_wrap(~gamma1_labels) + labs(x = "True g
```

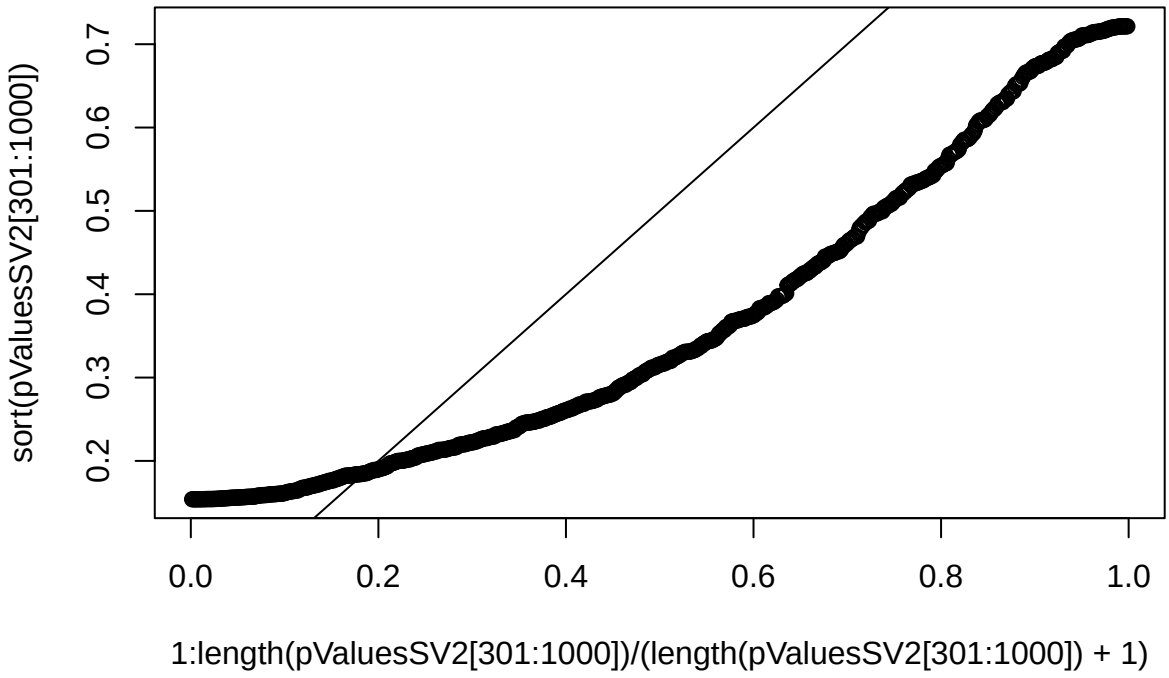


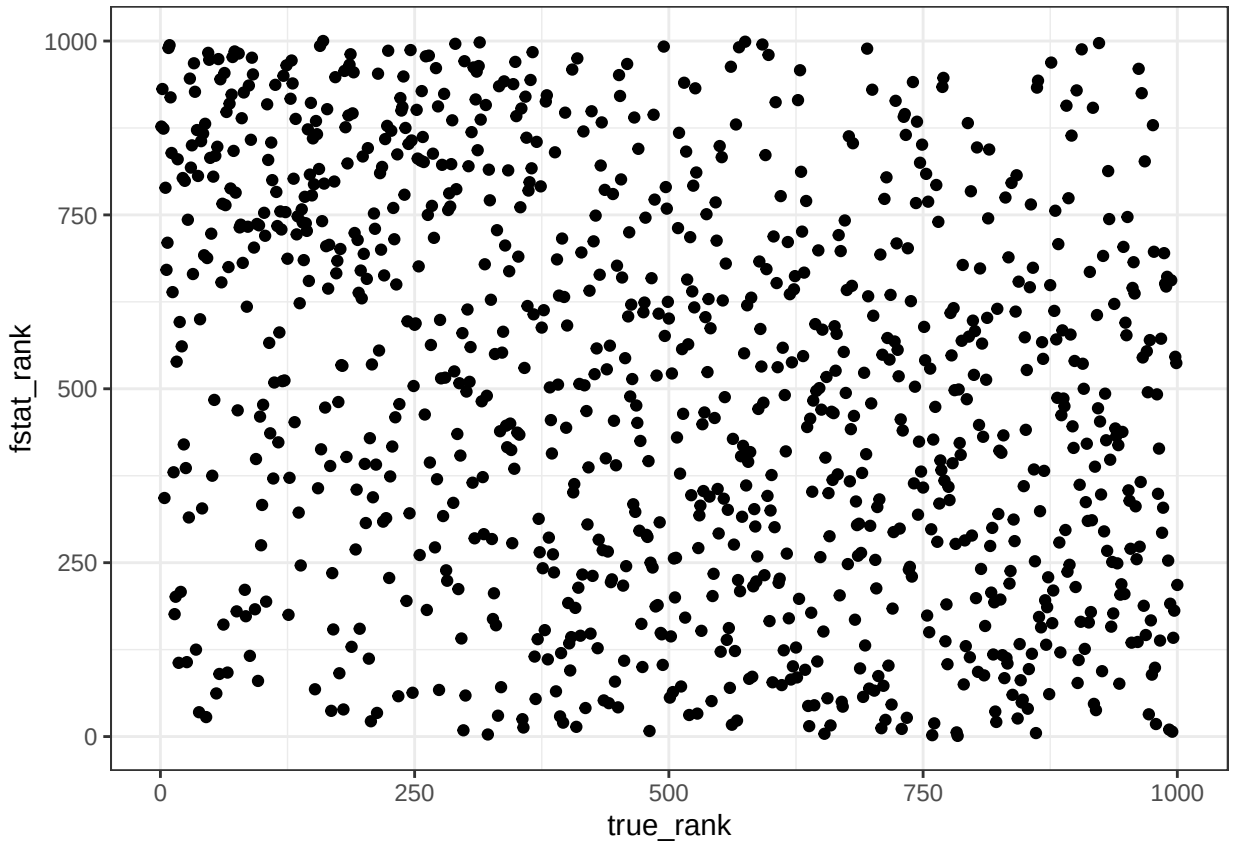

```
ggplot(plot_df, aes(gamma2, gamma2_hat)) + geom_point() + facet_wrap(~gamma2_labels) + labs(x = "True g
```



Not sure what's going on here yet regarding p-values and ranking.

- Why does `f.pvalue()` function look very different than `anova()` function?
- Why are we getting no p values $< .05$?





"Knobs to turn" in estimating the number of SVs

$\Gamma_{m,1}$: If strong effect relative to $b_{m,1}$ (fixed), then this will generate noise on control samples, leading to false positives.

$\Gamma_{m,2}$: If strong effect relative to $b_{m,2}$ (fixed), then this will generate noise on case samples, leading to false negatives.

Our certainty of Γ to effect case or control samples depends on "the percentage of row space of S explained by G". We appropx that by looking at $cor(G_r, S_2), r = 1, 2$. We probably can fix this value for now.

Knob Speculation, within one experiment

$\Gamma_{m,1}$	$\Gamma_{m,2}$	$cor(G_r, S_2)$	**DE**	**Scree plot**
strong	weak	strong	more FPs	more even PCA
weak	strong	strong	more FNs	more even PCA
weak	weak	strong	neutral	more dominated PCA
strong	strong	strong	more FPs and FNs	more even PCA

Simulation with multiple experiments

$$X = B_1 S_1 + \Gamma_1 G_1 + \alpha(B_2 S_2 + \Gamma_2 G_2) + U$$

where B_i and Γ_i are the same shape and distributions of B and Γ as before.

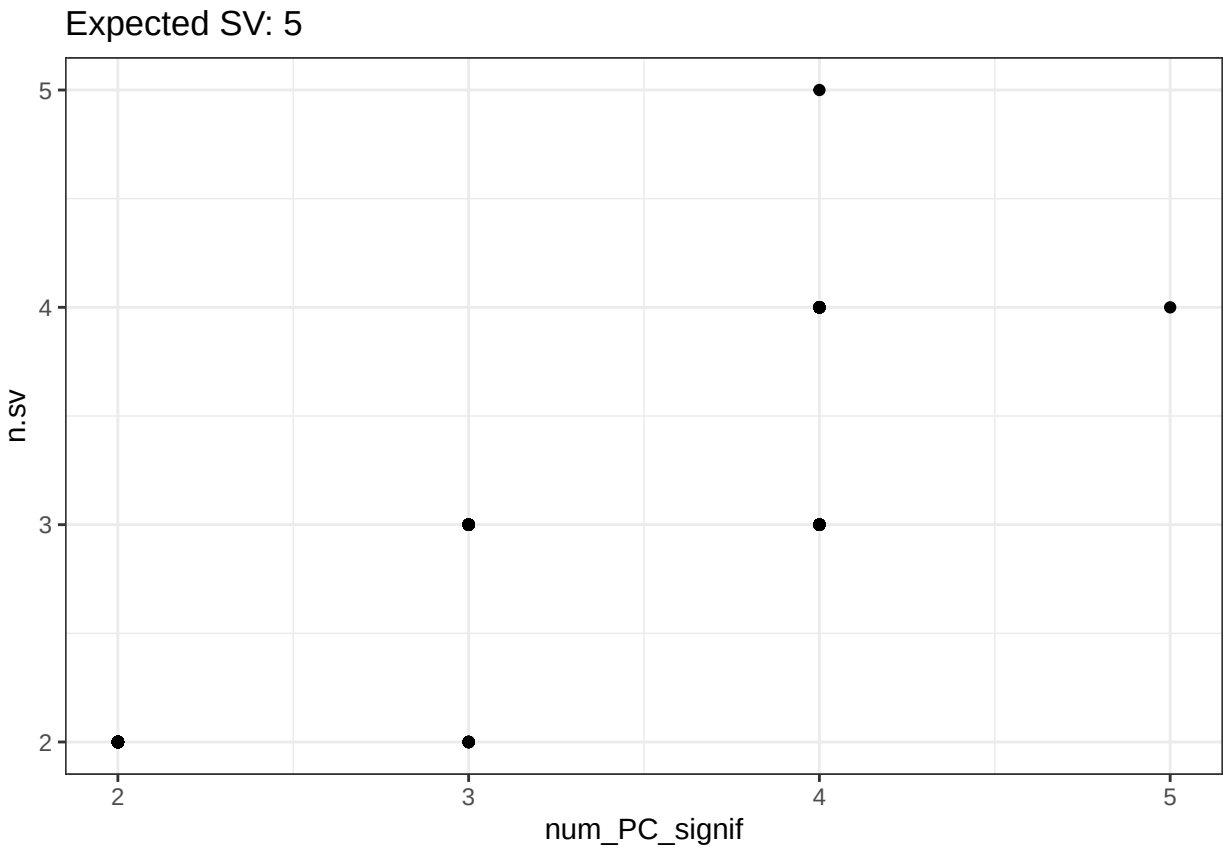
S_1 is the design matrix of the primary variables of the first experiment, elongated to 0s for the second experiment. etc.

We keep $cor(G_{ir}, S_{i2})$ at the same strength, and vary the SV effect $\Gamma_{m,1}$, $\Gamma_{m,2}$, and the study effect, α

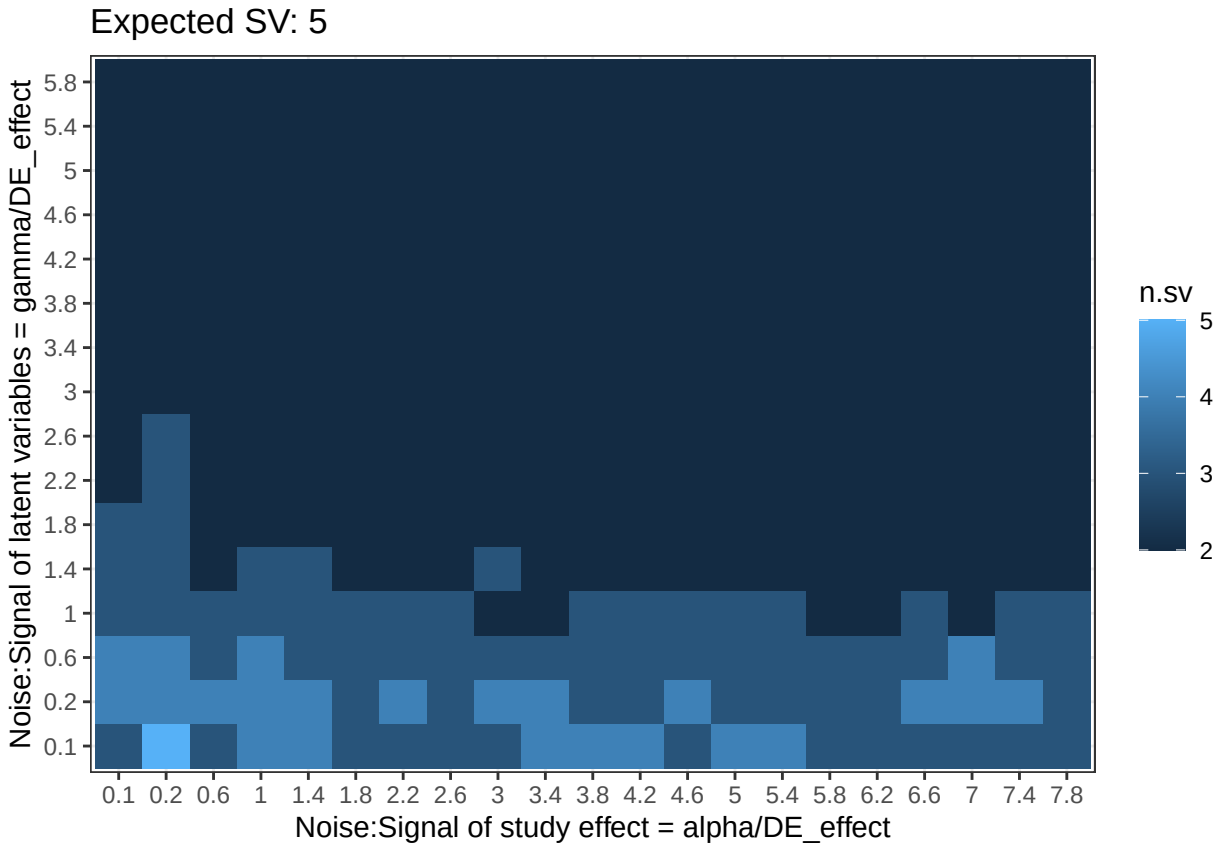
```
set.seed(2023)
m = 1000 #number of genes (tests)
n = 20 #number of samples
r = 2 #number of latent variables per studies
s = 2 #number of studies

DE_effect = 5
simulation = expand.grid(alpha = c(.5, seq(1, 40, 2)), #effect of the second experiment relative to the
                         gamma = c(.5, seq(1, 30, 2))) #effect of latent variables, relative to DE_effe
simulation$n.sv = NA
simulation$num_PC_signif = NA
simulation$var_explained_PC1 = NA

for(i in 1:nrow(simulation)) {
  alpha = simulation$alpha[i]
  gamma = simulation$gamma[i]
  #cat(alpha, "\t", gamma, "\n")

  #generate data
  U = rnorm(n * s, mean = 0, sd = 1)

  b11 = rnorm(m, mean = 0, sd = 1) #intercept
  b12 = mapply(mu = c(rep(DE_effect, 300), rep(0, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
               function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  B1 = cbind(b11, b12)
  b21 = rnorm(m, mean = 0, sd = 1)
  b22 = mapply(mu = c(rep(DE_effect, 300), rep(0, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
               function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  B2 = cbind(b21, b22)


  S1 = matrix(c(rep(1, n), rep(0, n), rep(0, n/2), rep(1, n/2), rep(0, n)), byrow = T, ncol = s*n)
  S2 = matrix(c(rep(0, n), rep(1, n), rep(0, n), rep(0, n/2), rep(1, n/2)), byrow = T, ncol = s*n)

  gamma11 = mapply(mu = c(rep(0, 300), rep(gamma, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
               function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  gamma12 = mapply(mu = c(rep(-gamma, 300), rep(0, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
               function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  Gamma1 = cbind(gamma11, gamma12)
  gamma21 = mapply(mu = c(rep(0, 300), rep(gamma, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
               function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  gamma22 = mapply(mu = c(rep(-gamma, 300), rep(0, m - 300)),
               sigma = c(rep(1, 300), rep(1, m - 300)),
```

```
                function(mu, sigma) rnorm(1, mean = mu, sd = sigma))
  Gamma2 = cbind(gamma21, gamma22)

  G_zeros = matrix(rep(0, n*2), byrow = T, ncol = n)
  G1 = cbind(mapply(p = c(rep(.2, 10), rep(.8, 10)),
             function(p) rbinom(2, 1, p)), G_zeros)
  G2 = cbind(G_zeros, mapply(p = c(rep(.2, 10), rep(.8, 10)),
             function(p) rbinom(2, 1, p)))


  X = B1 %*% S1 + Gamma1 %*% G1 + alpha * (B2 %*% S2 + Gamma2 %*% G2) + U
  S_merged = cbind(S1[, 1:n], S2[, (n+1):(2*n)])

  #Normalize
  # X[X < 0] = 0
  # dds <- DESeqDataSetFromMatrix(countData = X, colData = t(S_merged), design = ~t(S_merged)[, 2])
  # dds <- estimateSizeFactors(dds)
  # geneCountsNormalized <- counts(dds, normalized = TRUE)

  #Inference
  simulation$n.sv[i] = num.sv(X, t(S_merged), method = "be")
  pca = prcomp(t(X))
  variance = pca$sdev^2 / sum(pca$sdev^2)
  simulation$num_PC_signif[i] = length(which(variance > 1/ncol(X)))
  simulation$var_explained_PC1[i] = variance[1]
}
```

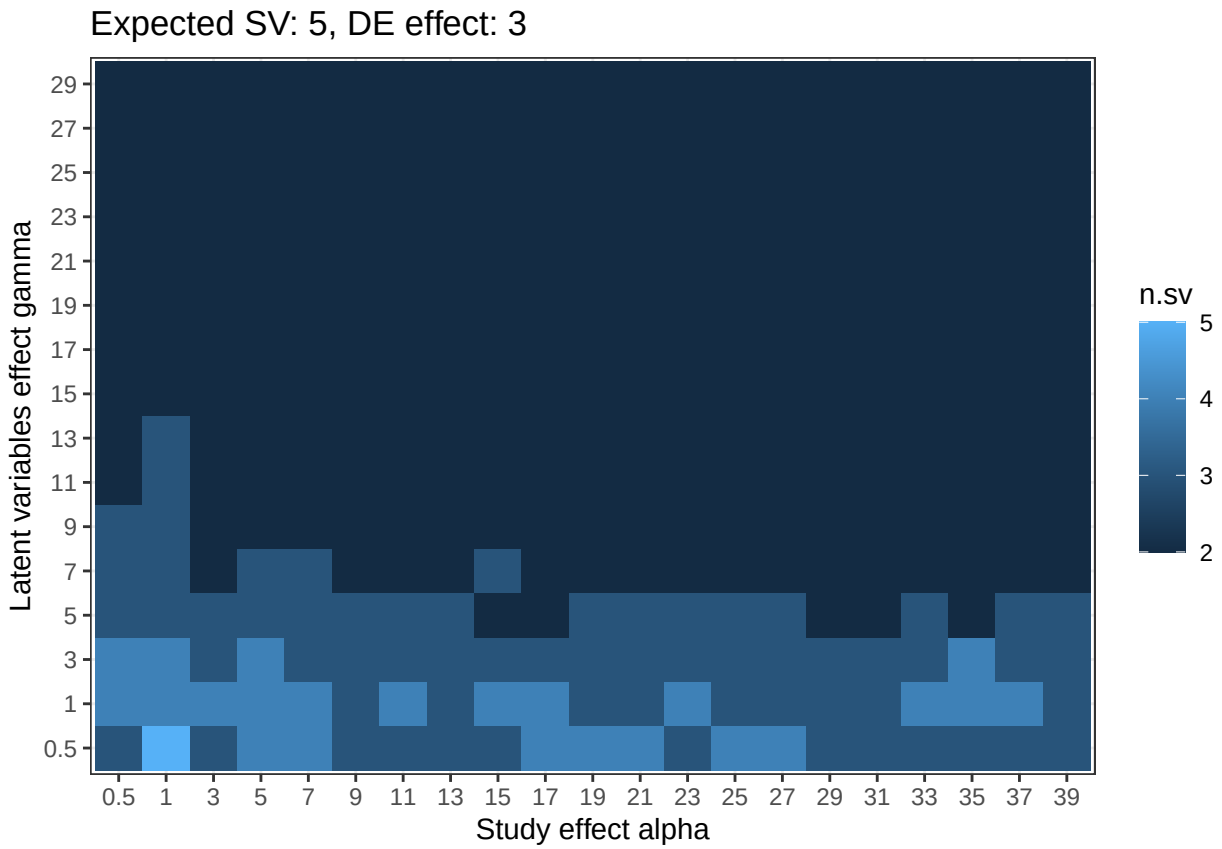
```
ggplot(simulation, aes(x = num_PC_signif, y = n.sv)) + geom_point() + ggtitle("Expected SV: 5")
```



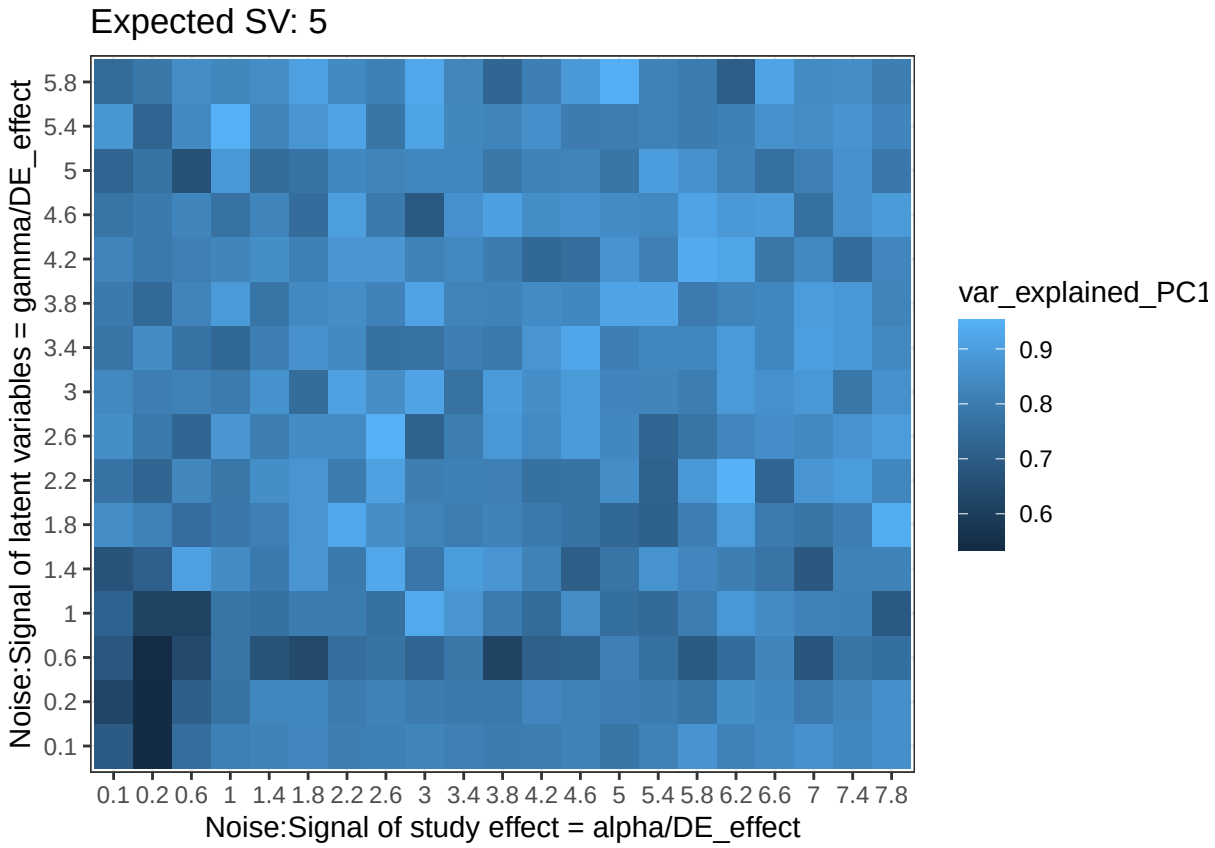

```
ggplot(simulation, aes(x = as.factor(alpha/DE_effect), y = as.factor(gamma/DE_effect), fill =  n.sv)) +
  labs(x = "Noise:Signal of study effect = alpha/DE_effect",
       y = "Noise:Signal of latent variables = gamma/DE_effect") +
   ggtitle("Expected SV: 5")
```




```
ggplot(simulation, aes(x = as.factor(alpha), y = as.factor(gamma), fill = n.sv)) + geom_tile(width = 1)
  labs(x = "Study effect alpha",
       y = "Latent variables effect gamma") +
   ggtitle("Expected SV: 5, DE effect: 3")
```




```
ggplot(simulation, aes(x =  as.factor(alpha/DE_effect), y = as.factor(gamma/DE_effect), fill = var_expla
  labs(x = "Noise:Signal of study effect = alpha/DE_effect",
       y = "Noise:Signal of latent variables = gamma/DE_effect") +
   ggtitle("Expected SV: 5")
```


Expected SV: 5
Noise:Signal of latent variables = gamma/DE_effect
5.8
5.4
5
4.6
4.2
3.8
3.4
3
2.6
2.2
1.8
1.4
1
0.6
0.2
0.1
0.1 0.2 0.6 1 1.4 1.8 2.2 2.6 3 3.4 3.8 4.2 4.6 5 5.4 5.8 6.2 6.6 7 7.4 7.8
Noise:Signal of study effect = alpha/DE_effect
var_explained_PC1
0.9
0.8
0.7
0.6