Starting Values for SUCCOTASH

David Gerard

2016-06-29

Abstract

I run SUCCOTASH from different intial values of Z and λ. Likelihood ratios are always below $1 + \epsilon$ when starting at the null MLE. There are 19 cases where the optimization gets stuck at a local maximum when starting at random locations where the LR > 1. Estimates of λ and π_0 are very similar between the two starting position strategies.

Simulation Setup

I ran through 100 repetitions of generating data from GTEX muscle data under the following parameter conditions:

- $n \in \{10, 20, 40\}$,
- $p = 1000$.
- $\pi_0 \in \{0.5, 0.9, 1\}$,
- The alternative distribution being just a standard normal. New alternatives are generated every iteration.

I extracted the most expressed p genes from the GTEX muscle data and n samples are chosen at random. Half of these samples are randomly given the "treatment" label 1, the other half given the "control" label 0. Of the p genes, $\pi_0 p$ were chosen to be non-null. Signal was added by a Poisson-thinning approach, where the log-2 fold change was sampled from a standard normal. That is

$$A_1, \ldots, A_{p/2} \sim N(0, 1) \tag{1}$$

$$B_i = 2^{A_i} \text{ for } i = 1, \ldots, p/2, \tag{2}$$

If $A_i > 0$ then we replace $Y_{[1:(n/2),i]}$ with $Binom(Y_{[j,i]}, 1/B_i)$ for $j = 1, \ldots, n/2$. If $A_i < 0$ then we replace $Y_{[(n/2+1):n,i]}$ with $Binom(Y_{[j,i]}, B_i)$ for $j = n/2 + 1, \ldots, n$.

I now describe the justification for this. Suppose that

$$Y_{ij} \sim Poisson(\lambda_j). \tag{3}$$

Let x_i be the indicator of treatment vs control for individual i. Let Ω be the set of non-null genes. Let Z be the new dataset derived via the steps above. That is

$$Z_{ij}|Y_{ij} = \begin{cases} Binom(Y_{ij}, 2^{A_j x_i}) & \text{if } A_j < 0 \text{ and } j \in \Omega \\ Binom(Y_{ij}, 2^{-A_j(1-x_i)}) & \text{if } A_j > 0 \text{ and } j \in \Omega \\ Y_{ij} & \text{if } j \notin \Omega. \end{cases} \tag{4}$$

Then

$$Z_{ij}|A_j, A_j < 0, j \in \Omega \sim Poisson(2^{A_j x_i}\lambda_j) \tag{5}$$
$$Z_{ij}|A_j, A_j > 0, j \in \Omega \sim Poisson(2^{-A_j(1-x_i)}\lambda_j), \tag{6}$$

and

$$E[\log_2(Z_{ij}) - \log_2(Z_{kj})|A_j, A_j < 0, j \in \Omega] \approx A_j x_i - A_j x_k, \text{ and} \tag{7}$$
$$E[\log_2(Z_{ij}) - \log_2(Z_{kj})|A_j, A_j > 0, j \in \Omega] \approx -A_j(1-x_i) + A_j(1-x_k). \tag{8}$$

if individual i is in the treatment group and individual k is in the control group, then this just equals A_j. I treat the A_j's as the true coefficient values when calculating the MSE.

Methods

I ran two versions of SUCCOTASH that differ only in their initial values for Z (the confounders) and λ (the variance inflation parameter). The two versions are:

- Z and λ start at their MLE's assuming that the unimodal distribution is a pointmass at 0.
- Each element of Z is a standard normal draw and λ is a χ^2_1 draw.

Both versions use a normal likelihood and a mixture of normals.

I kept their estimates of π_0, their AUC's, their MSE's, their estimates of the variance inflation parameter, their maximized log-likelihood, and their maximixed null log-likelihood (maximized over Z and λ but not over the space of unimodal densities).

Results

Read in simulation results.

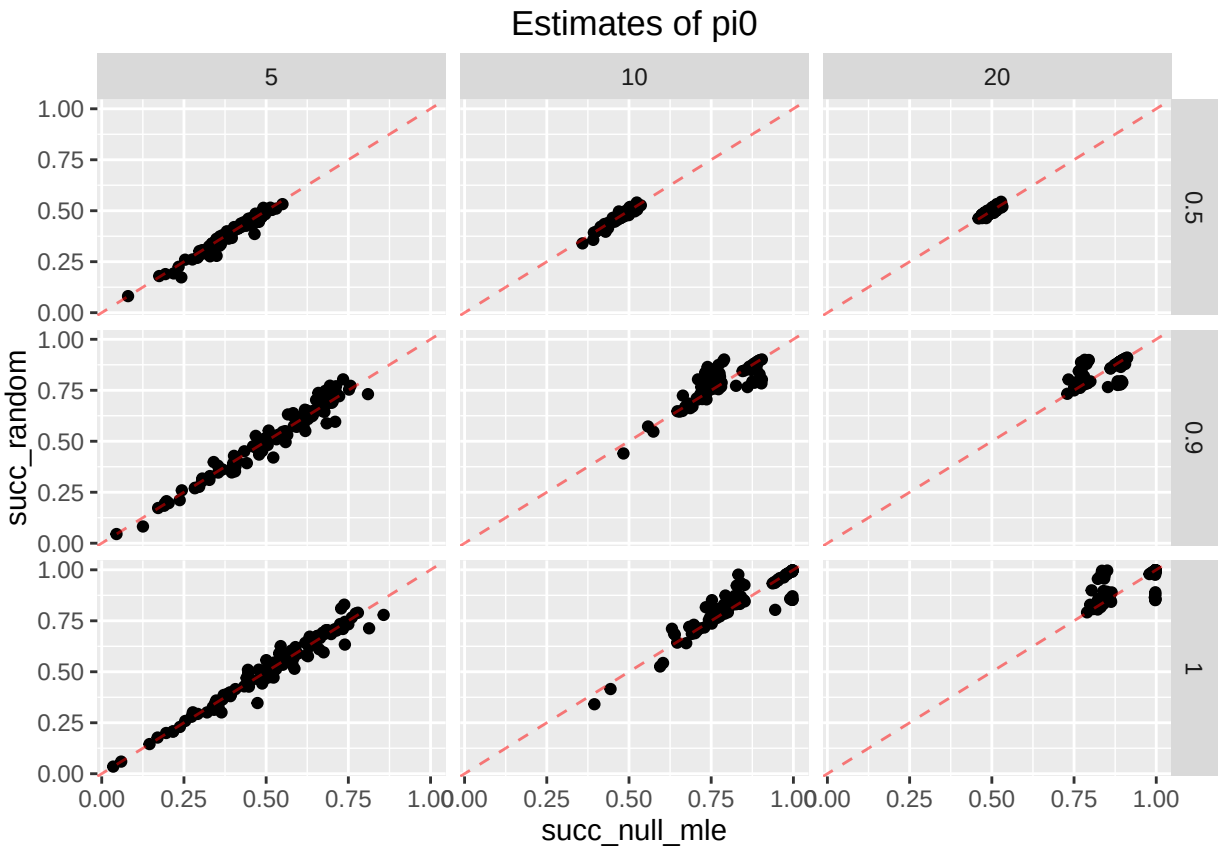
```
library(ggplot2)
library(reshape2)
library(dplyr)

pi0_mat <- read.csv(file = "pi0_ruvash_alpha1.csv")
mse_mat <- read.csv(file = "mse_ruvash_alpha1.csv")
auc_mat <- read.csv(file = "auc_ruvash_alpha1.csv")
scale_val_mat <- read.csv(file = "scale_val_ruvash_alpha1.csv")
llike_mat <- read.csv(file = "llike_ruvash_alpha1.csv")
null_llike_mat <- read.csv(file = "null_llike_ruvash_alpha1.csv")
```

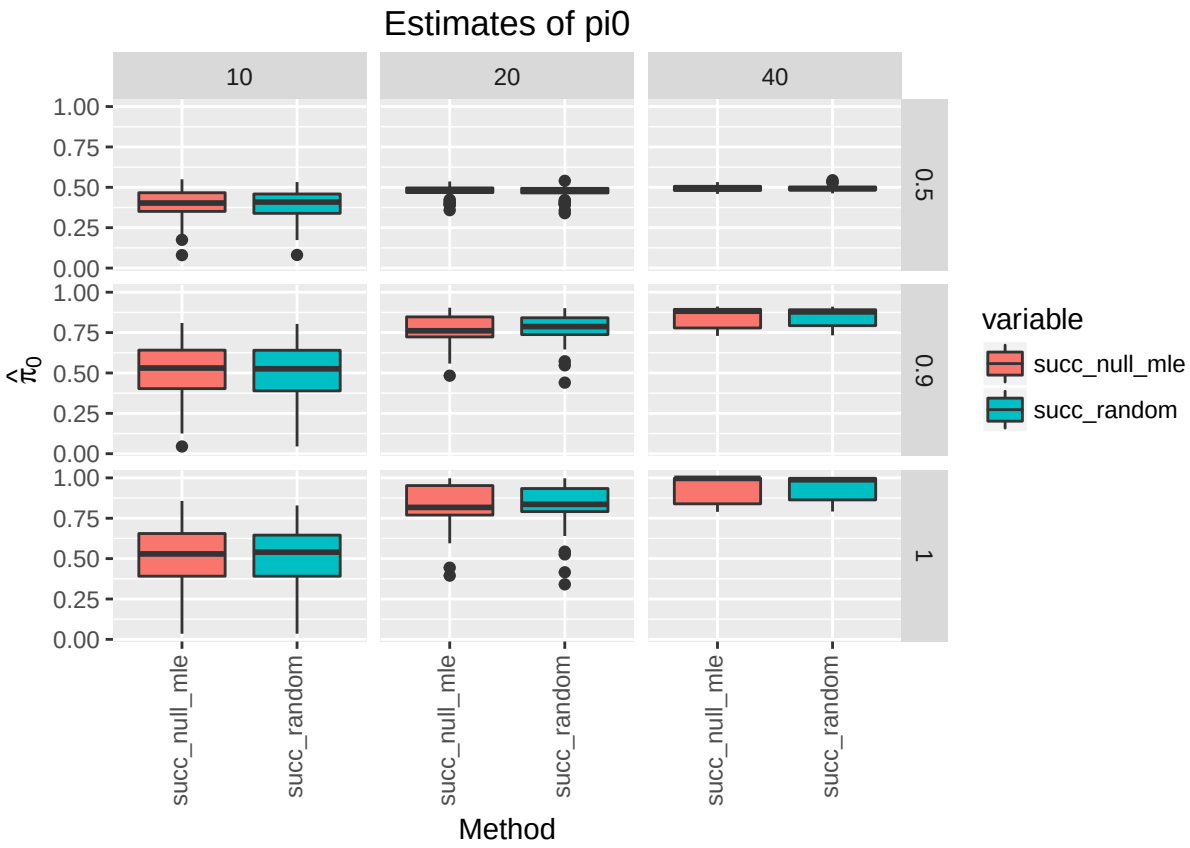
Estimates of π_0 and λ are about the same

Estimates of π_0 are pretty good and are usually about the same for both methods.

```
ggplot(data = pi0_mat, mapping = aes(x = succ_null_mle, y = succ_random)) +
    geom_point() +
    facet_grid(nullpi ~ Nsamp) +
    geom_abline(intercept = 0, slope = 1, col = 2, lty = 2, alpha = 1/2) +
    ggtitle("Estimates of pi0")
```

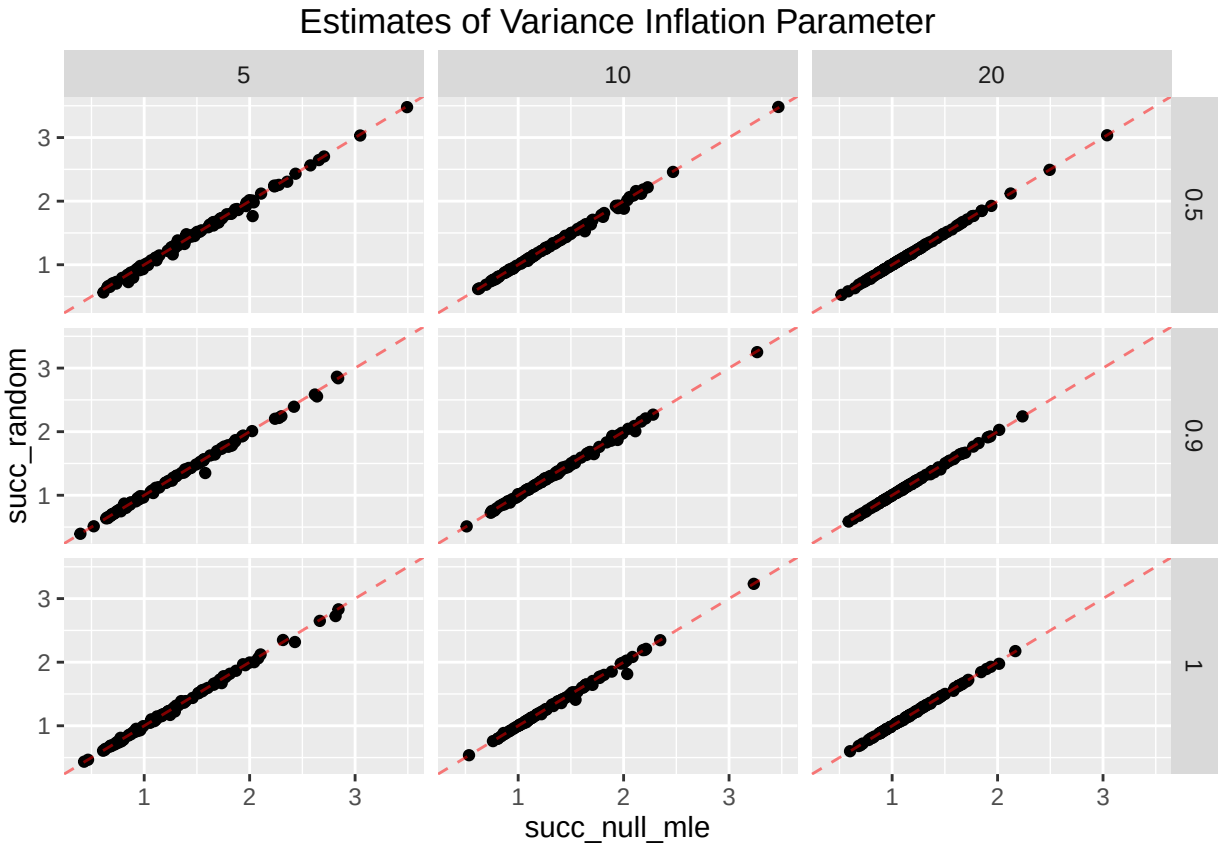



```
longdat <- melt(data = pi0_mat, measure.vars = c("succ_null_mle", "succ_random"),
                id.vars = c("Nsamp", "nullpi"))
longdat$Nsamp <- longdat$Nsamp * 2
ggplot(data = longdat, mapping = aes(x = variable, y = value, fill = variable)) +
    geom_boxplot() +
    facet_grid(nullpi ~ Nsamp) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.3)) +
    xlab("Method") + ylab(expression(hat(pi)[0])) +
    ggtitle("Estimates of pi0")
```

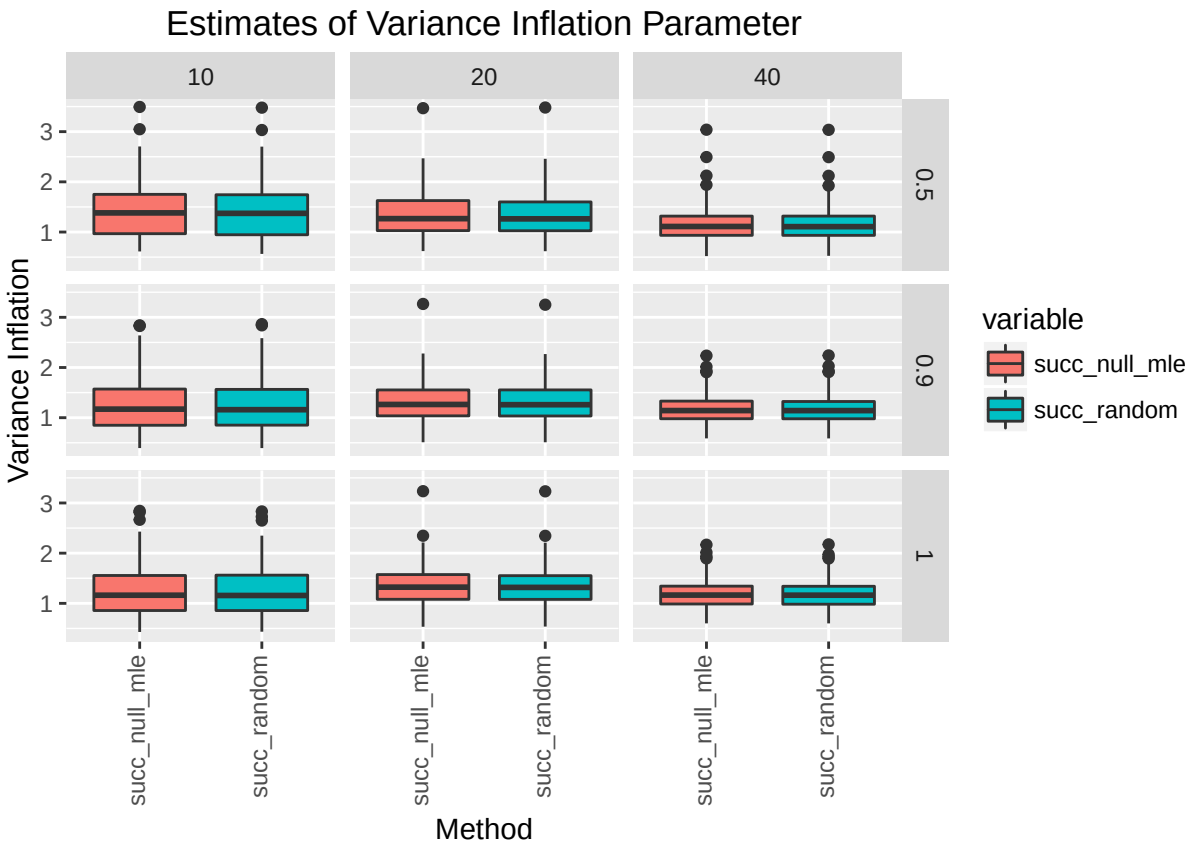


Estimates of the scaling parameter are almost always exactly the same

```
ggplot(data = scale_val_mat, mapping = aes(x = succ_null_mle, y = succ_random)) +
    geom_point() +
    facet_grid(nullpi ~ Nsamp) +
    geom_abline(intercept = 0, slope = 1, col = 2, lty = 2, alpha = 1/2) +
    ggtitle("Estimates of Variance Inflation Parameter")
```




```
longdat <- melt(data = scale_val_mat, measure.vars = c("succ_null_mle", "succ_random"),
                id.vars = c("Nsamp", "nullpi"))
longdat$Nsamp <- longdat$Nsamp * 2
ggplot(data = longdat, mapping = aes(x = variable, y = value, fill = variable)) +
    geom_boxplot() +
    facet_grid(nullpi ~ Nsamp) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.3)) +
    xlab("Method") + ylab("Variance Inflation") +
    ggtitle("Estimates of Variance Inflation Parameter")
```

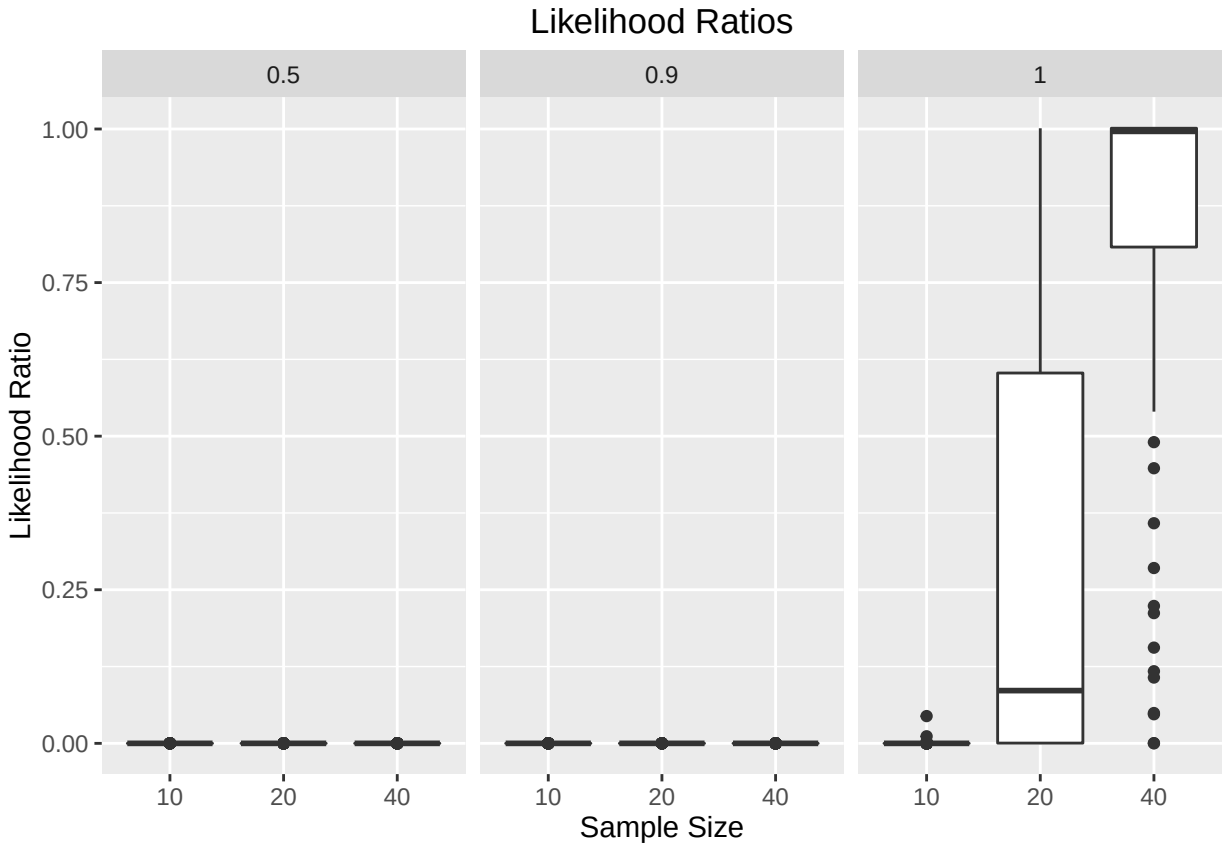


Likelihood Ratios

Now I'll just look at starting at the null-MLEs and compare the maximized log-likelihood vs the null maximized log-likelihood. All of the likelihood ratios are less than 1.

```
llike_nml <- select(llike_mat, c(Nsamp, nullpi, succ_null_mle))
colnames(llike_nml)[3] <- "max_loglike"
llike_nml$null_loglike <- null_llike_mat$succ_null_mle
llike_nml$logLR <- llike_nml$null_loglike - llike_nml$max_loglike
llike_nml$pi0hat <- pi0_mat$succ_null_mle
pi0_max <- pi0_mat$succ_null_mle[which.max(llike_nml$logLR)]

ggplot(data = llike_nml, mapping = aes(x = as.factor(2 * Nsamp), y = exp(logLR))) +
    geom_boxplot() +
    facet_grid(.~nullpi) +
    ylab("Likelihood Ratio") +
    xlab("Sample Size") +
    ggtitle("Likelihood Ratios")
```

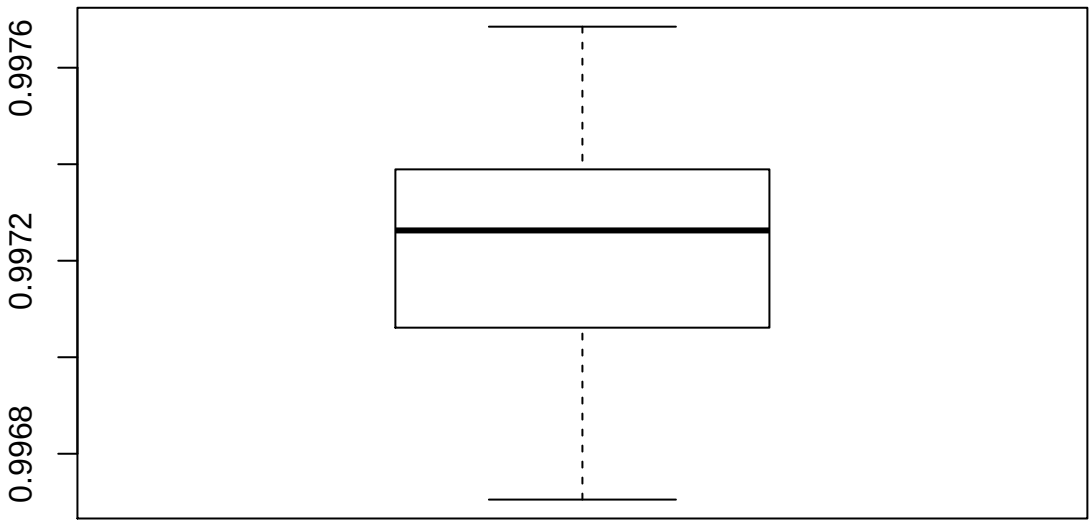


Well, the max log-likelihood ratio is actually 0.0018, but the estimate of π_0 in this scenario is 0.9972. In general, the estimates of π_0 for all scenarios that have a log-likelihood ratio greater than 0 is

```
pi0_mat$succ_null_mle[llike_nml$logLR > 0]
```

```
##  [1] 0.9973 0.9973 0.9973 0.9972 0.9970 0.9971 0.9969 0.9968 0.9972 0.9973
## [11] 0.9967 0.9974 0.9971 0.9976 0.9973 0.9972 0.9975 0.9971 0.9971 0.9970
## [21] 0.9972 0.9974 0.9971 0.9973 0.9974 0.9974 0.9973 0.9970 0.9975 0.9976
## [31] 0.9974 0.9971 0.9973 0.9974 0.9974 0.9974 0.9970 0.9973 0.9975 0.9968
## [41] 0.9977 0.9975 0.9974 0.9974 0.9973 0.9975 0.9972 0.9970 0.9969 0.9974
## [51] 0.9968 0.9973 0.9969 0.9973 0.9967 0.9974 0.9970
```

```
boxplot(pi0_mat$succ_null_mle[llike_nml$logLR > 0])
```



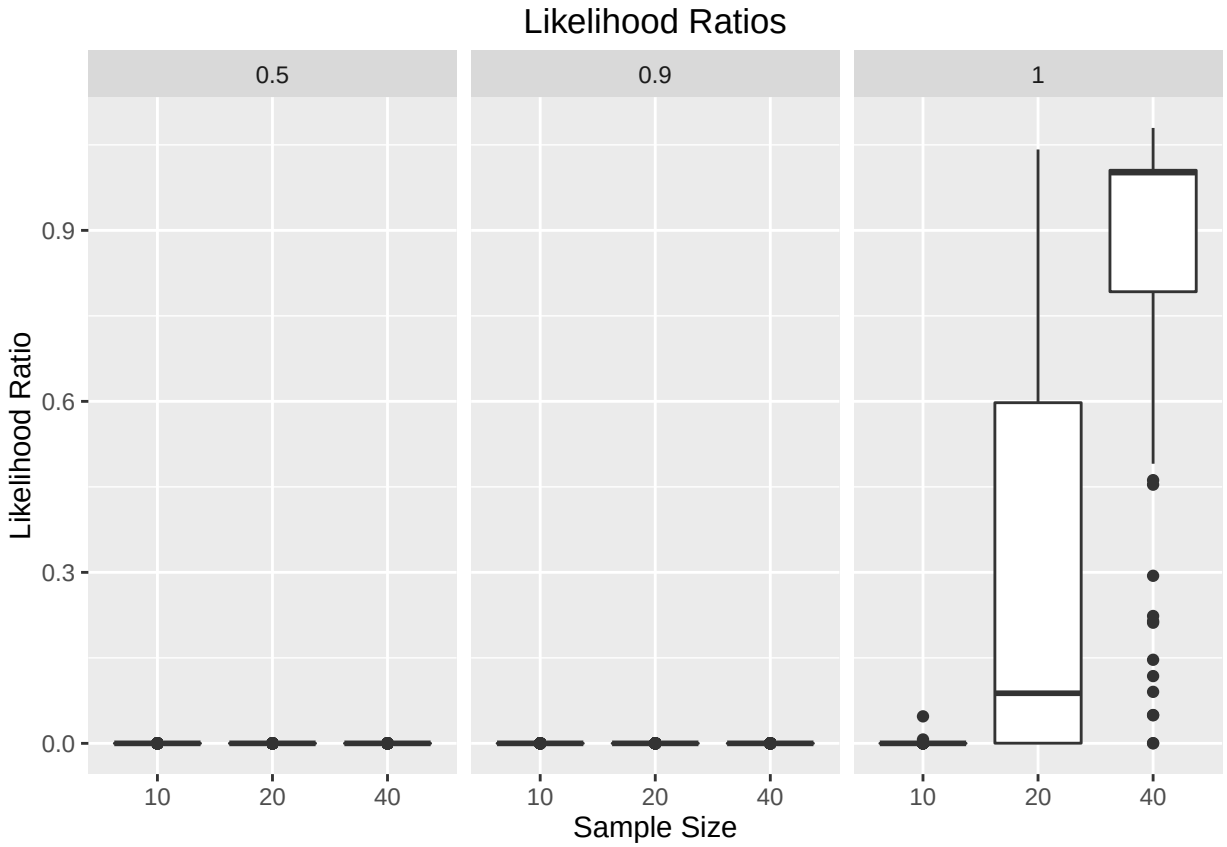
So I think the optimization algorithm leaves the all-null setting then reaches some tolerance threshold before returning to the all-null setting.

Same as above but using random starting locations

Now I'll just look at starting at random locations and compare the maximized log-likelihood vs the null maximized log-likelihood. Most of the likelihood ratios are less than one.

```
llike_ran <- select(llike_mat, c(Nsamp, nullpi, succ_random))
colnames(llike_ran)[3] <- "max_loglike"
llike_ran$null_loglike <- null_llike_mat$succ_random
llike_ran$logLR <- llike_ran$null_loglike - llike_ran$max_loglike
llike_ran$pi0hat <- pi0_mat$succ_random
pi0_max <- pi0_mat$succ_random[which.max(llike_ran$logLR)]

ggplot(data = llike_ran, mapping = aes(x = as.factor(2 * Nsamp), y = exp(logLR))) +
    geom_boxplot() +
    facet_grid(.~nullpi) +
    ylab("Likelihood Ratio") +
    xlab("Sample Size") +
    ggtitle("Likelihood Ratios")
```

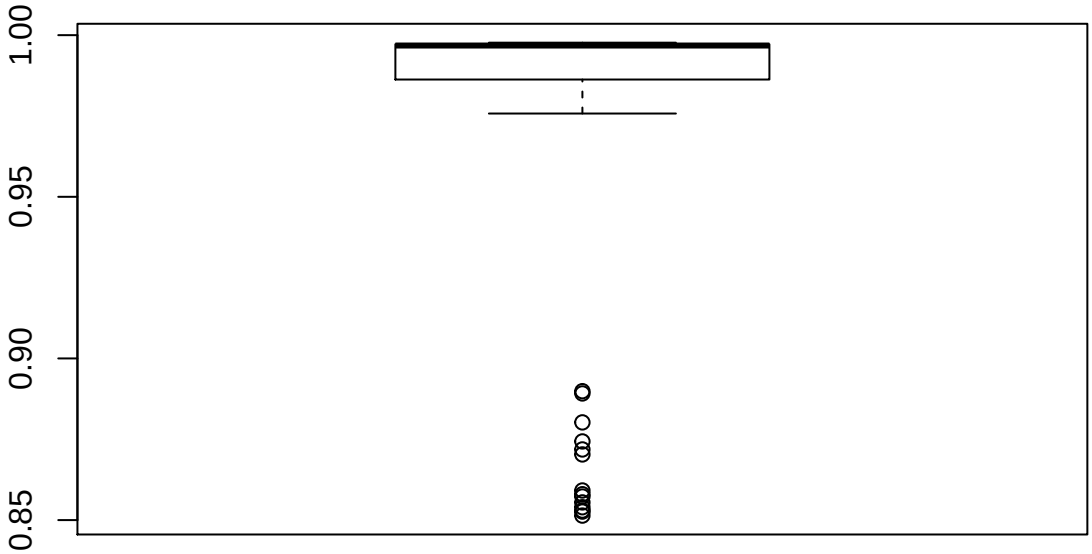


The max log-likelihood ratio is actually 0.0767, but the estimate of π_0 in this scenario is 0.8899. In general, the estimates of π_0 for all scenarios that have a log-likelihood ratio greater than 0 is

```
pi0_mat$succ_random[llike_ran$logLR > 0]
```

```
##  [1] 0.9968 0.9971 0.9972 0.8703 0.9971 0.9944 0.8526 0.8555 0.8540 0.9973
## [11] 0.8573 0.9969 0.9758 0.9971 0.9975 0.8592 0.9972 0.9969 0.9975 0.9970
## [21] 0.9970 0.9906 0.9863 0.9954 0.9972 0.9972 0.9917 0.9973 0.9972 0.9974
## [31] 0.8530 0.9898 0.9974 0.9973 0.9963 0.9970 0.9902 0.9974 0.9974 0.9966
## [41] 0.9966 0.9972 0.9973 0.9961 0.9977 0.9867 0.9973 0.9967 0.8899 0.9959
## [51] 0.9974 0.9922 0.8892 0.9969 0.8514 0.9971 0.9907 0.8802 0.8743 0.9813
## [61] 0.9968 0.9968 0.8719 0.9970 0.8580
```

```
boxplot(pi0_mat$succ_random[llike_ran$logLR > 0])
```



Since I'm starting at random locations, it's clear that I'm just getting stuck at local maxima in the 19 cases (LR > 1 but $\hat{\pi}_0 < 0.99$) where $\hat{\pi}_0$ is not really close to 1.

```
sessionInfo()
```

```
## R version 3.3.0 (2016-05-03)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.4 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_0.4.3    reshape2_1.4.1 ggplot2_2.1.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.5      knitr_1.12.28    magrittr_1.5     munsell_0.4.3
##  [5] colorspace_1.2-6 R6_2.1.2         stringr_1.0.0    plyr_1.8.4
##  [9] tools_3.3.0      parallel_3.3.0   grid_3.3.0       gtable_0.2.0
## [13] DBI_0.4          htmltools_0.3.5  yaml_2.1.13      lazyeval_0.1.10
## [17] digest_0.6.9     assertthat_0.1   formatR_1.3      evaluate_0.9
## [21] rmarkdown_0.9.6  labeling_0.3     stringi_1.0-1    compiler_3.3.0
## [25] scales_0.4.0
```