**Universal inference in R**

Here I’ll attempt to code their method in R.

I say ‘attempt’ because I’m not at all sure I interpreted their paper  
correctly. So please don’t take this blog as a ‘how to’ for the method.  
Instead, treat it as an exploration of type I and type II errors and how  
we can check them.

The method is based on likelihood ratio tests, so at the very least  
we’ll get to learn about LRTs in this blog.

If you are seriously interested in using universal inference, I suggest  
you consult a statistician.

**Testing the null that a mean equals zero**

Let’s first test if a some data drawn from a normal distribution are  
consistent with a null that the mean = 0.

We’ll define the data:

mu <- 0.4

n <- 100

set.seed(42)

y <- rnorm(n, mean = mu)

Now, do a likelihood ratio test the usual way:

(test\_stat <- 2\*(sum(dnorm(y, mean(y), sd(y), log = TRUE)) - sum(dnorm(y, 0, sd(y), log = TRUE))))

## [1] 17.25054

1 - pchisq(test\_stat, 1)

## [1] 3.276042e-05

We just compared the likelihoods of the data given a mean of zero versus  
the maximum likelihood estimate of the mean (which is just the mean of  
the data). Then we take that ratio and find its quantile on the chisq,  
that’s our (very small) p-value = 3.27E-5.

If we were using linear models, we could do the same thing like this:

m1 <- lm(y~1) #intercept only, ie a mean

m0 <- lm(y~0) #no intercept, so mean = -

anova(m0, m1, test = "Chisq")

## Analysis of Variance Table

##

## Model 1: y ~ 0

## Model 2: y ~ 1

## Res.Df RSS Df Sum of Sq Pr(>Chi)

## 1 100 126.06

## 2 99 107.36 1 18.707 3.276e-05 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

**Split LRT**

Wasserman et al. propose a ‘split LRT’ where we split the data in half.  
This has some pleasing similarities to out of sample validation. As I  
understand it, this is how we’d do the test:

First split the data at random:

i <- sample(1:n, n/2, replace = FALSE)

y0 <- y[i]

y1 <- y[-i]

Now calculate the MLEs for both splits:

mu0 <- mean(y0)

sd0 <- sd(y0)

mu1 <- mean(y1)

sd1 <- sd(y1)

Then our split test statistic is (note we are taking likelihood of the y0 split  
split using the y1 split’s mean):

split\_test\_stat0 <- sum(dnorm(y0, mu1, sd1, log = TRUE)) -

sum(dnorm(y0, 0, sd1, log = TRUE))

And we can ask if its significant like this:

exp(split\_test\_stat0) >= 1/0.05

## [1] TRUE

TRUE, so reject the null, which is the same result as our Chi square  
test above.

They also propose a cross-fit test, that is just the average of the two  
split tests:

split\_test\_stat1 <- sum(dnorm(y1, mu0, sd0, log = TRUE)) -

sum(dnorm(y1, 0, sd0, log = TRUE))

split\_test\_stat <- (split\_test\_stat1 + split\_test\_stat0)/2

exp(split\_test\_stat) >= 1/0.05

## [1] TRUE

**Power**

What about the test’s power? Well it seems a shortcoming is that the  
split LRT can have lower power (higher type II rate, or chance of  
missing real differences) than some other tests. So for a simple test  
like that above we are better of doing the test the regular way.

Let’s check its power for our simple test vs a chisq. I’ll write a  
function to do this, then iterate it.

splitLRT <- function(seed, n, mu){

set.seed(seed)

y <- rnorm(n, mean = mu)

i <- sample(1:n, n/2, replace = FALSE)

y0 <- y[i]

y1 <- y[-i]

mu0 <- mean(y0)

sd0 <- sd(y0)

mu1 <- mean(y1)

sd1 <- sd(y1)

#split test stat

split\_test\_stat0 <- sum(dnorm(y0, mu1, sd1, log = TRUE)) -

sum(dnorm(y0, 0, sd1, log = TRUE))

split\_test\_stat1 <- sum(dnorm(y1, mu0, sd0, log = TRUE)) -

sum(dnorm(y1, 0, sd0, log = TRUE))

split\_test\_stat <- (split\_test\_stat1 + split\_test\_stat0)/2

#regular Chisq LRT

test\_stat <- 2\*(sum(dnorm(y, mean(y), sd(y), log = TRUE)) - sum(dnorm(y, 0, sd(y), log = TRUE)))

chisqtest <- 1 - pchisq(test\_stat, 1)

#output results as a dataframe

data.frame(splitLRT = exp(split\_test\_stat), chisq = chisqtest)

}

Now let’s use our function:

xout <- lapply(1:1000, splitLRT, n = 50, mu = 0.5)

dfout <- do.call("rbind", xout)

sum(dfout$splitLRT >= (1/0.05))/1000

## [1] 0.415

sum(dfout$chisq <= 0.05)/1000

## [1] 0.932

So the split test only rejects the null 41.5% of the time, whereas the  
chisq rejects it 93% of the time. In other words the split test comes at  
the cost of lower power, as is explained in the paper.

It would be worth trying the suggestion in the paper of using k-fold  
cross-validation to do the splits too, maybe that would improve the  
power.

For larger sample sizes, the split test method does better:

xout <- lapply(1:1000, splitLRT, n = 150, mu = 0.5)

dfout <- do.call("rbind", xout)

sum(dfout$splitLRT >= (1/0.05))/1000

## [1] 0.974

sum(dfout$chisq <= 0.05)/1000

## [1] 1