Bayes Factors (BFs) in the TADA model

Code

- Dataset
- Create a function to calculate numerical integration
- Compare between two approaches

Inside the previous version of TADA, BFs for de novo mutations (DNMs) are calculated by using a negative binomial distribution (as a Poisson distribution with a Gamma prior). This approach uses the function $dnbinom$ inside the R language. This approach can reduce computing time, but does rely on the function $dnbinom$. **[1]**

One possible question: the way `TADA` uses parameters for the $dnbinom$ in R may create an error for BFs and downstream analyses.

- To answer this question, and also to test if there are any differences in analysis results or/and possible bugs inside the package `extTADA/TADA` from that function, we use a traditional way to calculate BFs.

We calculate BFs by using a numerical integration approach as in Calculus 1.

Bayes Factors = $P(X|H_1)/P(X|H_0)$ **[2]**

in which $P(X|H_1 = \int Poisson(X|2N\mu\gamma)Gamma(\gamma|\bar{\gamma}\beta, \beta)d\gamma$

and $P(X|H_0) = Poisssson(X|2N\mu)$.

Note: Numerical integration is an approximation, and might not be the same as analytical integration.

We will test **[1]** and **[2]** to see any differences by using some steps below.

Dataset

We will use the dataset from De Rubeis et al., (2015) [1]

[1]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4402723/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4402723/)

- Read the data frame into `R`.

Hide

```
x <- read.table("test_Data_from_ASD.txt", header = TRUE, as.is = TRUE)
##https://www.nature.com/articles/nature13772#Sec9
Ntrio = 3871
```

- Print some lines of this dataset

Hide

```
head(x)
```

	Gene <chr>	mut.rate <dbl>	dn.LoF <int>
1	SCN2A	0.00007400	4
2	SYNGAP1	0.00006600	5
3	CHD8	0.00009200	3
4	ARID1B	0.00009000	4

	Gene	mut.rate	dn.LoF
	<chr>	<dbl>	<int>
5	ANK2	0.00014923	3
6	SUV420H1	0.00003500	3

6 rows

Create a function to calculate numerical integration

We write a function for the Gamma prior.

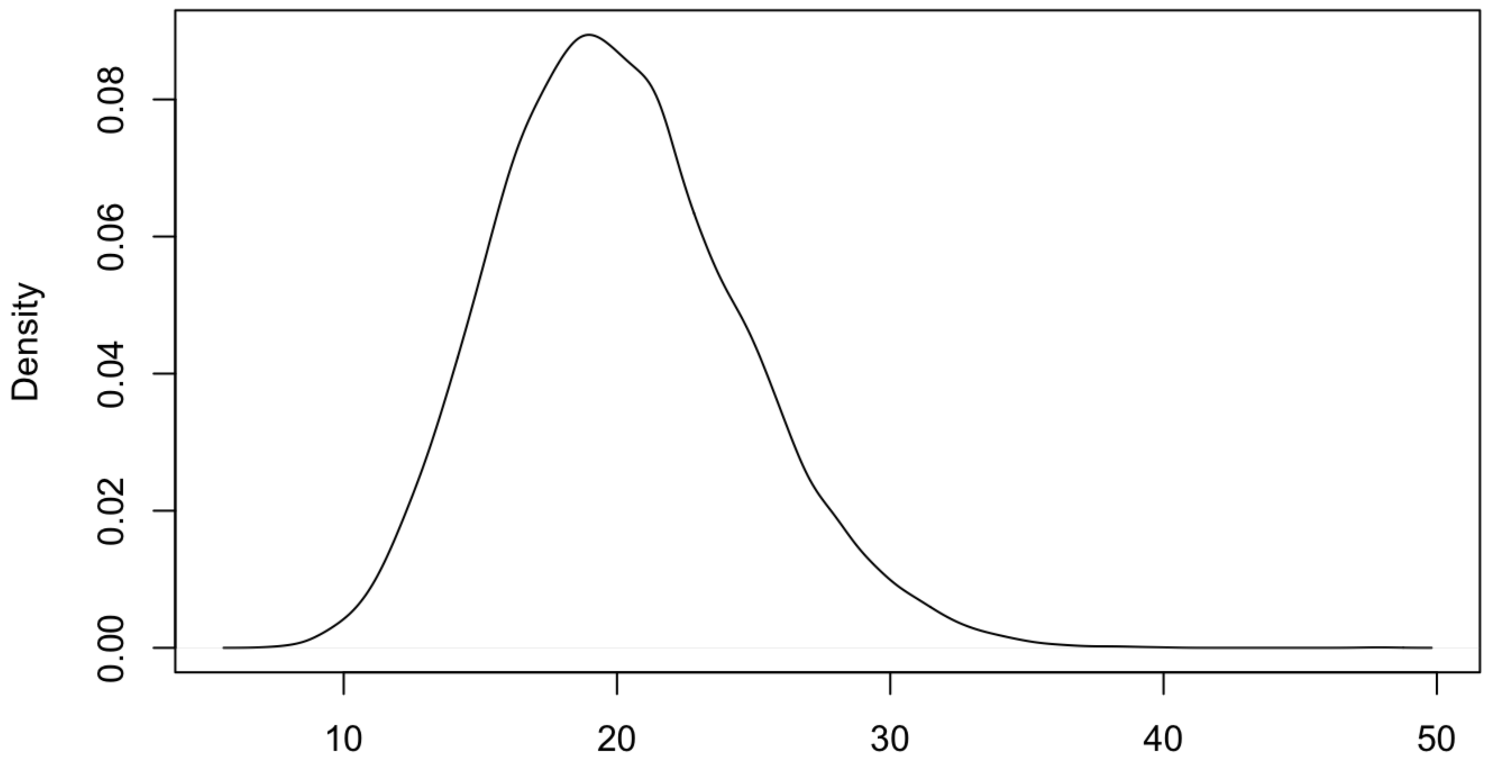
Hide

```
gamma0 = 20
beta0 = 1
f1 <- function(x.gamma){dgamma(x.gamma, shape = gamma0*beta0, rate = beta0)}
```

and have a look at its density distribution with parameters from the paper.

Hide

```
xGamma <- rgamma(10000, shape = gamma0*beta0, rate = beta0)
plot(density(xGamma), main = '', xlab = '')
```



Hide

```
gammaMin = min(xGamma)
gammaMax = max(xGamma)
```

Compare between two approaches

We will test **[1]** and **[2]** by using three examples.

Example 1

- We test for $X = 0$.

Hide

```
##numerical integration
xdn = 0
gamma0 = 20
beta0 = 1
mu0 = 10^-6
fM1 <- function(x.gamma){dpois(xdn, lambda = 2*Ntrio*mu0*x.gamma)*dgamma(x.gamma, shape = gamma0*b
eta0, rate = beta0)}

integrate(fM1, gammaMin, gammaMax, subdivisions = 200L)$value/dpois(xdn, lambda = 2*Ntrio*mu0)
```

```
[1] 0.8636122
```

Hide

```
xdn = 0
x.bf.test = bayes.factor.denovo(xdn, N = Ntrio,
                           mu = 10^-6,
                           gamma.mean = gamma0, beta = beta0)
x.bf.test
```

```
[1] 0.8637243
```

Example 2

- We test for $X = 2$.

Hide

```
##numerical integration

xdn = 2

fM1 <- function(x.gamma){dpois(xdn, lambda = 2*Ntrio*mu0*x.gamma)*dgamma(x.gamma, shape = gamma0*b
eta0, rate = beta0)}

integrate(fM1, gammaMin, gammaMax, subdivisions = 200L)$value/dpois(xdn, lambda = 2*Ntrio*mu0)
```

```
[1] 357.2032
```

Hide

```
xdn = 2
x.bf.test = bayes.factor.denovo(xdn, N = Ntrio,
                           mu = 10^-6,
                           gamma.mean = gamma0, beta = beta0)
x.bf.test
```

```
[1] 357.2117
```

Example 3

- We test for all genes from the dataset.

Hide

```
##numerical integration

x.bf.numericalIn <- apply(data.frame(x$mut.rate, x$dn.LoF), 1, function(y){

  fM1 <- function(x.gamma){dpois(y[2], lambda = 2*Ntrio*y[1]*x.gamma)*dgamma(x.gamma, shape = gamm
a0*beta0, rate = beta0)}
    integrate(fM1, gammaMin, gammaMax, subdivisions = 200L)$value/dpois(y[2], lambda = 2*Ntrio*y[1
])

})
```

Hide

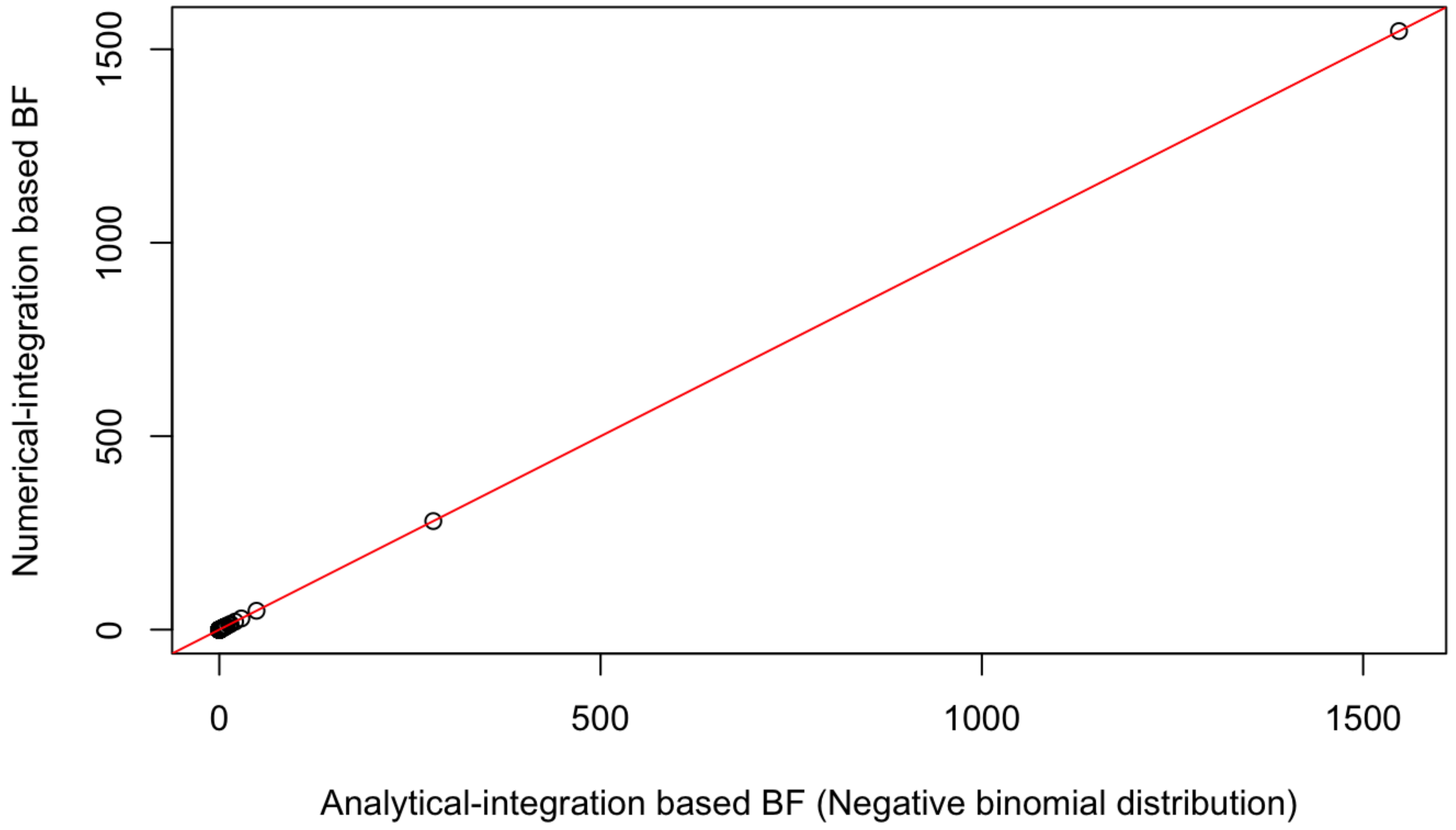
```
source("TADA.R")
gamma0 = 20
beta0 = 1
pi0 = 0.06
x.bf = bayes.factor.denovo(x$dn.LoF, N = Ntrio,
                           mu = x$mut.rate,
                           gamma.mean = gamma0, beta = beta0)
###Calculate posterior probabilities
x.pp <- pi0*x.bf/(1 - pi0 + pi0*(x.bf))
x.pp.numericalIn <- pi0*x.bf.numericalIn/(1 - pi0 + pi0*(x.bf.numericalIn))

dataTemp <- data.frame(Gene = x[, 1],
                       x.pp = x.pp,
                       x.pp.numericalIn = x.pp.numericalIn)
```

- Compare between the two methods.

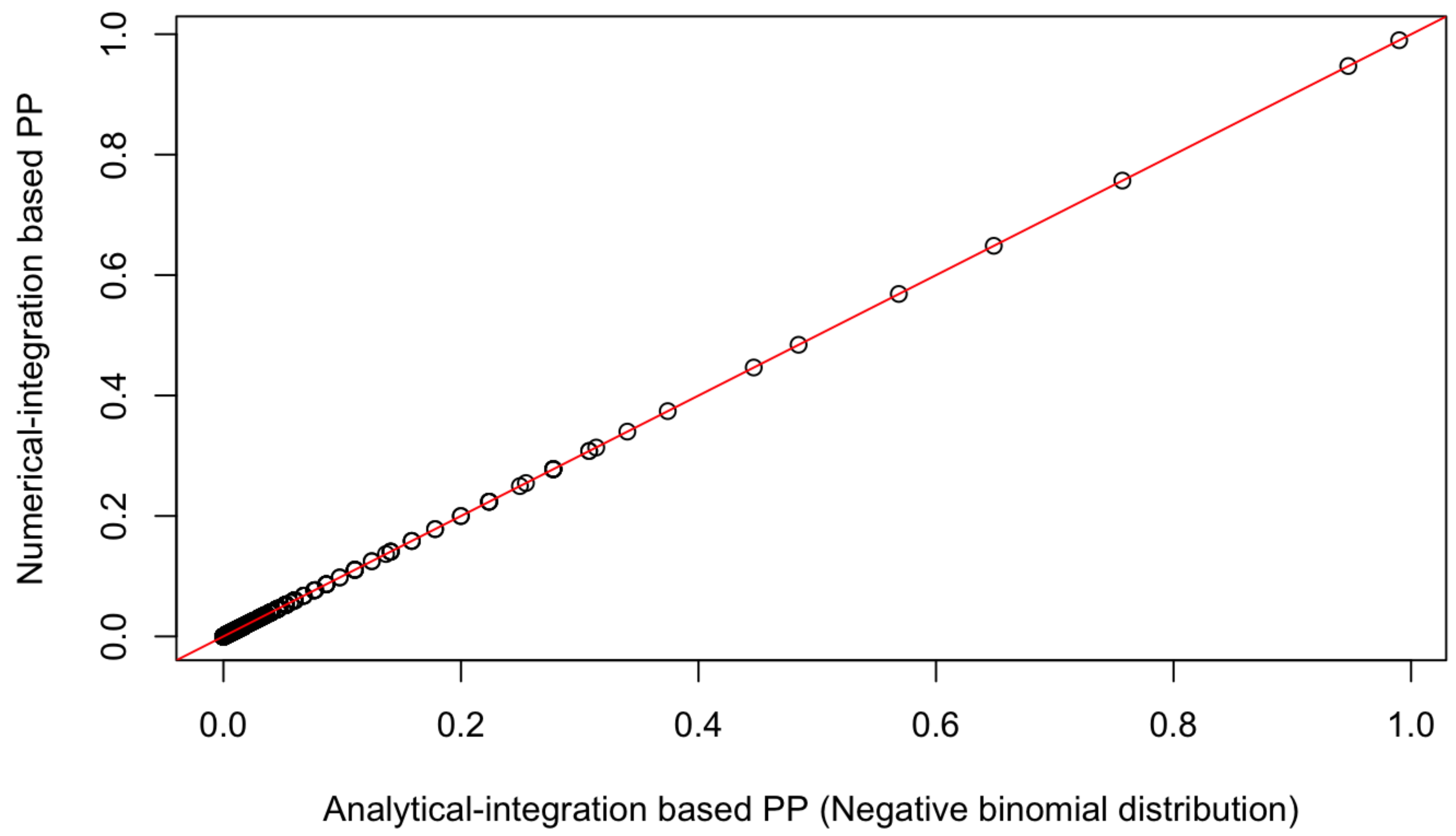
Hide

```
plot(x.bf, x.bf.numericalIn, xlab = 'Analytical-integration based BF (Negative binomial distributi
on)', ylab = 'Numerical-integration based BF')
abline(a = 0, b = 1, col = 'red')
```



Hide

```
plot(x.pp, x.pp.numericalIn, xlab = 'Analytical-integration based PP (Negative binomial distributi
on)', ylab = 'Numerical-integration based PP')
abline(a = 0, b = 1, col = 'red')
```



Hide

```
dataTemp <- data.frame(Gene = x[, 1],
                       x.pp = x.pp,
                       x.pp.numericalIn = x.pp.numericalIn)

dataTemp[dataTemp[, 'x.pp'] > 0.8, ]
```

	Gene <chr>	x.pp <dbl>	x.pp.numericalIn <dbl>
2	SYNGAP1	0.9471226	0.9471065
7	DYRK1A	0.9899745	0.9899741

2 rows

Those results are similar.