Log-norm jako default

Petr Tureček

3 9 2020

Normální rozdělení

Normální rodělení je očekávaným výsledkem aditivního procesu. Číslo z tohoto rozdělení dostaneme, pokud sečeteme n náhodně vylosovaných kladných nebo záporných čísel. Můžeme si představit, že tímto procesem aproximujeme určení hodnoty geneticky determinovaného znaku, například tělesné výšky jedince (Tímhle směrem uvažuje Fisher 1918). Každý jedinec začíná na "populačním průměru" - v tomto případě volím hodnotu 180 - a pak vylosuju 100 náhodných "alel", z nichž některé tělesnou výšku o centimetr zvýší, některé sníží.

```
# Mějme jedince A
A <- c(180, sample(c(-1, 1), 100, replace = T))
A
```

```
##   [1] 180   1   1  -1   1  -1  -1  -1  -1  -1   1   1   1   1   1  -1  -1
##  [18]   1   1   1  -1  -1  -1   1  -1  -1   1   1   1  -1   1   1  -1   1
##  [35]   1  -1  -1   1   1   1   1   1   1  -1   1   1  -1   1  -1  -1   1
##  [52]   1  -1  -1  -1   1  -1   1   1   1   1  -1   1  -1   1   1   1   1
##  [69]   1   1   1  -1  -1  -1  -1   1  -1   1   1   1   1  -1  -1   1  -1
##  [86]  -1  -1  -1   1   1  -1  -1   1   1   1  -1   1  -1  -1  -1  -1
```

```
# Jeho tělesná výška je pak sumou průměru a vylosovaných náhodných vlivů
sum(A)
```

```
## [1] 190
```

Takových jedinců vyrobíme například 200. Suma průměru a oněch náhodných vlivů u každého jedince nám dá vektor tělesných výšek v našem vzorku. Plotneme to jako základní histogram.

```
# Mějme 200 jedinců definovaných jako průměr výšky a 100 náhodných vlivů +1
# nebo -1 cm
listA <- lapply(1:200, function(i) {
    c(180, sample(c(-1, 1), 100, replace = T))
})

# Můžeme se podívat na prvních 5
str(listA[1:5])
```

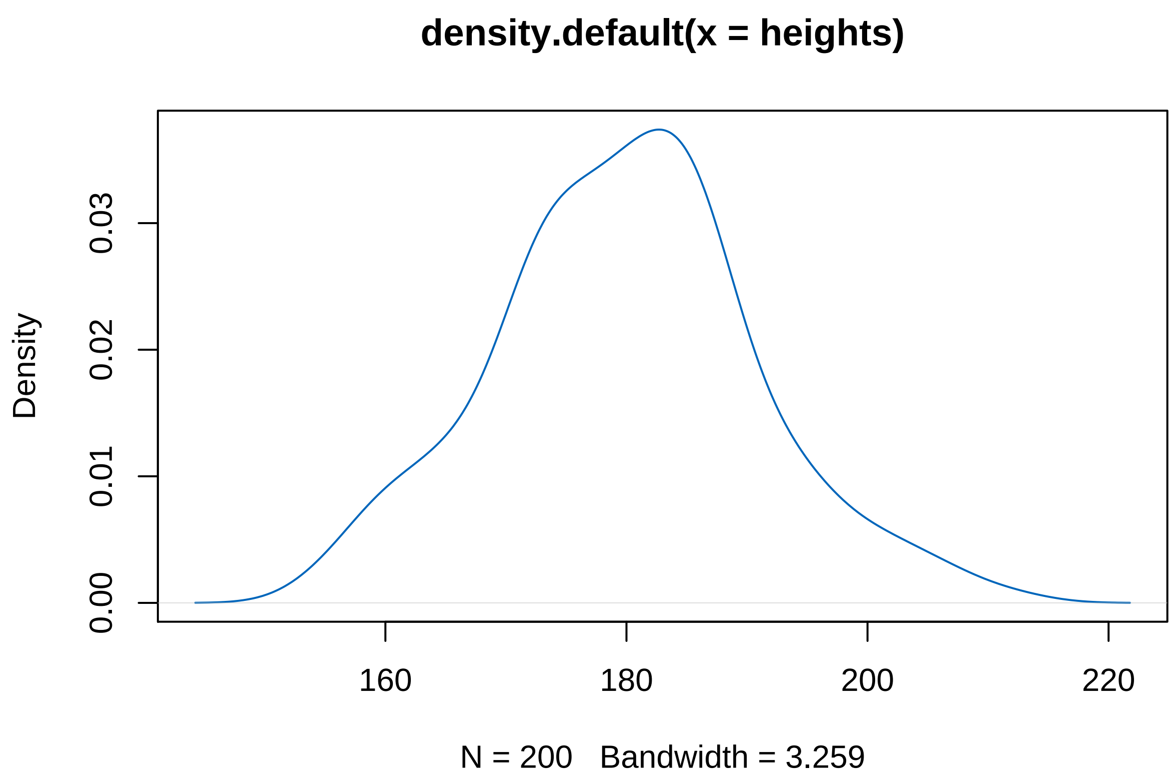
```
## List of 5
##  $ : num [1:101] 180 -1 -1 -1 -1 -1 -1 1 1 1 ...
##  $ : num [1:101] 180 1 -1 1 1 1 -1 1 1 1 ...
##  $ : num [1:101] 180 -1 1 -1 -1 -1 1 1 1 -1 ...
##  $ : num [1:101] 180 -1 1 1 1 1 -1 1 -1 1 ...
##  $ : num [1:101] 180 -1 1 -1 1 -1 1 -1 -1 1 ...
```

```
# Sumy těchto vektorů v seznamu jedinců jsou tělesnými výškami
heights <- sapply(listA, sum)
heights
```

```
##   [1] 176 184 186 174 184 190 190 176 174 172 186 174 186 190 188 182 174
##  [18] 202 188 180 180 176 180 188 172 180 196 188 186 184 190 184 160 170
```

```
##  [35] 176 178 184 160 172 172 192 172 196 166 188 156 176 180 160 184 170
##  [52] 206 172 200 178 168 174 178 182 186 170 170 182 174 162 194 174 184
##  [69] 172 180 158 164 182 186 182 198 198 194 186 190 174 186 182 170 198
##  [86] 182 164 174 188 186 180 184 204 200 176 186 174 186 184 186 188 180
## [103] 170 166 178 160 202 184 164 190 176 186 194 166 156 166 186 158 182
## [120] 184 176 186 176 182 174 180 182 176 188 174 154 190 186 160 172 180
## [137] 208 166 160 184 168 170 178 184 176 178 182 190 172 184 170 194 186
## [154] 184 178 172 184 180 202 180 166 190 186 164 176 178 196 206 178 180
## [171] 194 184 178 192 172 174 180 194 212 162 194 182 162 172 176 180 178
## [188] 170 174 166 174 172 188 180 180 192 170 184 180 166
```

```r
# Plotneme to jako základní histogram.
plot(density(heights), col = "#0066BB")
```



Tento vektor výšek by měl +- projít testem, zda se jedná o čísla z normálního rozdělení (Shpairo-Wilkův test normality, který zde používám -spíš z lenosti- ukazuje tím nižší p hodnoty, čím větší vzorek si vyrobím, je lepší koukat na to W. Pokud je větší než 0.95, je v podstatě jisté, že distribuci čísel ve vektoru je možné dobře popsat normálním rozdělením.)

```r
shapiro.test(heights)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  heights
## W = 0.9904, p-value = 0.2044
```

Log-normální rozdělení

Podobné cvičení je možné udělat pro multiplikativní proces. Nebude nás zajímat součet čísel v každém vektoru, ale jejich součin. Můžeme o tom mluvit klidně rovnou jako o velikosti areálu. Pro jednoduchost můžeme předpokládat, že medián velikosti areálu je velký asi jako naše republika (cca 80 tisíc čtverečních kilometrů). Vůbec nevím, jestli je tohle pravda, ale rychlé googlení "median species area", nedává jednoznačnou a uspokojivou odpověď. Ono je to jedno... Byl jsem v pokušení dát medián areálu prostě 1, ale tohle je pro představu asi lepší.
Nesampluju čísla +1 a -1 ale čísla 1.2 a 0.83 (což je 1/1.2). Beru, že toto jsou náhodné vlivy, které nám ze "startovní pozice" běžný areál zvětší o 20%, nebo ho o příslušný počet procent (asi 17%) zmenší. Když se tedy areál jednou zvětší a jednou zmenší, vrátí se na výchozí hodnotu.

```
# Kolik je 1/1.2?
1/1.2
```

```
## [1] 0.8333333
```

```
# Areál 80 (tis. km2) se jednou zvětší a jednou zmenší, vrátí se na původní
# hodnotu
(80 * 1.2) * 0.83333333
```

```
## [1] 80
```

```
# Je tomu tak i když se nejdříve zmenší a pak zvětší. Na pořadí změn
# velikosti nezáleží, protože násobení je komutativní.
(80 * 0.83333333) * 1.2
```

```
## [1] 80
```

Kdybych přistupoval k procentním bodům aditivně (tedy zvolil bych výběr hodnot stejně daleko od 100% - např. 80% a 120%, nevrátili bychom se na původní hodnotu, protože 1.2*0.8 < 1

```
(80 * 0.8) * 1.2
```

```
## [1] 76.8
```

```
(80 * 1.2) * 0.8
```

```
## [1] 76.8
```

V příkladu ekvivalentním náhodnému aditivnímu procesu výše, je tedy nutné používat jedno číslo větší než 1 (zde 1.2) a jeho převrácenou hodnotu.

```
# Ručně zde nastavím seed generátoru náhodných čísel, na 'náhodnosti'
# procesu to nic neubírá, jen vím, že ta sekvence vyjde pokaždé stejně,
# takže můžu níže mluvit o rozdílech mezi sekvencemi s různými seedy.
set.seed(111)
B <- c(80, sample(c(1/1.2, 1.2), 100, replace = T))
B
```

```
##   [1] 80.0000000  1.2000000  0.8333333  1.2000000  0.8333333  0.8333333
##   [7]  0.8333333  0.8333333  0.8333333  0.8333333  1.2000000  1.2000000
##  [13]  1.2000000  0.8333333  0.8333333  0.8333333  1.2000000  1.2000000
##  [19]  0.8333333  1.2000000  1.2000000  1.2000000  0.8333333  1.2000000
##  [25]  0.8333333  1.2000000  0.8333333  0.8333333  0.8333333  0.8333333
##  [31]  0.8333333  0.8333333  0.8333333  0.8333333  1.2000000  0.8333333
##  [37]  0.8333333  0.8333333  1.2000000  1.2000000  1.2000000  1.2000000
##  [43]  0.8333333  0.8333333  1.2000000  1.2000000  0.8333333  1.2000000
```

```
## [49] 1.2000000 0.8333333 0.8333333 0.8333333 1.2000000 1.2000000
## [55] 1.2000000 0.8333333 1.2000000 1.2000000 1.2000000 1.2000000
## [61] 1.2000000 0.8333333 0.8333333 0.8333333 0.8333333 0.8333333
## [67] 1.2000000 0.8333333 0.8333333 0.8333333 1.2000000 1.2000000
## [73] 1.2000000 1.2000000 0.8333333 1.2000000 1.2000000 0.8333333
## [79] 0.8333333 0.8333333 1.2000000 0.8333333 1.2000000 1.2000000
## [85] 0.8333333 0.8333333 0.8333333 0.8333333 0.8333333 0.8333333
## [91] 1.2000000 1.2000000 1.2000000 1.2000000 1.2000000 0.8333333
## [97] 0.8333333 0.8333333 1.2000000 1.2000000 1.2000000
```

```r
prod(B)
```

```
## [1] 26.79184
```

Areál je malý, přitom poměr 1.2 a její převrácené hodnoty ve vektoru je relativně vyrovnaný.

```r
summary(as.factor(B))
```

```
## 0.833333333333333               1.2                80
##                53                47                 1
```

Náhodný string založený na seedu 222 obsahuje o něco více čísel 1.2, rozdíly mezi součiny prvků ve vektorech jsou ale obrovské.

```r
set.seed(222)
B <- c(80, sample(c(1/1.2, 1.2), 100, replace = T))
summary(as.factor(B))
```

```
## 0.833333333333333               1.2                80
##                45                55                 1
```

```r
prod(B)
```

```
## [1] 495.3389
```

```r
# Mějme takových náhodných areálů zase 200; průměr jako republika a 100
# náhodných vlivů *1.2 nebo /1.2
listB <- lapply(1:200, function(i) {
    c(80, sample(c(1/1.2, 1.2), 100, replace = T))
})

# Můžeme se podívat na prvních 5 zase
str(listB[1:5])
```

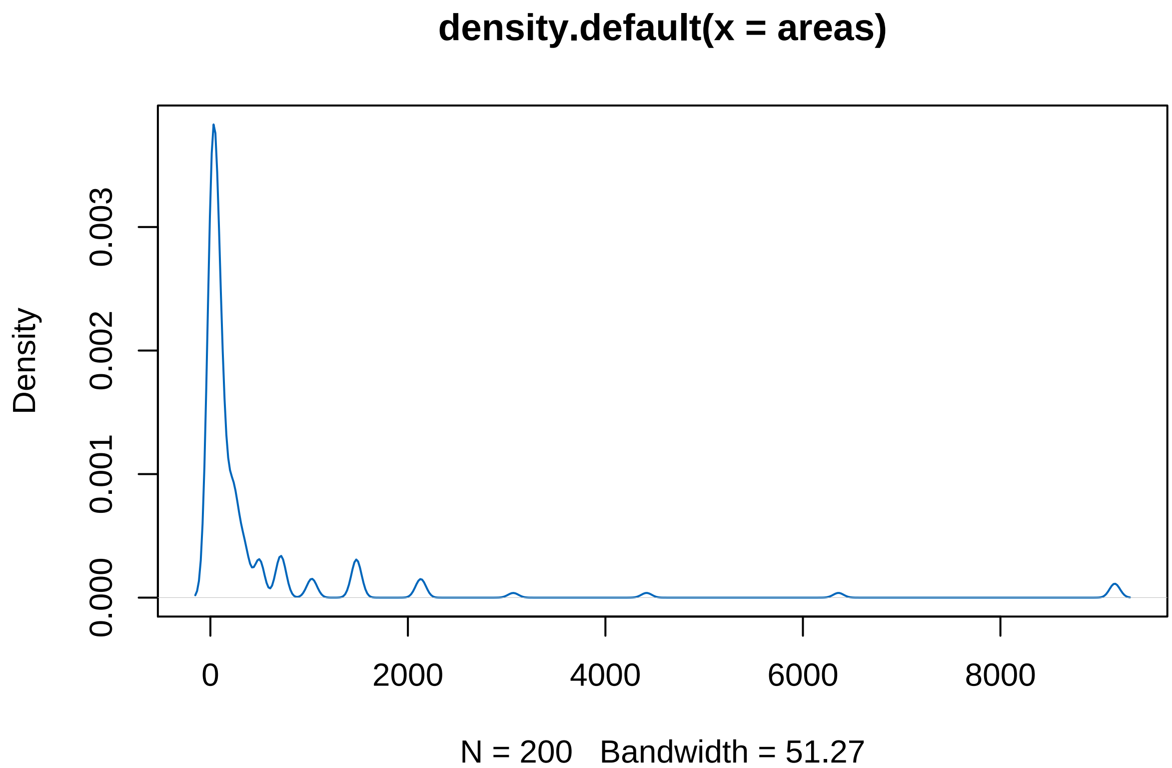
```
## List of 5
##  $ : num [1:101] 80 0.833 0.833 0.833 0.833 ...
##  $ : num [1:101] 80 1.2 1.2 1.2 1.2 ...
##  $ : num [1:101] 80 1.2 0.833 1.2 0.833 ...
##  $ : num [1:101] 80 0.833 1.2 0.833 0.833 ...
##  $ : num [1:101] 80 1.2 1.2 1.2 0.833 ...
```

```r
# Spočítáme plochy
areas <- sapply(listB, prod)
areas
```

```
##   [1]   18.6054431  238.8787200  238.8787200  165.8880000 1479.0740712
##   [6] 2129.8666625  343.9853568  115.2000000  343.9853568   55.5555556
##  [11]  713.2880359    3.0048829    6.2309253  238.8787200   26.7918381
##  [16]    8.9725324  713.2880359  713.2880359   18.6054431    2.0867243
##  [21]   26.7918381   12.9204466    6.2309253   38.5802469   55.5555556
```

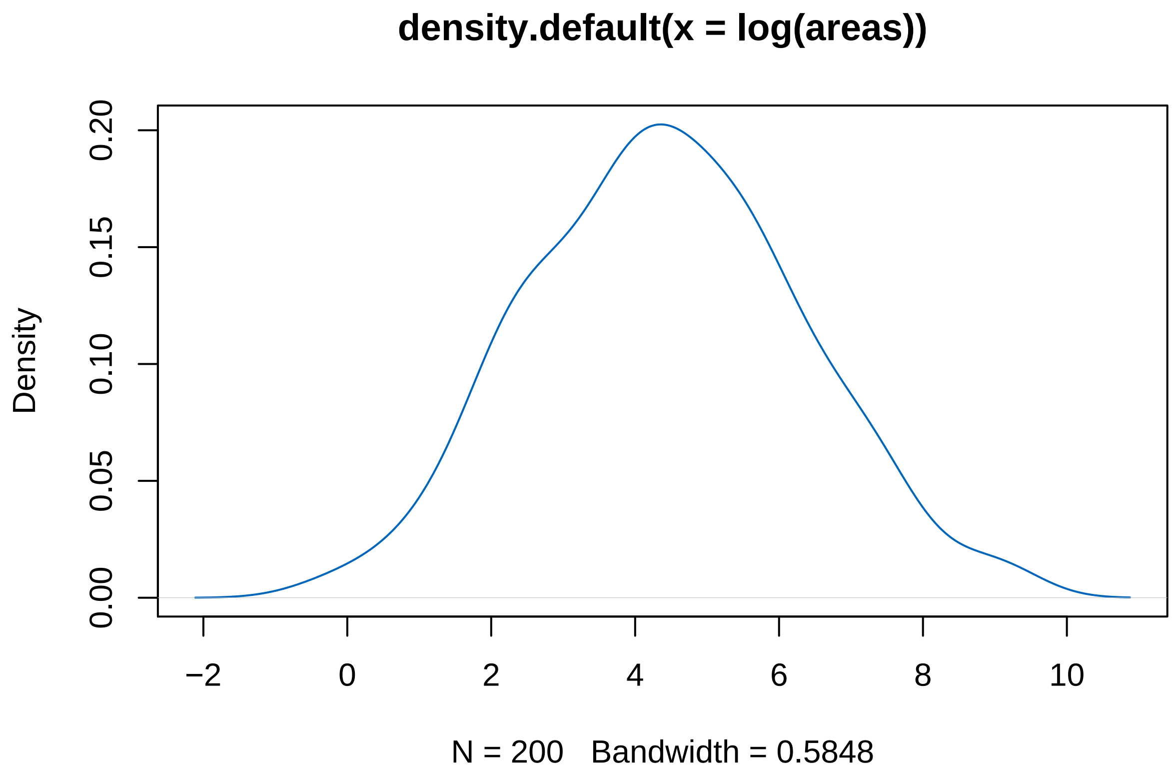
```
##  [26] 1479.0740712  115.2000000  713.2880359   18.6054431 1479.0740712
##  [31]    3.0048829   55.5555556    8.9725324    1.4491141   38.5802469
##  [36]  115.2000000   55.5555556    4.3270314  495.3389138  165.8880000
##  [41]   38.5802469   55.5555556 1479.0740712   80.0000000 2129.8666625
##  [46]  238.8787200  115.2000000   80.0000000   55.5555556    6.2309253
##  [51]  115.2000000  115.2000000   18.6054431  115.2000000  115.2000000
##  [56]   12.9204466  115.2000000  495.3389138   12.9204466  238.8787200
##  [61]  238.8787200   55.5555556   38.5802469 9158.0367978  238.8787200
##  [66]   55.5555556    8.9725324  165.8880000  238.8787200   12.9204466
##  [71]   55.5555556  713.2880359   80.0000000   18.6054431 9158.0367978
##  [76]  115.2000000  238.8787200   18.6054431  238.8787200   18.6054431
##  [81]    4.3270314  115.2000000    3.0048829   55.5555556  238.8787200
##  [86]  495.3389138  165.8880000  713.2880359   12.9204466  238.8787200
##  [91] 1027.1347716   55.5555556   12.9204466   26.7918381   38.5802469
##  [96]   80.0000000   80.0000000    0.6988397  238.8787200   38.5802469
## [101]  115.2000000  713.2880359    8.9725324  343.9853568  115.2000000
## [106]  713.2880359  495.3389138 1479.0740712    8.9725324   18.6054431
## [111] 6359.7477763   12.9204466   55.5555556    6.2309253    4.3270314
## [116]   26.7918381   55.5555556  495.3389138   38.5802469 3067.0079940
## [121]   80.0000000  238.8787200   26.7918381  343.9853568  495.3389138
## [126]   80.0000000  115.2000000 1479.0740712  115.2000000   38.5802469
## [131]   38.5802469   55.5555556  115.2000000   12.9204466   55.5555556
## [136]   80.0000000   38.5802469  115.2000000   80.0000000  238.8787200
## [141]  343.9853568   55.5555556    6.2309253   12.9204466  343.9853568
## [146]   26.7918381    2.0867243  115.2000000   55.5555556   12.9204466
## [151]   55.5555556   26.7918381 1479.0740712    6.2309253  115.2000000
## [156]   26.7918381   38.5802469  495.3389138 4416.4915113  165.8880000
## [161]    8.9725324   55.5555556  343.9853568  343.9853568  165.8880000
## [166]  713.2880359  238.8787200    8.9725324 1027.1347716  115.2000000
## [171]  115.2000000   55.5555556    8.9725324  115.2000000 1027.1347716
## [176]  238.8787200  495.3389138   26.7918381    8.9725324   18.6054431
## [181]   12.9204466  343.9853568  343.9853568    8.9725324    1.0063292
## [186]   55.5555556   26.7918381   55.5555556    4.3270314  238.8787200
## [191]    8.9725324   18.6054431  238.8787200 9158.0367978 2129.8666625
## [196] 1479.0740712 1027.1347716   26.7918381 2129.8666625    8.9725324
```

```r
# A plotneme
plot(density(areas), col = "#0066BB")
```



Když se necháme unést tím, že je tam pár těch obrovských areálů, kde se zrovna sešlo dost případů náhodného zvětšení oproti zmenšení, můžeme dospět k závěru, že "má většina druhů malé areály". (Jsou ale vážně malé? Absolutně malé, enbo malé jen v porovnání s kosmopolitními druhy?) Na tom rozdělení ale není nic divného, je jen důsledkem toho, že vývoj velikosti areálu druhu je multiplikatvní proces. Klást si tuto otázku je podobné jako ptát se (a i ta odpověď je pak podobná), proč má většina geneticky determinovaných znaků přibližně normální rozdělení (no ono asi nakonec nemá, protože Fischerovská jednoduchá aditivní variance vedoucí na normální rozložení je abstraktním ideálem. Budou se uplatňovat různé epistatické interakce, určitě najdeme geny s účinky multiplikativní povahy - proto ty časté tlusté konce rozložení biologických znaků směrem k vysokým hodnotám a tak...) Takhle prostě log-normální rozdělení vypadá. Ono když se podíváme na distribuci logaritmů tohoto vektoru, dostaneme rozdělení normální, které projde i tím přísným Shapiro testem.

```
plot(density(log(areas)), col = "#0066BB")
```




```r
shapiro.test(log(areas))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(areas)
## W = 0.99208, p-value = 0.3511
```

Není divu. Takhle je logaritmus vynalezen - převádí komplikované násobení na jednoduché sčítání. Celý ten multiplikativní proces se na logaritmické škále změní na proces aditivní, který je naprosto identický s tím příkladem vzniku tělesných výšek (akorát jsou tam jiná čísla, místo +1 a -1 konkrétně -0.182 a +0.182).

```r
listBlog <- lapply(listB, log)
str(listBlog[1:5])
```

```
## List of 5
##  $ : num [1:101] 4.382 -0.182 -0.182 -0.182 -0.182 ...
##  $ : num [1:101] 4.382 0.182 0.182 0.182 0.182 ...
##  $ : num [1:101] 4.382 0.182 -0.182 0.182 -0.182 ...
##  $ : num [1:101] 4.382 -0.182 0.182 -0.182 -0.182 ...
##  $ : num [1:101] 4.382 0.182 0.182 0.182 -0.182 ...
```

Logaritmus součinu je součet logaritmů jednotlivých činitelů, což lze velmi snadno oddemonstrovat.

```r
head(data.frame(sapply(listBlog, sum), log(areas)))
```

```
##   sapply.listBlog..sum. log.areas.
```

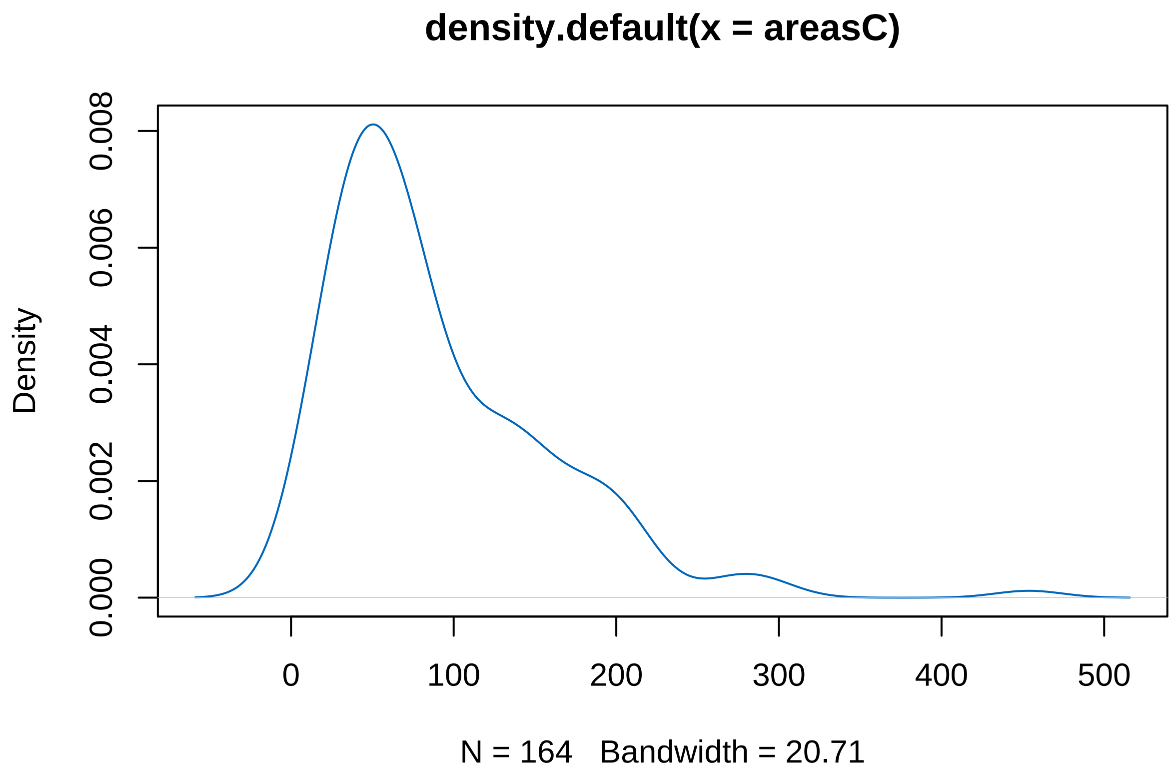
```
## 1              2.923454   2.923454
## 2              5.475956   5.475956
## 3              5.475956   5.475956
## 4              5.111313   5.111313
## 5              7.299172   7.299172
## 6              7.663815   7.663815
```

Dá se představit spousta různých procesů, které na tohle rozdělení povedou (třeba takové, co byly/měly být představeny v té BP), ale je klíčové, že my si nemusíme vybírat žádný z nich. Jednou se areál zvětší, protože se druh přizpůsobí i na jinou niku/stane se větším generalistou, jindy se zvětší, protože se obývaný biotop kvůli změně podnebí rozšíří na větší území, někdy se zmenší, protože shoří půlka lesa atd. Jde jen o to, že všechny tyto změny budou s větší pravděpodobností vypadat jako "zvětšení o procento" než "zvětšení o jeden hektar". Tohle jsem měl na mysli, když jsem říkal, že log-normal by měl být výchozí neinformovaný předpoklad a odchylka od tohoto rozdělení teprve tím, co je potřeba odvysvětlit. Třeba velmi malé areály hostí velmi malé populace, takže když do toho bude sahat ještě jiný stochastický proces aditivního rázu (fluktuace počtu organismů kolem hodnoty predikované velikostí areálu např.), budou druhy s velmi malými populacemi - potažmo velmi malými areály - častěji náhodou mizet (jak se jejich populace dostane na 0, nebude druh a nebude areál). Nebo naopak - ty největší areály, které bychom na záladě log-normálního rozložení predikovali, se nevejdou na kontinenty (nebo na planetu!), což povede k mírné nadreprezentaci těch areálů "těsně pod vrcholem" (Storch a spol 2012 doi:10.1038/nature11226).

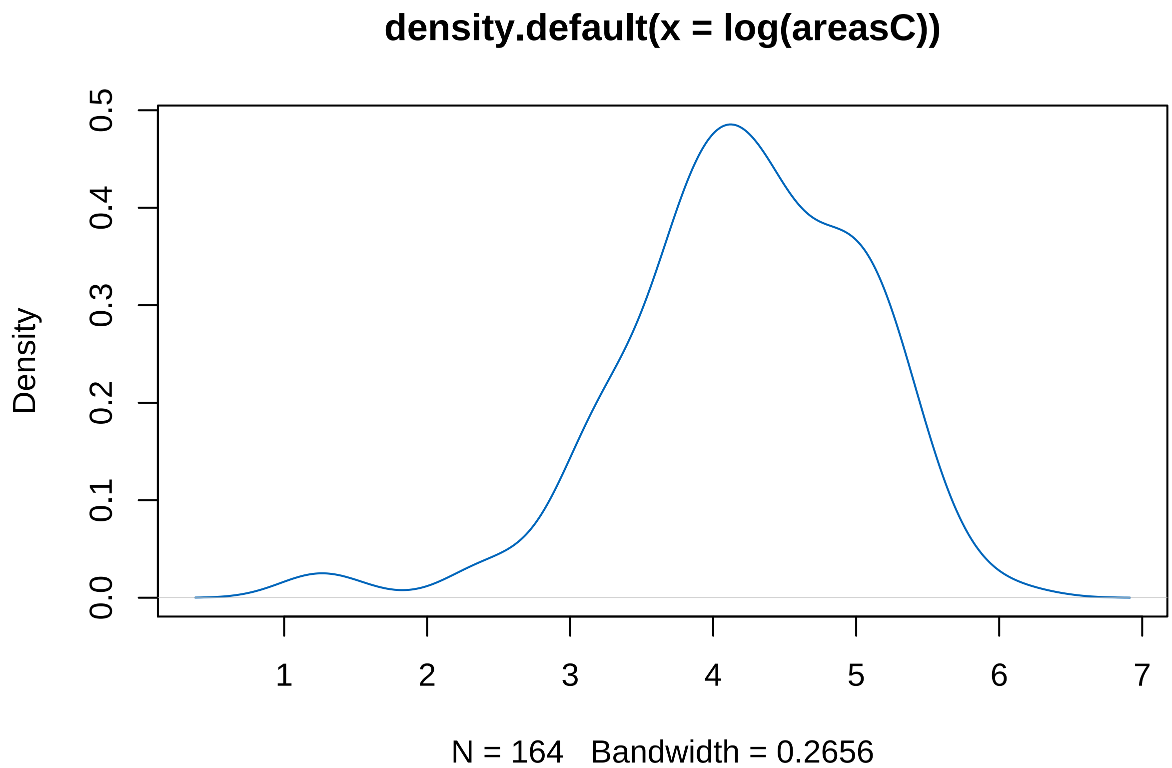
Další možnosti - exponenciální rozložení, stochasticita

Koukal jsem ještě na Wikipedii sem: https://en.wikipedia.org/wiki/Maximum_entropy_probability_distribution A log-normální rozdělení je rozdělením s maximální entropií pro čísla, která jsou kladná (záporný areál rozšíření být nemůže) a které mají netriviální varianci. Pokud ten argument tedy přeženu (ale vlastně oprávněně, vzhledem k těmhle minimálním constraintům maximální entropie), nemusím hledat důvod, proč by měl mít stochastický proces vzniku areálů multiplikativní povahu, stačí mi vědět, že areály jsou kladná čísla s nějakou variancí. Tipuju, že právě na hraně mezi exponenciálním rozdělením (kde pravděpodobnostní hustota klesá celou dobu od modu nula a kde je očekávaná variance dopočitatelná z průměru, to rozdělení má tedy jediný parametr - "rate") a tím log-normem (dva parametry - průměr logaritmů a standardní odchylka logaritmů) se povede nějaká zajímavá debata. Exponenciální rozdělení má maximální entropii pro jedinou omezující podmínku: že se jedná o kladná čísla. V případě exponenciálního rozložení velikostí bychom čekali tedy ohromné množství maličkých areálů. Třeba někdo tvrdí, že do toho modelu nemusíme vůbec zanášet varianci těch areálů jako parametr (Tohle by nevyžadovalo vůbec předpoklad, že je vývoj velikosti areálů multiplikativní proces). Tento "minimalistický model" by si šlo představit tak, že se prostě z klobouku s definovaným průměrem vylosuje kladné číslo, což je ten areál, a ty malý areály časem zaniknou díky stochasticky kolísající velikosti populace. To povede na rozdělení, které bude připomínat log-norm, ale ve skutečnosti bude parametrizované jinak.

```r
# 200 čísel z náhodného exponenciálního rozdělení
listC <- rexp(200, 1/80)

# Největší záporný výkmit (dělám jako výkmit areálu, je to jedno, vztah mezi
# areálem a populací předpokládám nějaký hodně deterministický lineární) -
# polovina normálního rozložení od nuly dolů se směrodatnou odchylkou 20 (to
# je ten druhý parametr výsledků)
max.fluct.neg <- -abs(rnorm(200, 0, 20))

# Tam, kde to největší záporný výkmit v součtu s průměrem dostane pod nulu,
# druh zmizí, jinak beru ten průměr.
areasC <- listC[(listC + max.fluct.neg) > 0]
plot(density(areasC), col = "#0066BB")
```




```r
# Jak vypadá rozdělení po zlogaritmování a jak se k tomu staví shapiro test
plot(density(log(areasC)), col = "#0066BB")
```




```
shapiro.test(log(areasC))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(areasC)
## W = 0.96229, p-value = 0.000198
```

Ono to výsledné rozložení je hodně podobné tomu log-normálnímu (obvykle tam ale chybí ty "mega velké areály"). Chtělo by to dost dat, aby se vůbec dalo rozhodnout, které to rozdělení vystihuje realitu lépe (a taky asi přidat k tomu log-normálnímu parametr té odúmrtě malých areálů a parametr přetékání velkých areálů přes okraj kontinentu.)

Varianty původní simulace s extra náhodou

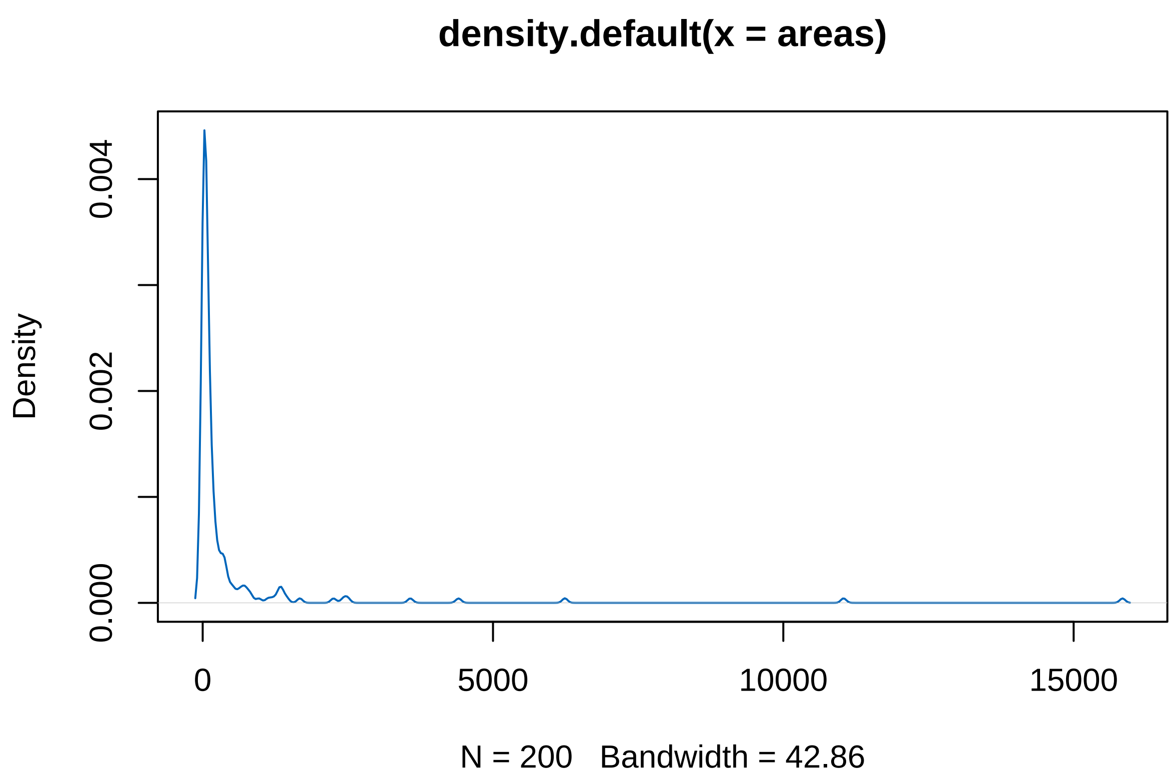
Každopádně ten multiplikativní element je tak dominantní, že i když nezačínáme v každém případě na mediánu (80), ale na náhodném čísle z nějakého aditivního rozdělení (třeba průměr 80 a SD 20), výsledek leze hodně podobný jako v té základní simulaci výše.

```
# Náhodný start kolem 80 km2 a 100 náhodných vlivů *1.2 nebo /1.2
listB <- lapply(1:200, function(i) {
    c(abs(rnorm(1, 80, 20)), sample(c(1/1.2, 1.2), 100, replace = T))
})

# Prvních 5
```

```r
str(listB[1:5])
```

```
## List of 5
##  $ : num [1:101] 82.225 0.833 0.833 0.833 0.833 ...
##  $ : num [1:101] 86.217 1.2 1.2 0.833 1.2 ...
##  $ : num [1:101] 79.307 1.2 1.2 0.833 0.833 ...
##  $ : num [1:101] 65.143 0.833 1.2 0.833 0.833 ...
##  $ : num [1:101] 48.426 1.2 0.833 0.833 1.2 ...
```

```r
areas <- sapply(listB, prod)
plot(density(areas), col = "#0066BB")
```




```r
shapiro.test(log(areas))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(areas)
## W = 0.9946, p-value = 0.6906
```

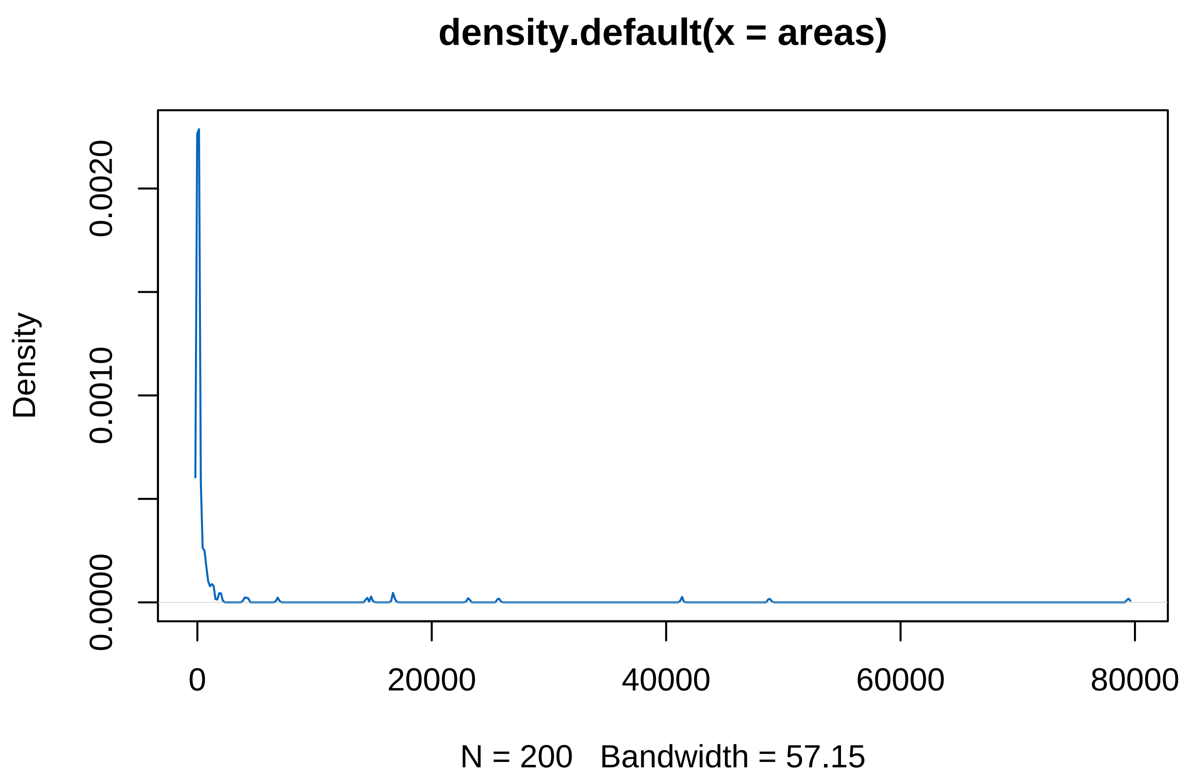
Stejně tak, když si to nebudu usnadňovat pouhými dvěma možnostmi 1.2 a 0.83, ale vezmu celý košík náhodných procentuálních změn

```r
listB <- lapply(1:200, function(i) {
    c(80, ifelse(sample(c(0, 1), 100, replace = T) == 0, 1/(1 + rexp(100, 4)),
        1 + rexp(100, 4)))
})
```

```r
# Čistě pozitivní procentuální změny, tahám je z exponenciálního rozložení s
# průměrem 0.25 (při větších hodnotách je to rozdělení šíleně sešikmené -
# pořád je lognormální, testem projde, ale graf vypadá blbě), ale je to
# jedno, mohly by být z lognormálního. Na tomhle to nezávisí.

# Prvních 5
str(listB[1:5])
```

```
## List of 5
##  $ : num [1:101] 80 1.64 1.24 1.05 1.21 ...
##  $ : num [1:101] 80 0.888 1.021 1.301 1.145 ...
##  $ : num [1:101] 80 1.569 1.015 1.735 0.665 ...
##  $ : num [1:101] 80 1.11 1.21 2.19 0.62 ...
##  $ : num [1:101] 80 1.14 1.46 1.153 0.821 ...
```

```r
areas <- sapply(listB, prod)
plot(density(areas), col = "#0066BB", )
```




```r
shapiro.test(log(areas))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(areas)
## W = 0.98759, p-value = 0.07829
```

Vychází to taky jako v pohodě log-normální rozdělení. Takhle se prostě chovají multiplikativní procesy. :)

Poslední, co bych chtěl ještě zkusit, je zapojení náhodné délky vektorů. Jakože velikost jednoho areálu ("starého") je určna dvěstě změnami, velikost jiného třeba jen deseti změnami. Zkombinuju tu úplně všechno - náhodný normálně rozložený start, a náhodný počet náhodně velkých multiplikativních změn. Shapiro testem to po zlogaritmování v pohodě projde. I takto vágně definovaný multiplikativní proces vede na log-normální rozdělení.

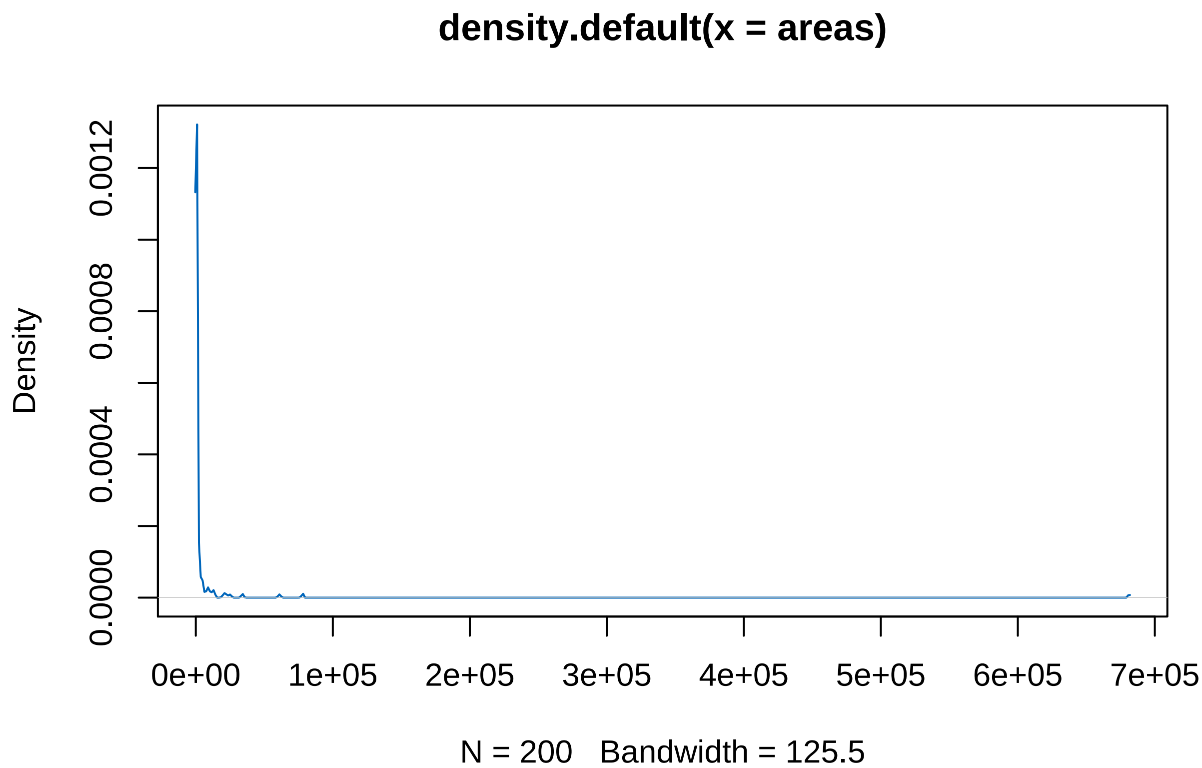
```
listB <- lapply(1:200, function(i, len = round(runif(1, 2, 200))) {
    c(abs(rnorm(1, 80, 20)), ifelse(sample(c(0, 1), len, replace = T) == 0,
        1/(1 + rexp(len, 4)), 1 + rexp(len, 4)))
})

# Zvolil jsem jako délku vektoru náhodné číslo mezi 2 a 200 (uniformní
# rozdělení)
sapply(listB, length)
```

```
##   [1] 176  50  74 161  55 142 127 117 101 146  18 171  83 132 163  45  78
##  [18]  38 166 176 177 157 170 174  42  25 199  91 115 109  32 173 155 181
##  [35] 133  11  94 154 155  98  51  67  44  44 161 180 135 174  20  42 175
##  [52] 115  48 135  21 153 145  49 123 197  18 153  81  32 187   7 199 169
##  [69]  66  91 155  76  54  92  27  36 119 111 146 123 138  16 128  16 100
##  [86]  55  77  75  10 118  16 172  42 123 119 109  22 104  74  13  32   8
## [103]   7 180 192  10 121  41  10  99  94 108 128  91  33 119  19 140 178
## [120]  65 123  62  54 150  93  47  92  83  36 135 150  37  44 192 197 182
## [137]  52 125 136  75  39 149  93  85 178 151 103 174 130  36 195  45  74
## [154] 172  63 142  35  80  66  54 190  94 173  67 166  59  11  30  97 175
## [171] 107 143   8 196 102 190  54  45 124 193  63  51 167 177 155  84  68
## [188] 174 103  30   5  79  91 113  45 167  49 129 101  31
```

```
# Prvních 5
str(listB[1:5])
```

```
## List of 5
##  $ : num [1:176] 81.941 0.947 0.966 1.439 2.645 ...
##  $ : num [1:50] 72.719 1.044 1.298 1.327 0.794 ...
##  $ : num [1:74] 95.634 0.967 1.165 1.518 0.969 ...
##  $ : num [1:161] 64.465 1.453 1.33 0.915 0.846 ...
##  $ : num [1:55] 86.998 0.76 0.815 0.625 0.588 ...
```

```
areas <- sapply(listB, prod)
plot(density(areas), col = "#0066BB", )
```




```r
shapiro.test(log(areas))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(areas)
## W = 0.9965, p-value = 0.9323
```