随机数生成及其在统计模拟中的应用

黄湘云

1 引言

随机数是统计模拟的基础，均匀分布随机数又是其中最重要的，因为由均匀分布随机数可以产生其它分布的随机数，所以好的随机数发生器，其实指的就是一个高质量、高效的均匀分布随机数发生器。高质量就是经得起纯随机性检验，高效就是要产生速度快。话不多说，先加载一些必要的 *R* 包。

```
library(ggplot2)
library(viridisLite)
library(viridis)
library(gridExtra)
library(R.matlab)
```

2 随机数生成

讲随机数发生器，不得不提及 Mersenne Twister（简称 MT），它的周期长达 $2^{19937}-1$，现在是 R 、Octave 和 Matlab 等软件（较新版本）的默认随机数发生器。在产生 2^{24} 个随机数的过程中，MT 发生器的 C 语言 64 位版本 [1]，我在 Dev-C++ 5.11 上编译运行 10 秒左右。（由于整个代码比较长就不贴了）

Matlab 新版本内置了早期的随机数发生器（这里以 1995 年的 Matlab 随机数发生器为例），两行代码就可以产生 2^{24} 个随机数。

```
% matlab code % 大约几秒
rng(1234, 'v5uniform')
x = rand(1,2^24);
```

randtx [2] 是实现这个随机数发生器的源代码（打包在 NCM 工具箱内），我把代码略作改动以适应在 Octave 中运行（没有 Matlab 照样玩得转），下面在 Octave 内产生可重复的随机数 2^{24} 个（1600 多万），保存成 mat 格式文件，方便后续检验使用。这一番从 Matlab 到 Octave 的折腾，只是为了避免使用 Matlab 而已，再说 Octave 对 Matlab 的兼容性很高（涉及 GUI 编程的代码是不兼容的，Octave 貌似只专注计算http://www.gnu.org/software/octave/about.html）。

```
% octave code
id = tic % 时间耗费大约一小时
randtx("state",0)
```

[1] http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt64.html

[2] https://www.mathworks.com/moler/chapters.html

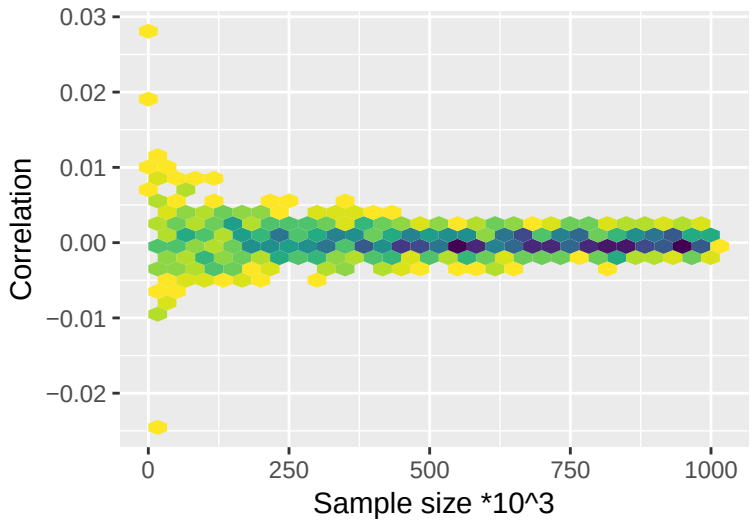
```
x = randtx(1,2^24);
toc (id)
save -mat random_number.mat x # 保存随机数到文件
```

3 统计检验

3.1 相关性检验

先来一个简单的，就用 *R* 产生的两个独立同均匀分布样本，调用 **cor.test** 做相关性检验，然后眼球验证一下，随着样本量增大，相关性趋于 0。

```
set.seed(1234)
corr <- rep(0,1000)
for(i in seq(from=1000,to=1000000,by=1000)){
    corr[i/1000] <-  cor.test(runif(i,min = 0,max = 1),
        runif(i,min = 0,max = 1))$estimate }
ggplot(data.frame(x = seq(1000), y = corr), aes(x = x, y = y)) +
    geom_hex(show.legend=FALSE)+
    scale_fill_viridis(direction = -1) + xlab("Sample size *10^3")+ylab("Correlation")
```



3.2 分布检验

检验产生的随机数是否服从指定的分布：原假设是样本来自指定的分布，计算的 P 值比较大，就不能拒绝原假设。

```
ks.test(runif(1000),"punif") # 分布检验
```

```
##
```

```
##  One-sample Kolmogorov-Smirnov test
##
## data:  runif(1000)
## D = 0.022302, p-value = 0.7025
## alternative hypothesis: two-sided
```

检验两样本是否来自同一分布：原假设是两样本来自同一分布，计算的 P 值比较小，就表示两样本不是来自同一分布。

```
ks.test(runif(1000),runif(1000)) # 同分布检验
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  runif(1000) and runif(1000)
## D = 0.04, p-value = 0.4005
## alternative hypothesis: two-sided
```

从结果来看，*R* 内置的随机数发生器通过了检验（嘿嘿，这是肯定的!!）。

3.3 游程检验

游程检验对随机数的随机性检验就相对严格，是一种非参数检验。先略作解释，简单起见，我们考虑 0-1 序列，抛掷均匀的硬币 1000 次，正面向上记为 1，反面向上记为 0，这是一个离散的均匀分布，每一次抛掷硬币都无法准确地判断出现的是正面还是反面，若记录的序列中 0 和 1 相对集中的出现，显然不是随机，0 和 1 交替出现，呈现周期性也不是随机，除了这两种情况基本就是随机了。游程检验的原假设是序列随机的，当计算的 P 值比较大时，不能拒绝原假设，即不能否认这个序列是随机的。

```
library(tseries)
x <- sample(c(0,1),1000,replace = TRUE,prob = c(1/2,1/2))
runs.test(factor(x))
```

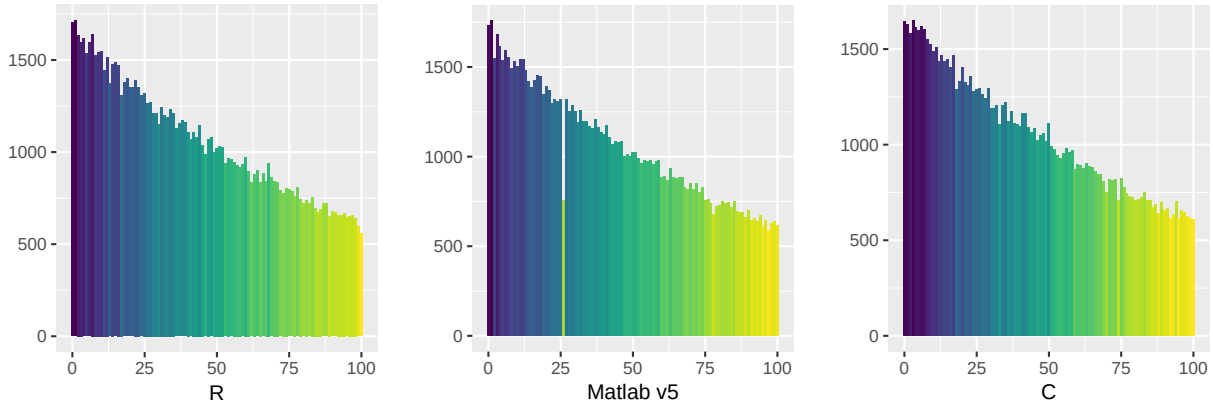
```
##
##  Runs Test
##
## data:  factor(x)
## Standard Normal = 0.45116, p-value = 0.6519
## alternative hypothesis: two.sided
```

我们现在拿 Matlab 早期随机数发生器、R 内置的 Mersenne-Twister 发生器和 C 语言实现的 Mersenne-Twister 发生器比较，程序实现仿 Matlab 版的游程检验[3]。

```
# 游程频数直方图
run_test_fun <- function(x,string,delta) {
  n <- length(x)
```

[3]https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/random.pdf

```
  len <- diff(c(0,which(x<delta),n+1))-1
  ggplot(data.frame(x=len[len < 101]),aes(x,fill=..count..)) +
    scale_fill_viridis(direction = -1)+
    geom_histogram(binwidth = 1,show.legend = FALSE) +
    xlab(string)+ylab("")
}
set.seed(1234) # R 默认采用 Mersenne Twister 发生器
r_data <- runif(2^24,0,1); # R 内生成均匀分布随机数
matlabv5_data <- readMat("random_number.mat") # 读取 Octave 生成的均匀分布随机数
temp <- read.table(file = "random_number.txt") # 读取 C 语言生成的均匀分布随机数
c_data <- c(as.matrix(t(temp)))
p1 <- run_test_fun(x = r_data,string = "R",delta = 0.01)
p2 <- run_test_fun(x = matlabv5_data$x,string = "Matlab v5",delta = 0.01)
p3 <- run_test_fun(x = c_data,string = "C",delta = 0.01)
grid.arrange(p1, p2, p3, ncol=3)
```



在游程长度为 27 的位置，有一条深沟，这是 George Marsaglia 提出的借位相减（subtract-with-borrow）算法的特性，显然不符合随机性的要求[4]。

4 应用

4.1 两个均匀分布的统计模拟

随机变量 $X_1, X_2 \overset{iid}{\sim}$ 某分布（比如二项分布，泊松分布，正态分布，指数分布，卡方分布，伽马分布），则 $X_1 + X_2$ 也服从该分布。常见的均匀分布是否具有这样的可加性？具体地说，就是 $X_1, X_2 \overset{iid}{\sim} U(0,1)$ ，$X_1 + X_2$ 是否服从 $U(0,2)$？如果有一台电脑在旁边，我首先想到的就是敲三五行代码，画个散点图、直方图，看图说话。

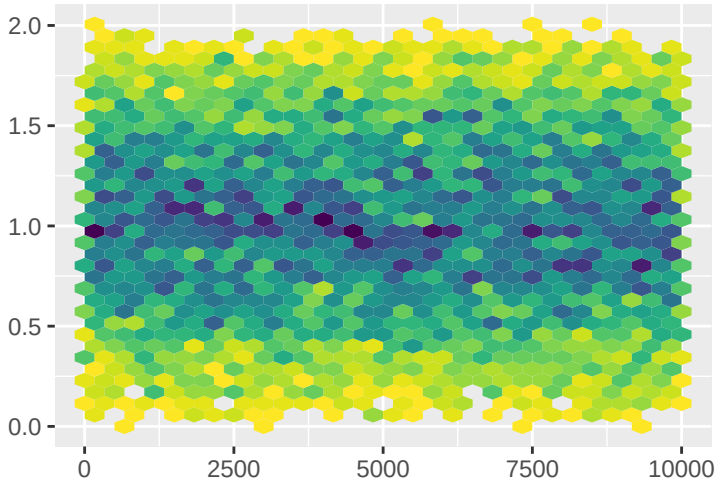
[4] https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/random.pdf

```r
set.seed(1234)
x <- runif(10000,min = 0,max = 1)
y <- runif(10000,min = 0,max = 1)
z <- x+y
plot(z) # 散点图
```

```r
hist(z) # 直方图
```

为美观起见，多写了一行，图就可以长下面那样

```r
ggplot(data.frame(x = seq(10000), y = z), aes(x = x, y = y)) +
    geom_hex(show.legend=FALSE)+
    scale_fill_viridis(direction = -1) + xlab("")+ylab("")
```

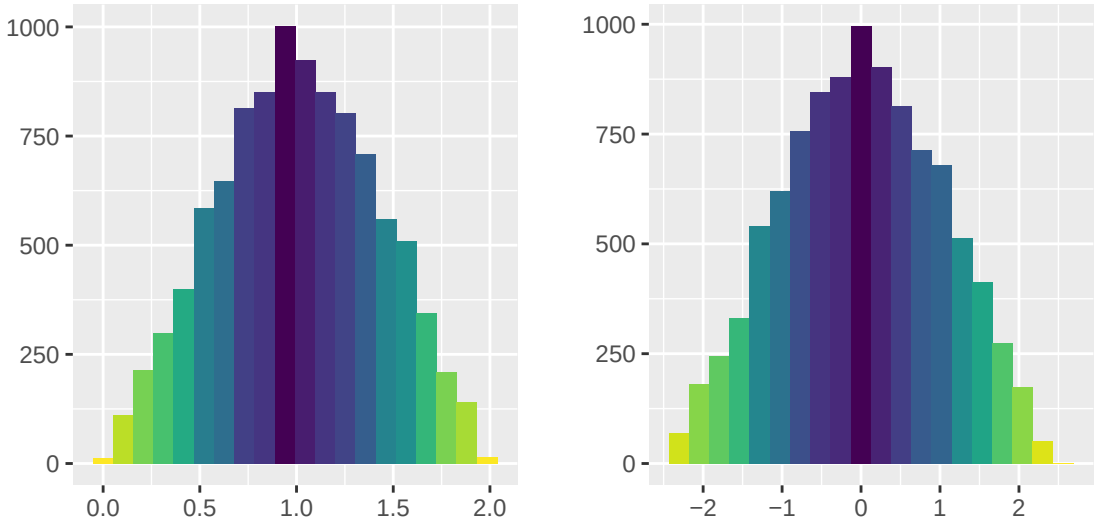


显然这不是均匀分布，在 $z=1$ 处，散点比较集中，看起来有点像正态分布，如果往中心极限定理上靠，将作如下标准化

$$Y_2^\star = \frac{X_1+X_2-2*\frac{1}{2}}{\sqrt{\frac{1}{12}}*\sqrt{2}} = \sqrt{6}(X_1+X_2-1)$$

则 $Y_2^\star$ 的期望为 0，方差为 1。

```r
p4 <- ggplot(data.frame(x=z),aes(x,fill=..count..))+
    scale_fill_viridis(direction = -1)+
    geom_histogram(bins=20,show.legend=FALSE) + xlab("")+ylab("")
p5 <- ggplot(data.frame(x=sqrt(6)*(z-1)),aes(x,fill=..count..))+
    scale_fill_viridis(direction = -1)+
    geom_histogram(bins=20,show.legend=FALSE) + xlab("")+ylab("")
grid.arrange(p4, p5,  ncol=2)
```



只是变换后的图像和之前基本一致，那么现在看来眼球检验不好使了，那就上 P 值呗！

```
ks.test(sqrt(6)*(z-1),"pnorm") # 分布检验
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sqrt(6) * (z - 1)
## D = 0.025778, p-value = 3.381e-06
## alternative hypothesis: two-sided
```

也不是正态分布，既然如此，那就在两个随机变量的情况下，把精确分布推导出来。

4.2 精确分布的推导及计算

课本如《概率论与数理统计教程》采用卷积的方法求分布函数，这种方法实行起来比较繁琐，也不利于后续编程，下面考虑用特征函数的方法求。我们知道标准均匀分布的特征函数

$$\varphi(t)=\frac{e^{it}-1}{it}$$

考虑 X_1 和 X_2 相互独立，它们的和用 S_2 表示，则随机变量 S_2 的特征函数为

$$\varphi_2(t)=\varphi(t)*\varphi(t)=(\frac{e^{it}-1}{it})^2=\frac{2(1-\cos(t))e^{it}}{t^2}$$

只要满足条件

$$\int_{-\infty}^{+\infty}|\varphi_2(t)|\mathrm{d}t<\infty$$

S_2 的密度函数就可以表示为

$$p_2(x)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\mathrm{e}^{-itx}\varphi_2(t)\mathrm{d}t$$

经计算

$$\int_{-\infty}^{+\infty}|\varphi_2(t)|\mathrm{d}t=4\int_0^{+\infty}\frac{1-\cos(t)}{t^2}\mathrm{d}t=4\int_0^{+\infty}\Big(\frac{\sin(x)}{x}\Big)^2\mathrm{d}x=2\pi$$

那么

$$p_2(x)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\mathrm{e}^{-itx}\varphi_2(t)\mathrm{d}t=\frac{2}{\pi}\int_0^{+\infty}\frac{(1-\cos(t))\cos(t(1-x))}{t^2}\mathrm{d}t=\frac{2}{\pi}\int_0^{+\infty}\cos\Big(2(1-x)t\Big)\Big(\frac{\sin(t)}{t}\Big)^2\mathrm{d}t$$

一般地，n 个独立随机变量的和

$$\varphi_n(t)=\Big(\frac{e^{it}-1}{it}\Big)^n=\Big(\frac{\sin(t/2)\mathrm{e}^{\frac{it}{2}}}{t/2}\Big)^n$$

那么，同理

$$p_n(x)=\frac{2}{\pi}\int_0^{+\infty}\cos\Big(2(n/2-x)t\Big)(\frac{\sin(t)}{t})^n\mathrm{d}t$$

要说数值计算一个 $p(x)$ 近似值，是一点问题没有！且看

```
integrate(function(t,x,n) 2/pi*cos((n-2*x)*t)*(sin(t)/t)^n ,x = 1,n = 2,
            lower = 0,upper = Inf,subdivisions = 1000)
```

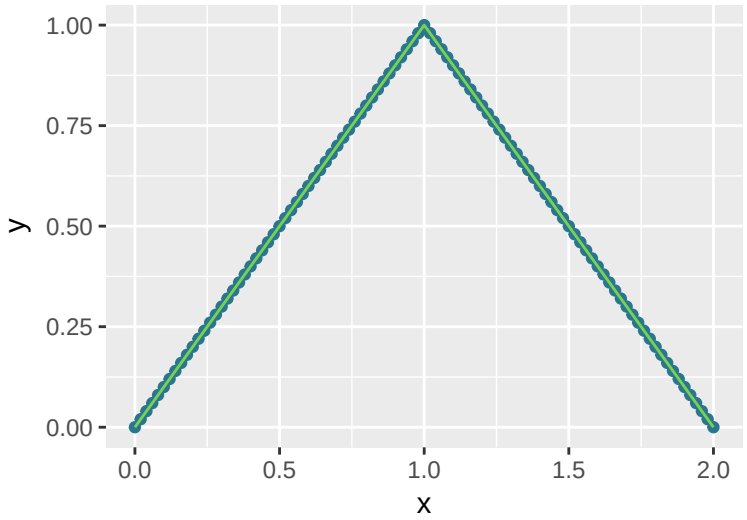
```
## 0.9999846 with absolute error < 6.6e-05
```

那如果要把上面的积分积出来，获得一个精确的表达式，在 $n=2$ 的时候还可以手动计算，主要使用分部积分，余弦积化和差公式和一个狄利克雷积分公式 $\int_0^{+\infty}\frac{\sin(ax)}{x}\mathrm{d}x=\frac{\pi}{2}\mathrm{sgn}(a)$，过程略，最后算得

$$p_2(x)=\frac{1}{2}\Big((2-x)\mathrm{sgn}(2-x)-x\mathrm{sgn}(-x)\Big)-(1-x)\mathrm{sgn}(1-x)=\frac{1}{2}(|x|+|x-2|)-|x-1|\,,0<x<2$$

$p_2(x)$ 的密度函数图象如下：

```
fun_p2_1 <- function(x) { 1 / 2 * (abs(x - 2) - 2 * abs(x - 1) + abs(x)) }
fun_p2_2 <- function(x) {
    x <- as.matrix(x)
    tempfun <- function(x) {
        integrate(function(t, x, n) 2 / pi * cos((n - 2 * x) * t) * (sin(t) / t) ^ n,
            x = x, n = 2,lower = 0, upper = Inf, subdivisions = 1000)$value
    }
   return( sapply(x,tempfun) )
}
ggplot(data.frame(x = c(0, 2)), aes(x = x)) +
    stat_function(fun = fun_p2_2, geom = "point", colour = "#2A768EFF") +
    stat_function(fun = fun_p2_1, geom = "line", colour = "#78D152FF")
```



从图中可以看出，两种形式的密度函数在数值计算的结果上很一致，当 $n = 100, 1000$ 时，含参量积分的表示形式就很方便啦！任意给定一个 n，符号计算上面的含参量积分，这个时候还是用软件计算比较合适，R 的符号计算仅限于求导，积分运算需要借助 Ryacas，rSymPy，可惜的是，这些包更新缓慢，即使 $\int_0^{+\infty} \frac{\sin(at)}{t} \mathrm{d}t$ 也算不出来，果断直接使用 Python 的 sympy 模块

```
from sympy import *
a=symbols('a', real=True)
t=symbols('t', real=True,positive=True)
print(integrate(sin(a*t)/t,(t,0,oo)))
```

```
## Piecewise((pi/2, Eq(Abs(periodic_argument(polar_lift(a)**2, oo)), 0)), (Integral(sin(a*t)/t,
```

。。。初次见到这样的结果，是不是一脸 mb，翻译一下，就是

$$\begin{cases} \frac{\pi}{2} & \text{for } \left|\text{periodic}_{\text{argument}}\left(\text{polar\_lift}^2\left(a\right), \infty\right)\right| = 0 \\ \int\limits_0^{\infty} \frac{1}{t} \sin\left(at\right) dt & \text{otherwise} \end{cases}$$

稍为好点，但是还是有一大块看不懂，那个绝对值里是什么？还是不要纠结了，路远坑多，慢走不送啊！话说要是计算 $p_2(x)$ 密度函数里的积分，

```
from sympy import *
x=symbols('x', real=True)
t=symbols('t', real=True,positive=True)
print(integrate(2/pi*cos(2*t*(1-x))*(sin(t)/t)**2,(t,0,oo)))
```

```
## Piecewise((Piecewise((2*x, (2*x - 2)**2/4 < 1), (0, 4/(2*x - 2)**2 < 1), (meijerg(((1/2,), (
```

那就更长了。。。

$$\begin{cases} \begin{cases} 2x & \text{for } \frac{1}{4}\left(2x-2\right)^{2}<1 \\ 0 & \text{for } \frac{4}{\left(2x-2\right)^{2}}<1 \\ G_{4,4}^{3,1}\left(\begin{matrix}\frac{1}{2} & 1,1,\frac{3}{2}\\ \frac{1}{2},1,0 & \frac{1}{2}\end{matrix}\middle|\frac{1}{4}\,\text{polar\_lift}^{2}\left(-2x+2\right)\right) & \text{otherwise} \end{cases} & \text{for } \left|\text{periodic}_{\text{argument}}\left(\text{polar\_lift}^{2}\left(-2\right.\right.\right. \\ \int_{0}^{\infty}\frac{2}{\pi t^{2}}\sin^{2}\left(t\right)\cos\left(2t\left(-x+1\right)\right)dt & \text{otherwise} \end{cases}$$

sympy 模块还是比较强的，化简可能比较弱，感觉是我的条件声明没有充分利用，要看懂，得知道一些复变函数的知识，这个时候，可以试试 Maple 或者 Mathematica，面对高昂的费用，我们可以使用在线的免费计算 WolframAlpha（http://www.wolframalpha.com/），输入

```
integrate 2/pi*cos(2*t*(1-x))*(sin(t)/t)^2 ,t ,0,oo
```

即可得 $p_2(x)=\frac{1}{2}(|x-2|-2\,|x-1|+|x|)$，$n$ 取任意值都是可以算的，由于式子比较复杂，就不展示了。

5 软件信息

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8.1 x64 (build 9600)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Chinese (Simplified)_China.936
## [2] LC_CTYPE=Chinese (Simplified)_China.936
## [3] LC_MONETARY=Chinese (Simplified)_China.936
## [4] LC_NUMERIC=C
## [5] LC_TIME=Chinese (Simplified)_China.936
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] tseries_0.10-40   hexbin_1.27.1     R.matlab_3.6.1    gridExtra_2.2.1
## [5] viridis_0.4.0     viridisLite_0.2.0 ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
```

```
##  [1] Rcpp_0.12.10      knitr_1.15.1      magrittr_1.5
##  [4] munsell_0.4.3     lattice_0.20-35   colorspace_1.3-2
##  [7] quadprog_1.5-5    stringr_1.2.0     plyr_1.8.4
## [10] tools_3.4.0       grid_3.4.0        gtable_0.2.0
## [13] R.oo_1.21.0       htmltools_0.3.6   yaml_2.1.14
## [16] lazyeval_0.2.0    rprojroot_1.2     digest_0.6.12
## [19] tibble_1.3.0      codetools_0.2-15  R.utils_2.5.0
## [22] evaluate_0.10     rmarkdown_1.5     labeling_0.3
## [25] stringi_1.1.5     compiler_3.4.0    scales_0.4.1
## [28] backports_1.0.5   R.methodsS3_1.7.1 zoo_1.8-0
```