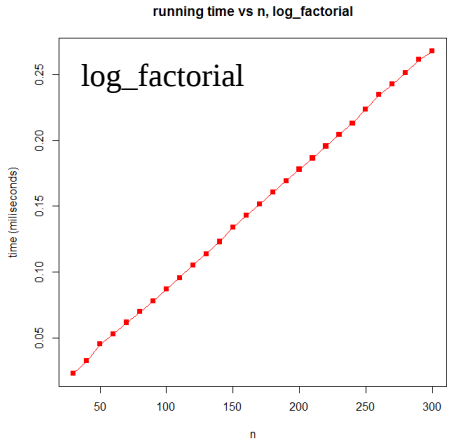
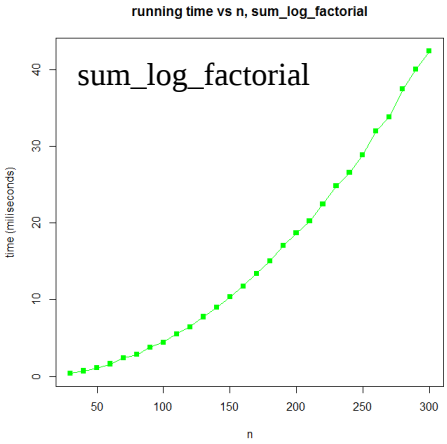
Title: ac2

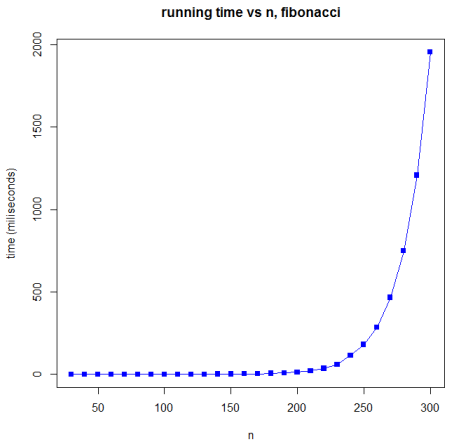
Date: September 26, 2017

3 separate plots showing running time vs n for the 3 functions.





fibonacci



For each function, its time complexity in Big-O notation.

function	Big-O notation
log_factorial	O(n)
sum_log_factorial	$O(n^2)$
fibonacci	$O(2^n)$

Code snippet I used to compute and plot the run times.

```
# Code snippet to compute and plot the run time vs n for log_factorial function
library(microbenchmark)
N=seq(30,300,by=10)
time1=array(dim=length(N))
i=1
for (n in N){
  time1[i]=median(microbenchmark (log_factorial(n))$time,times=1000)/1000000
  i=i+1
}
plot(N,time1, xlab="n", ylab="time (miliseconds)", main="running time vs n, log_factorial",type="o",pch=15, col="red")

# Code snippet to compute and plot the run time vs n for sum_log_factorial function
N=seq(30,300,by=10)
time2=array(dim=length(N))
```

```
i=1
for (n in N){
  time2[i]=median(microbenchmark (sum_log_factorial(n))$time,times=1000L)/1000000
  i=i+1
}
plot(N,time2, xlab="n", ylab="time (miliseconds)", main="running time vs n, sum_log_factorial",type="o",pch=15, col="green")

# Code snippet compute and plot the run time vs n for fibonacci function
N=seq(3,30,by=1)
time3=array(dim=length(N))
i=1
for (n in N){
  time3[i]=median(microbenchmark (fibonacci(n))$time)/1000000
  i=i+1
}
plot(N,time3, xlab="n", ylab="time (miliseconds)", main="running time vs n, fibonacci",type="o",pch=15, col="blue")
```