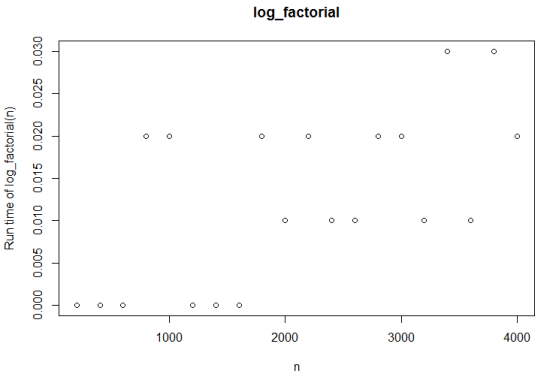
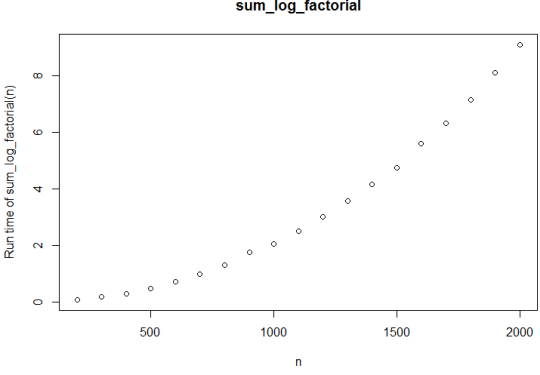
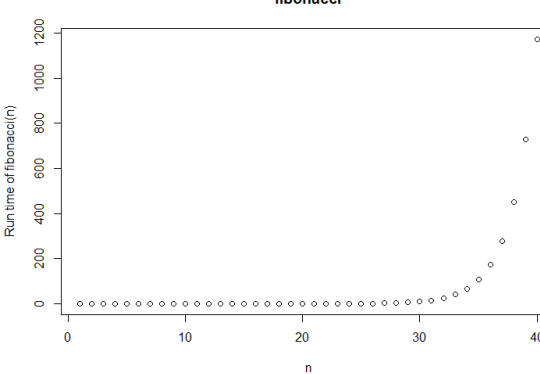


Function, Time Complexity	Plots	Code
log factorial $O(n)$		<pre>options(expressions=500000) log_factorial &lt;- function(n) {   # Return the log of factorial(n) for any integer n &gt; 0   if (n &lt;= 1)     return(0)   return(log(n) + log_factorial(n - 1)) }  sum_log_factorial &lt;- function(n) {   # Return the sum of log_factorial(i) for i in 1..n   sum &lt;- 0   for(i in seq(1, n, 1)) {     sum &lt;- sum + log_factorial(i)   }   return(sum) }  fibonacci &lt;- function(n) {   # Return nth Fibonacci number   if (n &lt;= 1)     return(n)   return(fibonacci(n - 1) + fibonacci(n - 2)) }</pre>
Sum Log Factorial $O(n^2)$		<pre>n_l&lt;-seq(200,4000,200) RT_log_f&lt;- sapply(n_l,   function(x) system.time(log_factorial(x))[1]) plot(n_l, RT_log_f, main="log_factorial", xlab='n',   ylab='Run time of log_factorial(n)')  n_s&lt;-seq(200,2000,100) RT_sumlog_f&lt;- sapply(n_s,     function(x) system.time(sum_log_factorial(x))[1]) plot(n_s, RT_sumlog_f, main="sum_log_factorial",   xlab='n', ylab='Run time of sum_log_factorial(n)')</pre>
Fibonacci $O(2^n)$		<pre>n_f&lt;-seq(1,40,1) RT_fib&lt;- sapply(n_f,   function(x) system.time(fibonacci(x))[1]) plot(n_f, RT_fib, main="fibonacci", xlab='n',   ylab='Run time of fibonacci(n)')</pre>