# Supplementary Materials for **Relatively semi-conservative replication and a folded slippage model for** **simple repeat sequences** Hongxi Zhang1,#, Douyue Li1,#, Xiangyan Zhao1,#, Saichao Pan1,#, Xiaolong Wu1, Shan Peng1, HanrouHuang1, Ruixue Shi1 and Zhongyang Tan1,\* 1Bioinformatics Center, College of Biology, Hunan University, Changsha 410082, China #Co-first author \*Corresponding author: Zhongyang Tan Email: zhongyangtan@yeah.net

This file includes:  
supplementary figs. S1-S4

supplementary program S1

Other Supplementary Materials for this manuscript include the following:supplementary tables S1 and S2

## supplementary fig. S1

supplementary fig. S1.2

Impossible straight slippage models for tetra- to hexanucleotide repeats when the slippage bubble occurs at the replicating strand. The model drawing was based on the strict geometric calculation of the space of a nucleotide and the stability of hydrogen and phosphodiester bonds.

## supplementary fig. S2

supplementary fig. S2.2

Impossible curved slippage models for tri- to hexanucleotide repeats when the template strand in the inner side of the models.

## supplementary fig. S3

The possible folded slippage models for hepta- to decanucleotide repeat amplification. Repeat units tend to be expanded in the replicating strands when the template strands are on the inner side of the folded slippage models respectively.

## supplementary fig. S4

The possible folded slippage models for hepta- to decanucleotide repeat contraction. Repeat units tend to be subtracted in the replicating strands when the template strands are on the outside of the folded slippage models respectively.

## Supplementary program 1

The random model written by C language (which was written by one of our authors named Xiaolong Wu) was designed to generate random sequences, and the number of adenine, thymine, guanine and cytosine in each random sequence are same to corresponding reference sequence downloaded from the database. The program is run by Vsial C 6.0 in windows 7. Once the program is successfully compiled, the software can be directly used without installation.

The loop process of generated a random sequence corresponding to reference sequence with length of *L* (the counts of each base: A=m, T=n, G=p, C=q and m+n+p+q=*L*): 1) input m, n, p and q, then a nonrandom sequence is generated; 2) a random number *N* is generated by computer system, and *N* is controlled between 0 and *L*; 3) copy the base with order number *N* from nonrandom sequence to random sequence; 4) in nonrandom sequence, move the base with order number *L* to the position of the base with order number *N* and cover it; 5) *N* and *L* substract 1 at the same time, and a new *N* and *L* are generated; 6) loop the steps from 2 to 6 until generate a complete random sequence.

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main()

{ printf("The randomly generated sequence will be saved at C://sequence.txt!!!\n");

int a,t,g,c,genome;

do

{

printf("Enter the number of each base(the genome size<=500000 bp), respectively!:\nA=");

scanf("%d",&a);

printf("T=");

scanf("%d",&t);

printf("G=");

scanf("%d",&g);

printf("C=");

scanf("%d",&c);

genome=a+t+c+g;

if(genome>500000)

{

printf("Wrong!!! The length of genome is more than 500000 bp!!!\nPlease try again!\n");

}

}while(genome>500000);

int n,i,p,q,k;

k=genome;

char G[500000],GG[500000];

int num[4]={0,0,0,0};

srand((unsigned)time(NULL));

for (i=0;i<genome;)/\*generate nonrandom sequence corresponding to reference sequence \*/

{

n=rand()%4+1;

if(n==1&&num[0]<a)

{

G[i]='A';

num[0]++;

i++;

}

if(n==2&&num[1]<t)

{

G[i]='T';

num[1]++;

i++;

}

if(n==3&&num[2]<g)

{

G[i]='G';

num[2]++;

i++;

}

if(n==4&&num[3]<c)

{

G[i]='C';

num[3]++;

i++;

}

}

for (i=0;k-1>=0;i++)/\*generate the random sequence from nonrandom sequence\*/

{

p=rand()%10+1;

q=rand();

n=(p\*q+q)%k;

GG[i]=G[n];

G[n]=G[k-1];

k--;

}

FILE \*fp=fopen("C://sequence.txt","w");/\*save the random sequence\*/

fprintf(fp,"%s",">random sequence\n");

for(i=0;i<genome;i++)

{

fprintf(fp,"%c",GG[i]);

}

fputs("\n\n",fp);

fclose(fp);

printf("\ngenome=%d\nFinished!!!\nPlease check file in the disk C!!!!!!!!!\n",genome);

system("pause");

return 0;

}