randomizeSeq, v. 0.8: Randomize Biological Sequences

Bernhard Haubold
Max-Planck-Institute for Evolutionary Biology, Plön, Germany

March 7, 2012

1 Introduction

Randomized sequences are the starting point for many sequence analysis tasks. The program randomizeSeq is a simple tool for efficiently shuffling sequences.

2 Getting Started

randomizeSeq was written in C on a computer running Linux and should work on any standard UNIX system. However, please contact me at haubold@evolbio.mpg.de if you have any problems with the program.

- Unpack the program

  ```
  tar -xvzf randomizeSeq_XXX.tgz
  ```

  where XXX indicates the version.

- Change into the newly created directory

  ```
  cd RandomizeSeq_XXX
  ```

  and list its contents

  ```
  ls
  ```

- Generate randomizeSeq

  ```
  make
  ```

- List its options

  ```
  ./randomizeSeq -h
  ```

- randomizeSeq takes FASTA-formatted input

  ```
  ./randomizeSeq testSeq.fasta
  ```

- The user can set the line length and the number of iterations

  ```
  ./randomizeSeq -l 2 -n 3 testSeq.fasta
  ```

3 Listing

The following listing documents the driver program for `randomizeSeq`.

```
/***** randomizeSeq.c ****************************
 * Description: In-place randomization of bio-
 *   logical sequences.
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Wed Mar  7 12:02:22 2012
 ***********************************************/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "sequenceData.h"
#include "stringUtil.h"
#include "interface.h"
#include "eprintf.h"
#include "ran.h"

void scanFile(FILE *fp, Args *args);

int main(int argc, char *argv[]){
  int i, idum;
  char *version;
  Args *args;
  FILE *fra, *fp;

  version = "0.8";
  setprogname2("randomizeSeq");
  args = getArgs(argc, argv);

  if(args->s != 0){
    idum = args->s;
  }else if((fra = fopen("randomSeed.dat","r")) != NULL){
    i = fscanf(fra,"%d",&idum);
    fclose(fra);
  }else{
    idum = -time(NULL);
  }
  init_genrand(idum);


  if(args->h || args->e)
    printUsage(version);
  if(args->v)
    printSplash(version);
  if(args->numInputFiles == 0){
    fp = stdin;
    scanFile(fp, args);
  }else{
    for(i=0;i<args->numInputFiles;i++){
      fp = efopen(args->inputFiles[i],"r");
      scanFile(fp, args);
      fclose(fp);
```

```
    }
  }
  if(args->s == 0){
    fra = fopen("randomSeed.dat","w");
    fprintf(fra,"%d\n",(int)genrand_int32());
    fclose(fra);
  }
  free(args);
  free(progname());
  return 0;
}

void scanFile(FILE *fp, Args *args){
  int c, i, k, residues, r;
  char tmp;
  Sequence *seq;

  while((seq = getNextSequence(fp)) != NULL){
    residues = seq->len - 1; /* - 1 to leave out border */
    for(k=0;k<args->n;k++){
      /* permutate sequence */
      for(i=residues-1;i>=0;i--){
        r = (int)(genrand_real1()*i);
        tmp = seq->seq[i];
        seq->seq[i] = seq->seq[r];
        seq->seq[r] = tmp;
      }
      if(args->n > 1)
        printf("%s - PERMUTATED #%d\n",chomp(seq->id),k+1);
      else
        printf("%s - PERMUTATED\n",chomp(seq->id));
      /* output sequence */
      c = 0;
      for(i=0;i<residues;i++){
        if(c == args->l){
          printf("\n");
          c = 0;
        }
        printf("%c",*(seq->seq+i));
        c++;
      }
      if(c-1 != args->l)
        printf("\n");
    }
    seq = freeSequence(seq);
  }
}
```

4 Change Log

- Version 0.8 (7th March 2012)
  - First released version

- Cleaned up code.