`sam2pro`, v. 0.3: Convert SAMtools-Pileup to Profiles

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

May 7, 2013

1 Introduction

`sam2pro` converts the pileup format generated by SAMtools to the profiles read by `mlRho` [1]. The pileup variant of SAMtools is described here

```
http://samtools.sourceforge.net/pileup.shtml
```

2 Getting Started

`sam2pro` was written in C on a computer running Mac OS X; it is intended to run on any UNIX system with a C compiler installed. However, please drop me a line at `haubold@evolbio.mpg.de` if you have problems with the program.

- Unpack the program

  ```
  tar -xvzf sam2pro_.tgz
  ```

  where `XXX` indicates the version.
- Change into the newly created directory

  ```
  cd Sam2pro_XXX
  ```

  and list its contents

  ```
  ls
  ```
- Generate `sam2pro`

  ```
  make
  ```
- List its options

  ```
  ./sam2pro -h
  ```
- Test the program

  ```
  ./sam2pro test.sam > test.pro
  ```
- The contents of the file `test.pro` can now be analyzed using `mlRho`:

  ```
  formatPro test.pro
  mlRho -M 0
  ```

3 Listing: `sam2pro.c`

```c
/***** sam2pro.c **********************************
 * Description: Convert sam output to profiles.
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Wed Jul 21 22:46:11 2010
 **************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include "interface.h"
#include "stringUtil.h"
#include "eprintf.h"
#include "tab.h"

void runAnalysis(FILE *fp, Args *args, int *dic);
void scanCol(char *column, int *dic, int *count, char consensus, char *
    number);

int main(int argc, char *argv[]){

  Args *args;
  char *version;
  FILE *fp;
  int i;
  int dic[256];

  version = "0.3";
  setprogname2("sam2pro");
  args = getArgs(argc, argv);

  for(i=0;i<256;i++)
    dic[i] = 4;
  dic['A'] = 0;
  dic['C'] = 1;
  dic['G'] = 2;
  dic['T'] = 3;
  dic['a'] = 0;
  dic['c'] = 1;
  dic['g'] = 2;
  dic['t'] = 3;

  if(args->h || args->e)
    printUsage(version);
  tabSetFieldSep(args->d);
  if(args->numInputFiles == 0){
    fp = stdin;
    runAnalysis(fp, args, dic);
  }else
    for(i=0;i<args->numInputFiles;i++){
      fp = efopen(args->inputFiles[i],"r");
      runAnalysis(fp, args, dic);
      fclose(fp);
    }
  free(args);
  free(progname());
  return 0;
}

void runAnalysis(FILE *fp, Args *args, int *dic){
  int count[4];
  int i, s, n, l;
  char *line, consensus, *number, *name, *column;

  name = (char *)emalloc(256*sizeof(char));
  name[0] = '\0';
  number = (char *)emalloc(24*sizeof(char));
  l = 0;
  while((line = tabGetLine(fp)) != NULL){
    l++;
    n = tabNfield();
    if(n < 5){
      printf("WARNING␣[sam2pro]:␣Skipping␣line␣%d␣with␣only␣%d␣fields.\n",l
         ,n);
      continue;
    }
    for(i=0;i<4;i++)
      count[i] = 0;
    if(strcmp(name,tabField(0)) != 0){
      name[0] = '\0';
      name = strdup2(tabField(0));
      printf(">%s\n",name);
    }
    consensus = tabField(2)[0];
    column = tabField(4);
    scanCol(column, dic, count, consensus, number);
    s = 0;
    for(i=0;i<4;i++)
      s += count[i];
    if(s >= args->m){
      printf("%s",tabField(1));
      for(i=0;i<4;i++)
        printf("\t%d",count[i]);
      printf("\n");
    }
  }
}

void scanCol(char *column, int *dic, int *count, char consensus, char *
    number){
  int i, j;
  char c;

  for(i=0;i<strlen(column);i++){
    c = column[i];
    if(c == '$')
      continue;
    else if(c == '^')
      i++;
    else if(c == '+' || c == '-'){
      number[0] = '\0';
      while(isdigit(column[++i]))
        number = strncat(number,column+i,1);
      for(j=0;j<atoi(number)-1;j++)
        i++;
    }else if(c == ',' || c == '.')
      count[(int)dic[(int)consensus]]++;
    else
      count[(int)dic[(int)c]]++;
  }
}
```

4 Change Log

- v. 0.1 (July 22, 2010)
  - First running version.
- v. 0.2 (July 22, 2010)
  - Fixed handling of patterns.
- v. 0.3 (April 23, 2013)
  - Now tests that input lines are properly formatted.

References

[1] B. Haubold, P. Pfaffelhuber, and M. Lynch. `mlRho`: A program for estimating the population mutation and recombination rates from shotgun-sequenced diploid genomes. *Molecular Ecology*, 19:277–284, 2010.