고시료 유전체 시퀀싱 자료의 처리와 품질관리, 친족 관계 추정

문형민[1,2], 조은민[3], 김수연[3], 강소영[3], 정충원[1,2*]

요약: 죽은 후 오랜 시간이 지난 생물의 조직인 고시료에 대한 차세대 염기서열 시퀀싱 기반 유전체 분석은 그 유용성 덕분에 진화유전학 분야에서 점차 확대되고 있는 연구 방식이다. 그러나 고유전체 자료의 특성을 고려하지 않은 채 일반적인 유전체 분석 방법을 적용하면 큰 편향이 나타날 수 있어 주의를 요한다. 본 논문에서는 고대인 대퇴골 시료 11점에서 얻은 실제 데이터 분석 사례를 통해 고시료 특이적인 유전체 분석 방법의 원리와 실제 분석 결과를 설명한다. 이를 이용하여 고유전체 시퀀싱 자료의 처리 방법, 품질 관리와 같은 전처리 과정, 고대인 간 친족 관계 추정법 등 기초 분석 방법에 대해 서술하여 고유전체 시퀀싱 자료에 대한 이해를 돕고자 한다.

키워드: 고유전체, 시퀀싱, 전장유전체

[1]*자연과학대학 생명과학부, 서울대학교, 서울 08826, 대한민국*

[2]*과학데이터혁신연구소, 서울대학교, 서울 08826, 대한민국*

[3]*국립문화유산연구원 보존과학연구실, 대전 34122, 대한민국*

*Corresponding author: cwjeong@snu.ac.kr

서론

차세대 염기서열 시퀀싱(Next Generation Sequencing, 이하 NGS)의 발달에 힘입어 죽은 후 오랜 시간이 지난 생물의 조직인 고시료에서 추출한 DNA (이하 고DNA)의 전장유전체 분석(Whole Genome Sequencing, 이하 WGS)이 성행하고 있다. 고DNA 연구는 고대인과 고인류 등 사람을 대상으로 한 연구들을 위주로 성장해 왔다(Green et al. 2010; Rasmussen et al. 2010; Rasmussen et al. 2011; Raghavan et al. 2014; Allentoft et al. 2015; Skoglund et al. 2017; Damgaard et al. 2018; McColl et al. 2018; Mittnik et al. 2018; Moreno-Mayar et al. 2018; Sikora et al. 2019; Jeong et al. 2020; Yang et al. 2020). 하지만 공통된 연구방법론을 통해 가축(Daly et al. 2018; Gaunitz et al. 2018; Verdugo et al. 2019; Kim et al. 2023), 작물(da Fonseca et al. 2015; Mascher et al. 2016), 야생생물(Miller et al. 2008; Bos et al. 2011; Palkopoulou et al. 2015; Soubrier et al. 2016; Wang et al. 2022) 및 미생물(Bos et al. 2014) 등 다양한 분류군에 대한 연구들이 진행되고 있다. 고DNA 연구는 유전적 시계열 자료를 제공해줌으로써 시간에 따른 집단의 진화를 추론의 대상에서 관찰의 대상으로 바꾸었다. 즉, 현대 집단의 DNA분석이 갖는 한계를 극복하여 지금은 사라진 과거 집단의 유전적 프로필과 자연선택에 의한 대립유전자의 빈도 변화를 직접 관찰할 수 있게 해 준다. 특히, 옛사람의 뼈로부터 얻은 DNA는 집단유전학 분야 외에도 고고학과 역사학에 새로운 정보를 제공해 줄 수 있어 더더욱 각광받고 있다. 최초의 고대인 전장유전체가 해독된 2010년(Green et al. 2010; Rasmussen et al. 2010) 이래 현재까지 약 1만 명의 고DNA NGS 자료가 출판되었으며(Mallick et al. 2024), 이 수는 시간이 지남에 따라 더욱 빠르게 증가할 것으로

기대된다.

고DNA의 WGS 분석은 기존 모계 혹은 부계 하플로그룹에 기반을 둔 방법론들을 단순히 확장한 것에 그치지 않으며, 두 가지 측면에서 독자적이고 대체 불가능한 장점을 지닌다. 첫 번째로, 하플로그룹 기반 분석은 집단 계통수를 정확하게 추정하는 데 있어 생물학적 한계를 갖는다. 미토콘드리아 또는 Y 염색체 DNA의 경우 재조합이 일어나지 않아 모든 좌위가 하나의 유전자 계통수(gene tree)를 따르게 된다. 그러나 종 혹은 집단의 계통 관계가 항상 유전자의 계통수와 일치하지는 않는다. 일례로, 고릴라-침팬지-사람의 경우 (고릴라, (침팬지, 사람))의 종 계통수를 따르는 유전자 계통수의 비율은 70%이며, 나머지 30%는 사람이나 침팬지를 고릴라와 단계통군으로 묶어준다(Scally et al. 2012). 이 현상을 집단유전학에서는 불완전 계보 구분(Incomplete Lineage Sorting, ILS)이라 칭하며, 특히 최근에 분화한 가까운 집단 사이에서 더욱 두드러지게 발생한다(Maddison 1997). 그러므로 모계 혹은 부계로만 전달되는 하플로그룹에 의존한 분석은 집단 구조에 대한 편향된 추론으로 이어질 우려가 있으며, 전장유전체를 통해 모든 자리의 유전자 계통수를 고려해야만 정확한 집단 분화의 역사를 추정할 수 있다. 더 나아가 집단의 경우에는 생식적 격리가 일어나지 않은 경우가 대다수이므로 유전자 흐름이 흔히 발생하여 집단의 계통 관계가 단순한 계통수로 나타낼 수 있는 것보다 복잡한 경우가 많다. 두 번째로 하플로그룹 분석은 집단 간 혼합과 크기 변화를 감지하기 어렵다. 하플로그룹 기반 분석은 오직 하나의 유전자 계통수에 기반하므로 두 집단 간 혼합이 일어나더라도 혼합의 정확한 특징을 잡아내기에는 어렵다. 그러나 전장유전체는 재조합을 통해 집단 간 DNA가 섞이게 되므로 집단 간 혼합을 검정하거나 (Patterson et al. 2012), 혼합 시기(Loh et al. 2013; Hellenthal et al. 2014; Chintalapati et al. 2022) 및 과거 집단 크기(Li and Durbin 2011; Schiffels and Durbin 2014)를 추정하는 작업들이 가능하다. 이런 이점들 덕분에 짧은 시간대를 다루는 집단을 분석하는 데에는 전장유전체 분석이 필수적이다.

하지만 고DNA WGS 자료의 경우 신선한 조직에서 추출한 일반 DNA의 WGS 자료 분석과 다른 특성을 갖고 있다. DNA의 사후 변성 패턴, 짧은 DNA 길이, 시퀀싱 오류, 비교적 낮은 수율과 외부 DNA 오염은 고 DNA 분석에 영향을 미치는 요소들이며, 이를 고려하지 않았을 시 분석에 편향(bias)이 발생할 수 있다(Prüfer et al. 2010). 그렇기 때문에 고DNA의 WGS 방법론에는 일부 변형 사항들이 존재한다. 이런 고려 사항들은 시료 채취 부위에 따라서는 크게 영향을 받지 않으므로, 고 DNA의 특성을 고려한 자료 분석 파이프라인이 잘 설립되면 다양한 종류의 고DNA 시료에 대한 일괄적인 분석이 가능해진다. 따라서 성공적인 고DNA 분석을 위해서는 고DNA 특이적인 변이에 대한 이해가 선행되어야 한다.

본 리뷰에서는 몽골 알타이 시베트 하이르한(Shiveet Khairkhan) 유적에서 국립문화유산연구원(구 국립문화재연구소) 주도의 한-몽 연합 고고학 발굴단이 출토한 철기시대 고대인들의 대퇴골 11점으로부터 얻은 시료를 바탕으로 DNA를 추출, 시료별로 3개의 라이브러리를 제작하여 WGS 후 고DNA 분석에 통용되는 표준 방법론에 따라 따라 유닉스 계열의 운영체제(Unix, Linux) 기반 컴퓨터를 이용한 자료의 전처리와 분석을 진행하였다. 기존에 잘 정립된 고DNA 분석 파이프라인을 실제 데이터에 적용하여 기본적인 분석을 진행하는 일련의 과정들을 재현함으로써 고DNA 분석에서 중요하게 다뤄져야 하는 고려 사항들에 대한 안내와 결과의 예시를 제공하고자 한다. 또한, 부록에 실제 사용한 코드를 작업 순서에 따라 제시하였다.

본론

시료 준비

시료의 준비는 대전 소재 국립문화유산연구원 보존과학연구실의 고DNA 전용 청정실험실에서 진행되었다. 몽골 시베트 하이르한에서 출토된 고대인 유골 11구의 대퇴골 시료로부터 1x3 cm 크기를 다이아몬드 디스크로 절단한 후, 전동 드릴에 결착한 버(bur)를 이용해 표면 오염물질을 제거 후 5 - 7% Sodium hypochlorite 수용액을 처리하여 표면 잔존 미생물과 핵산 오염물을 제거하였다. 세척된 시료는 무균 작업대에서 24 시간 건조 후, 260 nm 파장의 자외선을 15분간 조사하였다. 시료는 이후 동결분쇄기(Freezer Mill 6770, SPEX SamplePrep, USA)를 이용하여 골분을 시료별로 획득하였으며, 실험까지 4℃에서 보관되었다.

DNA 추출, 라이브러리 제작 및 시퀀싱

DNA 추출과 라이브러리 제작은 골분 50 mg을 이용하여 해외 청정실험실 2 개소(일본 가나자와 대학, 미국 시카고대학)에서 독립적으로 진행되었다. 일본 가나자와 대학에서는 고DNA 전용 청정실험실에서 출판된 프로토콜(Rohland et al. 2018)에 따른 DNA 추출 1회, NEBNext® Ultra™ II library prep kit를 이용한 시퀀싱 라이브러리 제작 2회를 진행하였으며, 미국 시카고 대학교에서는 고DNA 전용 청정실험실에서 출판된 프로토콜(Dabney and Meyer 2019)에 따른 DNA 추출 1회, 라이브러리 제작 1회를 진행하였다. 일본 가나자와 대학에서는 1회 DNA 추출 후 라이브러리 제작 1회에 한해 우라실 염기와 당인산 사이의 N-glycosylic 결합을 가수분해하여 우라실 염기를 제거하는 Uracil-DNA glycosylase (이하 UDG) 처리를 하여 DNA 사후 변성을 제거한 라이브러리를 제작하였다. 미국 시카고 대학교에서는 기 출판된 프로토콜(Meyer and Kircher 2010)에 따라 DNA의 5’과 3’ 말단에 사후 변성을 일부 남기는 partial-UDG 처리를 진행한 후 라이브러리를 제작하였다.

모든 라이브러리들은 몰농도를 기반으로 라이브러리 별로 동일한 양을 혼합한 후, 가나자와 대학교에서 제작된 라이브러리들은 ㈜마크로젠에 Hiseq X paired-end (PE) 150 base-pair (bp) 시퀀싱을 의뢰하였으며, 시카고 대학교에서 제작된 라이브러리들은 대학교의 Genomics Facility에 NovaSeq 6000 SP 플랫폼에서 PE 150 bp 시퀀싱을 의뢰하였다. 일루미나 라이브러리의 기본 형태와 일루미나 시퀀싱의 기본 원리는 (그림 1)에 도식화하였다.

고DNA 시퀀싱 리드 내 어댑터(adapter) 서열 제거

고DNA의 경우 파편화되어 있는 경우가 대부분이며, 라이브러리에 포함된 고DNA 분자 길이의 중앙값이 100 bp를 넘지 못하는 경우가 많다(Sawyer et al. 2012; Dabney et al. 2013). 일루미나 시퀀싱 장비의 경우 시퀀싱 리드 길이가 100-250 bp 범위임을 고려했을 때 짧은 분자 길이를 지니고 있으며, 실제로 고DNA 분자 전체를 시퀀싱한 뒤 바코드 및 어댑터 서열까지 시퀀싱되는 경우가 대부분이다(그림 2). 바코드와 어댑터 서열이 3’ 끝에 포함된 리드를 그대로 참조유전체 서열에 매핑할 경우 리드 매핑 정확도 및 효율이 크게 감소하기 때문에 매핑 전에 바코드와 어댑터 서열을 리드의 3’ 부분에서 제거해야 한다. 또한 그 과정에서 PE 시퀀싱 자료의 경우 어댑터 서열 제거 후 forward 리드(R1)과 reverse 리드(R2)를 하나의 리드로 합치는 Collapsing 과정을 진행해준다(그림 2).

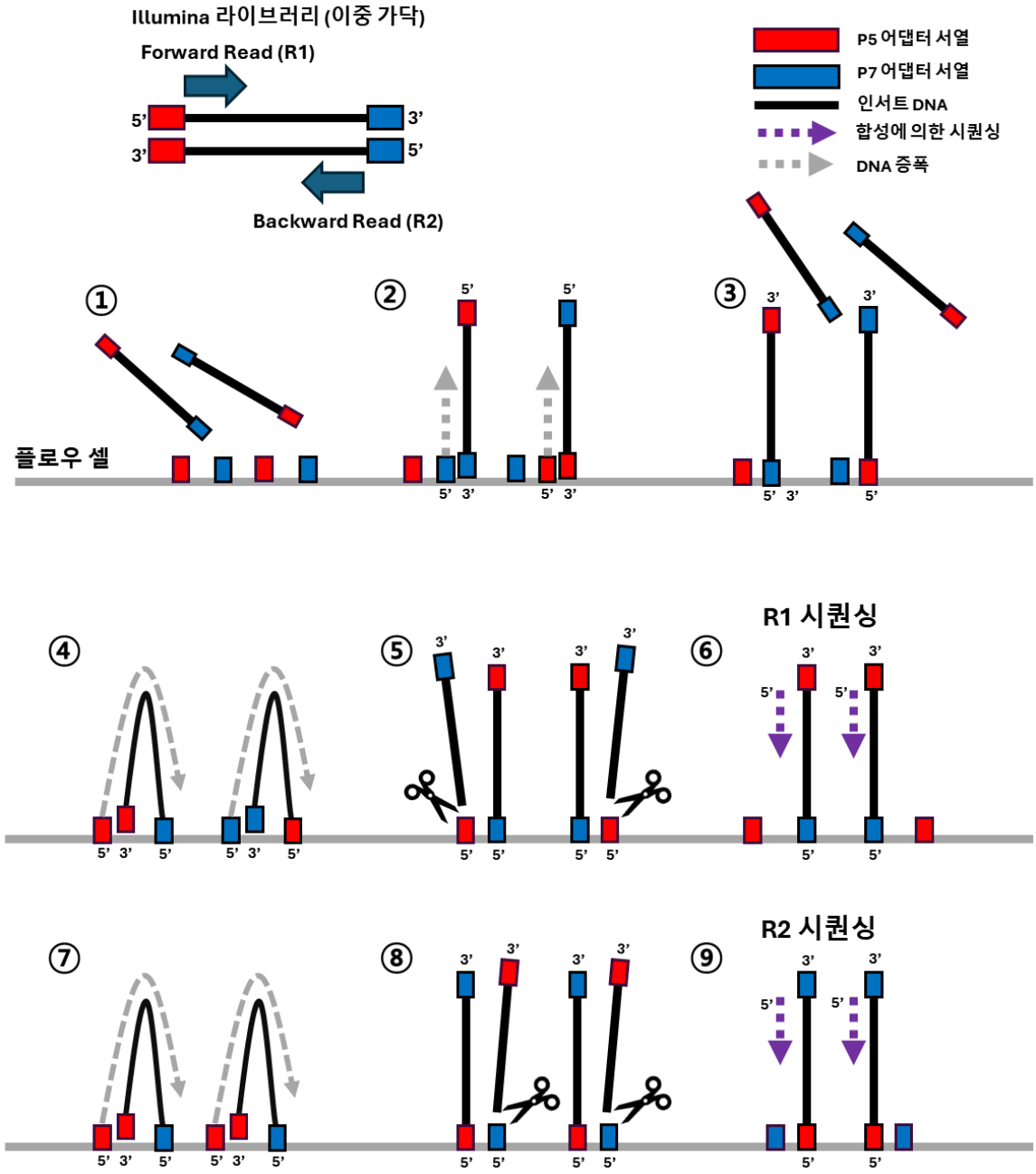

그림 1. 일루미나 라이브러리와 시퀀싱의 원리. 일루미나 라이브러리의 기본적인 형태와 시퀀싱되는 과정을 단계별로 나타낸 도식도. PE 시퀀싱의 경우 ① - ⑨ 과정이, SE 시퀀싱의 경우 ① - ⑥ 과정까지 진행된다. ① 라이브러리는 단일 가닥으로 분리되어 플로우 셀의 어댑터에 부착된다. ② 플로우 셀에 부착된 라이브러리에 상보적인 가닥이 합성된다. ③ 원래 라이브러리 가닥이 분리된다. ④ Bridge amplification을 통해 라이브러리 시퀀스가 대량으로 복제된다. ⑤ P5 어댑터 서열이 플로우 셀에 부착된 가닥들이 제거된다. ⑥ R1 리드 시퀀싱이 진행된다. ⑦ 한 번의Bridge amplification으로 플로우 셀에 부착된 가닥에 상보적인 가닥이 생성된다. ⑧ P7 어댑터 서열이 플로우 셀에 부착된 가닥들이 제거된다. ⑨ R2 리드 시퀀싱이 진행된다.

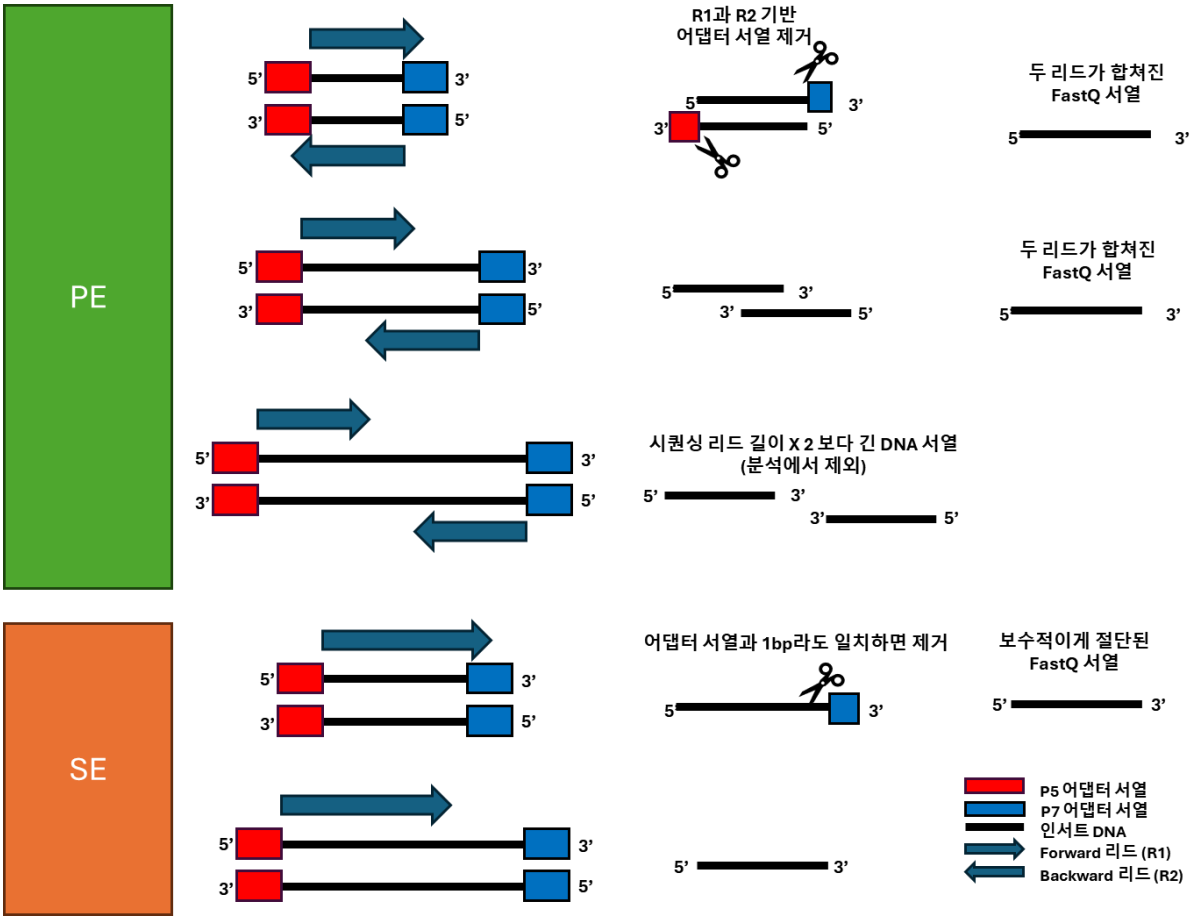

그림 2. 시퀀싱 방법론에 따른 어댑터 제거 방식의 도식도. PE 시퀀싱과 SE 시퀀싱에 따른 어댑터 제거 전략을 도식도로 나타내었다. 가위는 어댑터를 제거하는 위치를 표시하였다. 인서트 DNA의 길이와 시퀀싱 방법에 따라 어댑터 제거 및 리드 병합의 전략이 다르게 나타난다.

본 리뷰에서는 AdapterRemoval v2.3.0 (Schubert et al. 2016)을 사용하여 리드 내 어댑터 서열을 제거하였고, 고DNA 리드의 짧은 길이에 맞게 다음과 같은 옵션들을 적용해주었다.

1) --trimns --trimqualities --minquality 20: Base quality가 20보다 낮은, 5’ 와 3’ 영역에 연속적으로 이어진 염기서열 혹은 N (염기가 결정되지 않음)이 이어질 경우 그 서열을 제거해준다. 예시로 two-channel 시퀀싱 시스템의 경우 네 개의 염기를 빨강과 초록 형광신호 2개의 유무 조합으로 구분하는데, 짧은 리드의 시퀀싱이 전부 끝난 후 반응이 진행되지 않아 형광신호가 나오지 않는 것을 전부 G로 부르게 되어 3’ 에 긴 G 서열이 이어질 수 있으며, 이를 제거하지 않으면 참조유전체 서열에 특정 염기가 길게 나열된 구간에 리드들이 잘못 매핑되는 경우가 발생할 수 있다. 이런 영역의 제거를 통해 매핑률을 향상시킬 수 있으나, 낮은 복잡도의 라이브러리에서는 PCR로 인해 생긴 복제된 리드들을 인공적으로 다른 리드로 인식되도록 만드는 부작용이 생길 수 있으므로 주의해야 한다.
2) --adapter1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC
--adapter2 AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTA: Illumina 라이브러리의 Truseq 리드 각각 2와 1의 서열을 제공해주어 준다. 다른 어댑터를 사용하는 경우 그에 맞춰서 시퀀스를 바꾸어 제공해주어야 한다.
3) --minadapteroverlap 1: 본 연구에서 사용된 라이브러리는 삽입된 DNA 조각의 양쪽 끝에서 리드를 한 번씩 읽어 R1과 R2 리드를 생산하는 PE 시퀀싱을 진행하였다. PE 시퀀싱의 경우

두 서열의 겹치는 영역을 대조함으로써 어댑터 서열을 쉽게 찾아줄 수 있다(그림 2). 그러나 만약 single-end (SE) 시퀀싱을 진행하게 되는 경우, PE 시퀀싱과 다르게 어디까지가 정확히 어댑터 서열인지를 확인하는 것이 쉽지 않다. 해당 옵션은 SE 라이브러리에 대해 보수적으로 주어진 어댑터 서열과 1 bp라도 일치하는 경우, 3' 끝을 제거해준다.

4) --minlength 35: 35 bp 이하의 박테리아 유전체가 사람 유전체에 잘못 매핑 될 가능성이 높다는 연구결과(Meyer et al. 2016)와 짧은 리드의 경우 사람 참조유전체 서열에 확정적으로 매핑되기 어려운 점들을 고려하여 35 bp 보다 짧은 리드들을 제거해준다.
5) --collapse: PE 시퀀싱 리드의 경우 11개 이상의 염기서열이 겹치는 R1과 R2 리드를 하나의 리드로 병합해주었다.

사람 참조유전체로의 리드 매핑과 필터링

고DNA의 사람 참조유전체로의 매핑을 할 때에는 매핑 프로그램에 의한 편향을 방지하기 위해 일관적으로 Burrows-Wheeler Aligner backtrack 프로그램(aln + samse/sampe) (Li and Durbin 2009)을 사용한다. 본 연구에서 사용된 프로그램은 BWA aln/samse v0.1.17이었고 사람 참조유전체 서열은 hs37d5이었다. BWA aln의 경우 두 가지 옵션을 주었다. 매핑된 리드들은 bam 파일의 형식이 되었으며, samtools 프로그램(Li et al. 2009)을 사용하여 다루었다.

1) -n 0.01: -n은 허용 가능한 maximum edit distance를 수정하는 방법으로, 리드와 참조유전체 서열 간 차이나는 염기 서열의 정도(점 변이, 삽입, 결실)의 상한을 설정해준다. 값이 정수일 경우 그 값만큼을 절대적인 상한으로 설정하며, 0과 1 사이의 값을 줄 경우 리드의 길이에 비례하여 2%의 시퀀싱 오류를 가정했을 때 최대 해당 비율의 리드들이 매핑되지 않도록 상한을 결정한다. 본 분석에서는 BWA aln의 기본값인 0.04보다 낮은 0.01로 값을 주어 고DNA의 사후 변성과 낮은 시퀀싱 퀄리티를 반영, 약한 기준을 설정해주었다.
2) -l 32 (UDG 처리시)/9999 (UDG 미처리시): -l 옵션은 매핑의 속도를 올리기 위해 -l에 5' 으로부터 주어진 길이만큼의 서열을 "시드(Seed)"로 사용하여 시드와 참조 유전체가 정확히 일치되는 서열을 찾은 후 리드와 참조 유전체를 정렬한다. 리드 길이보다 큰 값을 줄 경우 시드를 무시하고 리드 전체를 이용해 매핑을 진행한다. 신선한 DNA를 시퀀싱한 경우 리드의 5' 부분에서 염기서열의 정확도가 가장 높지만, UDG 처리를 하지 않은 고DNA 라이브러리의 경우 아래에서 설명할 DNA 사후변성이 리드의 양 끝 부분에 집중적으로 나타나기 때문에 리드의 5' 부분의 염기서열 정확도가 낮아 "시드"의 이점을 살리지 못한다. 따라서 UDG 처리 없이 제작된 고DNA 라이브러리의 경우 리드의 길이보다 긴 시드 값(9999)을 주어 시드 기능을 해제하였다.

매핑 이후, 라이브러리 제작 또는 이후 과정에서의 중합효소연쇄반응(polymerase chain reaction; 이하 PCR)으로 인해 복제된 리드들을 제거해주는 과정을 진행하였다. 고DNA의 경우 어댑터 서열 제거 후 짧은PE 시퀀싱 된 리드들은 하나의 리드로 합쳐지게 되며, 어댑터 서열이 제대로 제거되었다면 5' 과 3' 의 위치가 동일한 두 리드는 동일한 분자로부터 유래한 PCR 산물로 인한 복제된 리드일 수 있기 때문에 보수적으로 판단하여 제거하였다. 본 연구에서는 EAGER 파이프라인의 DeDup v0.12 프로그램 (Peltzer et al. 2016)을 이용하여 동일한 분자가 두 차례 이상 시퀀싱된 복제서열들을 제거해주었다. SE 데이터의 경우 어댑터가 완전히 똑같이 제거되지 않는 경우들이 발생하여 동일

분자로 왔음에도 3’ 끝이 일치하지 않을 수 있으므로, 보수적이게 5’ 위치만 같더라도 리드를 제거해준다. PE와 SE 자료 모두 복제서열들 중 염기서열 품질 점수(base quality score)의 합이 가장 높은 리드를 남기고 나머지를 제거하였다. PCR 복제 산물을 제거한 리드들은 참조유전체에 고유하게 매핑된 수준을 평가하는 MAPQ 점수가 30 이상인 리드들만을 samtools view -q30 옵션을 사용하여 필터링하였으며 이후 분석에 사용하였다. MAPQ 점수 30은 리드가 현재 매핑된 위치에서 실제 왔을 가능도(likelihood)가 유전체 내에서 다음으로 좋은 위치에서 왔을 가능도보다 1000배 높다는 것을 의미한다.

DNA 사후변성 확인

고DNA의 경우 사후변성(post-mortem damage, PMD)이 일어나 DNA 분자가 화학적 변화를 겪게 된다 (Sawyer et al. 2012; Dabney et al. 2013). 이런 변화들 중 가장 특징적인 것은 사이토신 염기가 우라실로 탈아미노화(deamination)되는 현상이다. 탈아미노화 현상은 짧게 조각난 DNA 분자에서 이중가닥 중 한 쪽만 남아 노출되어 있는 single-strand overhang 영역에 가장 많이 일어나게 되며, 시퀀싱 시 사이토신 염기가 티민으로 읽히게 되는 현상을 일으킨다(C-to-T 치환). Single-strand overhang이 나타날 확률은 DNA 분자의 5’ 또는 3’ 말단으로 갈수록 지수적으로 증가한다. 주요하게 사용되는 프로그램인 mapDamage (Jónsson et al. 2013)는 리드와 표준유전체 간 염기 서열 일치도 양상을 바탕으로 탈아미노화 현상을 그래프와 수치로 정량화해준다.

사이토신에서 우라실로의 염기 변화는 고DNA의 라이브러리 제작 방식에 따라 리드 내에 다양한 형태의 염기 오편입(misincorporation)을 일으킬 수 있다. 라이브러리는 제작 방식과 UDG 효소 처리 방식에 따라 몇 가지로 분류가 가능하다. 제작 방식에 따라서는 크게 double-strand DNA 라이브러리와 single-strand DNA 라이브러리 방식이 있으며, DNA 사후 변성된 우라실 염기를 제거하는 UDG 효소의 처리 정도에 따른 분류로는 하나도 처리하지 않은 non-UDG, 부분적으로 처리한 UDG-half (partial-UDG) 라이브러리와 full-UDG 라이브러리로 나뉘어진다. 마지막으로 본 연구에서 가나자와 대학교에서 사용된 라이브러리 키트인 NEBNext® Ultra™ II library prep kit로 제작한 UDG 처리하지 않은 라이브러리의 경우 double-strand DNA 라이브러리에 해당하나 표준적인 double-strand DNA 라이브러리 제작 방식과 차이가 있어 따로 명시하였다. 기본적으로 고DNA의 사이토신 염기의 탈아미노화는 시퀀싱 결과에 크게 두 가지 형태의 영향을 미친다. 그대로 시퀀싱될 경우 원래 염기가 사이토신인 자리가 티민으로 읽히게 되는 경우(C-to-T), 그리고 5’ overhang이 상보적인 염기로 채워진 후 시퀀싱 되어 원래 염기가 구아닌인 자리가 우라실에 상보적인 아데닌으로 읽히게 되는 경우(G-to-A)가 있다. Double-strand 라이브러리의 경우 5’말단의 overhang은 DNA polymerase로 채워지고 3’ 말단의 overhang은 exonuclease로 잘려 나가게 되어 5’ 말단에 높은 빈도의 C-to-T 치환이, 3’ 말단에 높은 빈도의 G-to-A 치환이 관찰이 된다. 예외적으로 NEBNext® Ultra™ II 라이브러리 키트의 경우 5’말단의 overhang이 이미 상보적인 염기로 채워진 이후, 어댑터를 라이브러리에 붙인 후 USER 효소(UDG와 Endonuclease VIII의 기능을 가진 효소)를 처리하는 과정과 PCR에서 Q5 효소가 우라실 염기를 만나면 중합 과정이 멈추는 효과로 인해 5’말단의 C-to-T 치환이 두 세 염기에서만 나타나는 특이적인 패턴이 관찰된다. 만약 라이브러리 제작 전에 추출물에 대해 UDG 처리를 하는 경우, C-to-T 치환이 DNA 라이브러리 말단의 두 세 염기 정도에서만 한정적으로 나타나는 패턴이 관찰된다. Single-strand 라이브러리의 경우 DNA를 구성하는 두 서열이 분리된 후 각각이 다른 라이브러리로 시퀀싱이 되므로 G-to-A 치환이 시퀀싱되는 일이 없어

5’ 말단과 3’ 말단 모두 C-to-T 치환만이 관찰된다. (그림3)은 사후 변성 패턴의 종류와 라이브러리 준비 과정에서 발생한 원인들을 도식화한 것이다. 이런 탈아미노화로 인한 염기 치환의 양상을 확인함으로써 NGS 자료가 실제 고DNA 분자를 시퀀싱한 결과가 맞는지 여부를 검정하는데 사용된다. 또한 치환의 비율이 예상되는 것보다 낮게 나타남을 관찰함으로써 시료의 현대 DNA오염 여부를 추정할 수도 있다. 본 연구에서는 mapDamage v2.0.9를 사용하여 사후변성의 양상을 확인, 실제 고DNA를 시퀀싱 했음을 입증하였다(그림 4).

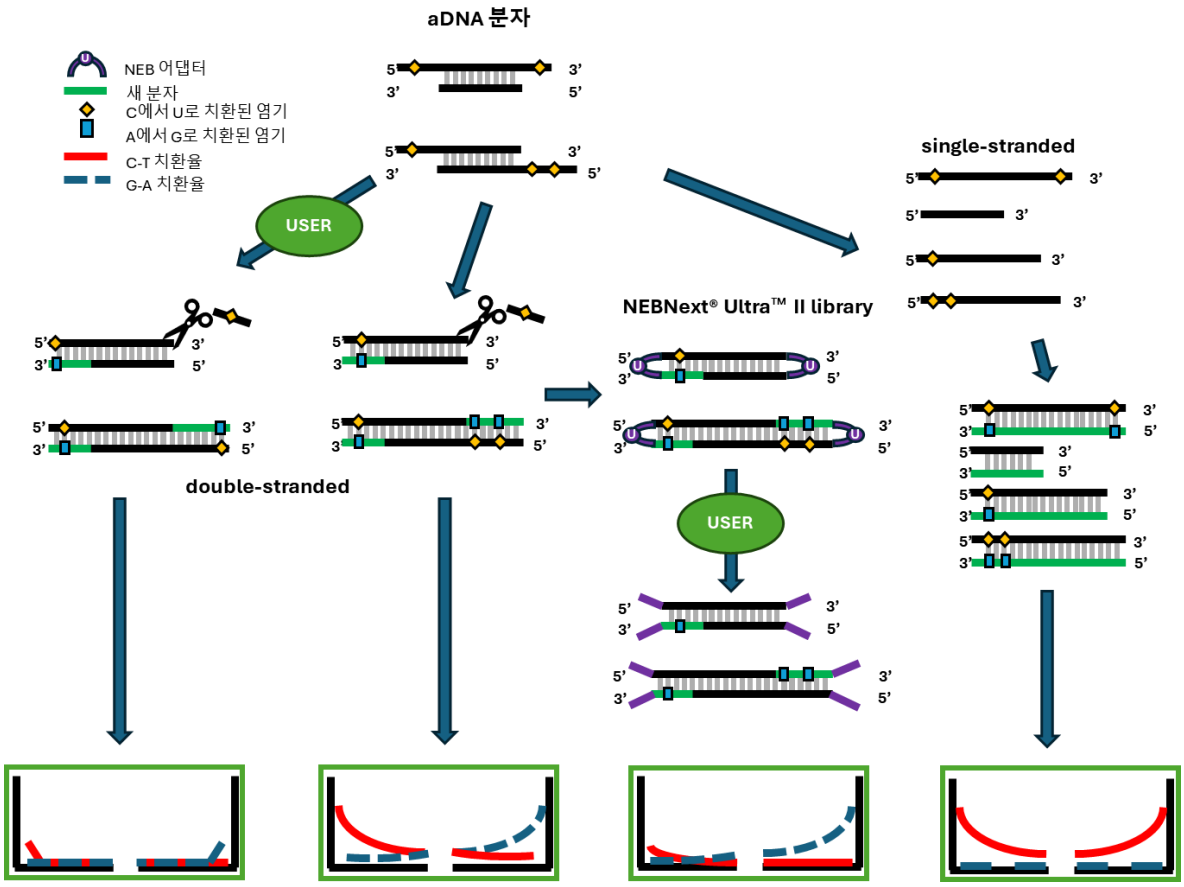

그림 3. 라이브러리 종류에 따른 다른 mapDamage 그림 패턴과 원리. 왼쪽에서 오른쪽까지 차례대로 partial-UDG 처리를 한 라이브러리, double-strand non-UDG 라이브러리, NEBNext® Ultra™ II 라이브러리 키트로 만들어진 non-UDG 라이브러리, 그리고 single-strand 라이브러리의 제작 방식에 따라 나타나는 다른 사후 변성 패턴의 종류를 나타낸 것이다. 녹색 네모 내의 그림은 mapDamage로 해당 라이브러리를 분석했을 때 나타나는 사후 변성 패턴을 나타낸 것으로, 좌측의 곡선은 리드의 5’ 영역에서의 염기 치환율을, 우측의 곡선은 리드의 3’ 영역에서 나타나는 염기 치환율을 나타낸다.

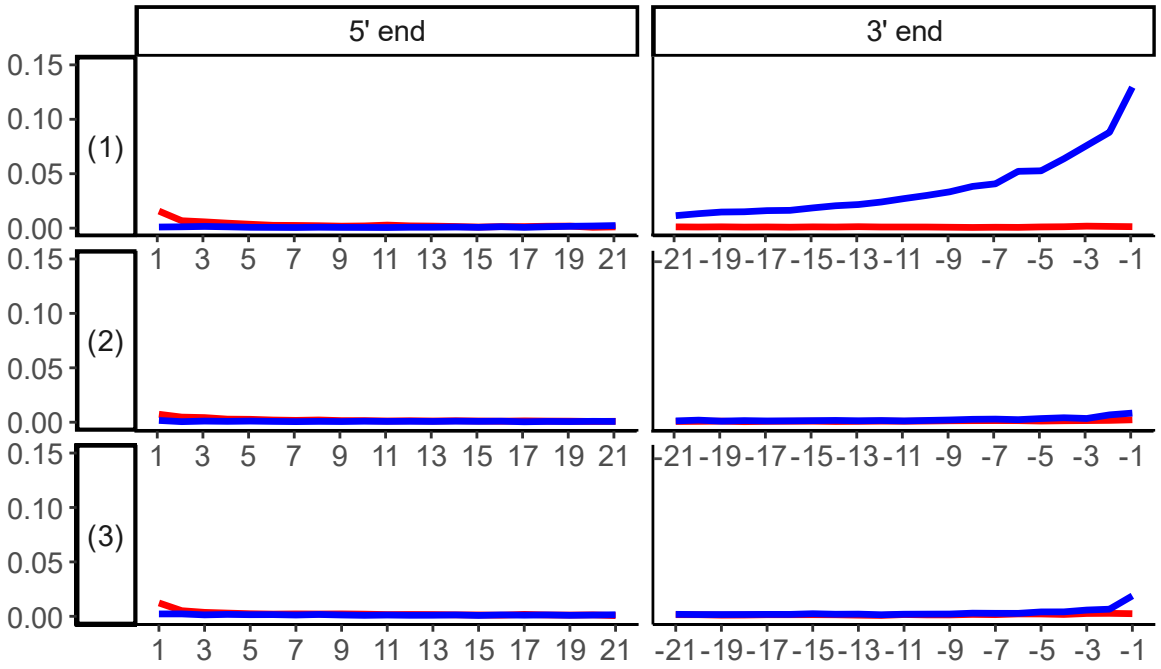

그림 4. MAB097 개체로부터 제작된 라이브러리에 따른 mapDamage 패턴의 비교. (1) non-UDG 라이브러리 (2) Full-UDG 라이브러리 (3) Partial-UDG 라이브러리의 mapDamage 그림. 붉은 실선은 참조유전체 기준 C->T 치환의 비율을, 푸른 실선은 참조유전체 기준 G->A 치환의 비율이다. 세 패널은 라이브러리의 준비 방식에 따른 사후 변성 패턴의 mapDamage 그림을 의미한다. 왼쪽의 곡선들은 리드의 5’ 영역에서의 염기 치환율을, 우측의 곡선들은 3’ 영역에서 나타나는 염기 치환율을 나타낸다.

1240K SNP 패널 유전자형 결정

NGS 자료로부터 집단유전학 분석을 하기 위해 다수의 전장유전체 변이 목록에 대하여 고대인별로 유전자형(genotype)을 결정지어야 한다. 사람 고유전체학에서 통상적으로 사용하는 변이 목록은 1,233,013 개의 대립유전자가 2개인(bi-allelic) 단일염기다형성(single nucleotide polymorphism; SNP)을 포함하는 통칭 “1240K” 패널이다(Fu et al. 2015; Haak et al. 2015; Mathieson et al. 2015). 그렇기 때문에 우리가 매핑한 리드로부터 1240K 유전자형을 결정해줄 필요가 있다.

고DNA 시료의 경우 매우 낮은 커버리지(1x 또는 그보다 아래의 리드 커버리지)의 데이터를 얻을 가능성이 높다. 낮은 커버리지의 NGS 데이터로부터 유전자형을 결정하는 경우, 유전자형 결정 과정에서 참조유전체의 유전자형이 대립 유전자형에 비해 더 선호되는 참조유전체 편향(reference bias)이 발생하며(Prüfer et al. 2010; Han et al. 2013), 참조유전체의 유전자형을 많이 갖는 리드들이 그렇지 않은 리드들에 비해 매핑이 잘 되는 현상부터 유전자형을 결정하는 통계적 모델들이 참조유전체 유전자형을 동형접합으로 갖는 경우에 유리하게 유전자형을 결정하는 등의 다양한 층위에서 문제가 발생한다. 참조유전체 편향이 발생하는 경우, 참조 유전체에 가까운 두 개체가 집단 구조와는 관계없이 유연관계를 갖는 것처럼 잘못된 결론에 도달할 위험성이 있다. 이를 극복하기 위해 낮은 커버리지의 데이터에서는 유전자형을 관찰한 리드들 중 랜덤하게 하나를 골라 반수체처럼 유전자형을 결정하는 pseudohaploid 방식이 선호된다. 이를 위해 먼저 samtools mpileup -R -B -q30 -Q30 -l [1240K SNP 자리 정보] 로 pileup 형식의 파일을 만들어주고, pileup 파일로부터 유전자형을 결정하는 pileupCaller 프로그램이 주로 사용이 되며, --randomHaploid 옵션을 주어 pseudo-haploid 유전자형을 결정지을 수 있다. samtools mpileup의 -B 옵션은 mpileup의 기본 옵션인 alignment 점수를 보정해주는 기능을 꺼주며, 이를 끄지 않을 경우 참조 유전체에 가까운 리드들에 유리한 편향이 발생할 수 있으므로 해당 옵션을 꼭 추가해주어야 한다.

고DNA로부터의 유전자형 결정을 할 때에는 또한 앞서 언급한 사후 변성으로 인한 염기 치환을 고려해주어야 한다. 기본적인 전략은 사후변성의 영향을 받았을 염기들에 해당하는 SNP들을 무시하는 방법으로 이루어진다. 그러므로 유전자형의 결정과정은 사후 변성의 양상, 즉 라이브러리를 어떻게 만들었는지에 따라 달라진다. 앞서 DNA 사후 변성 확인 섹션에서 크게 네 가지의 사후 변성 패턴에 대해 언급한 바가 있는데, 표준적인double-strand non-UDG 라이브러리, NEBNext® Ultra™ II 라이브러리 키트로 만들어진 non-UDG 라이브러리, single-strand non-UDG 라이브러리와 partial-UDG 처리한 라이브러리가 그것이다. 네 라이브러리의 사후 변성에 따라 다음과 같은 전략들을 사용할 수 있다(그림 5).

1) Partial UDG 처리를 한 라이브러리의 경우 사후 변성이 5’과 3’ 말단의 일부 염기들에만 국한되어 있기 때문에, 단순히 양 끝을 마스킹(masking)해줌으로서 C/T 와 G/A SNP 들을 쉽게 결정해 줄 수 있다. 대략 1 - 3 bp를 trimBam 프로그램으로 마스킹을 하여 유전자형을 결정해준다.
2) Double-strand non-UDG 라이브러리의 경우 리드에 관계없이5’에 C-to-T 변이와 3’에 G-to-A 변이가 높은 빈도로 존재하므로 보수적이게 C/T와 G/A SNP를 모두 무시하고 Transversion인 SNP만 사용하는 방법이 있다. 또는 리드의 5’과 3’염기 말단을 일괄적으로 5 - 10 bp 정도 마스킹하는 방법도 존재하며, bamUtils의 trimBam 모듈을 통해 쉽게 진행할 수 있다. 전자의 경우 많은 유전적 정보를 잃게 되는 단점이 있으며, 후자의 경우 1 - 2% 정도의 남은 사후변성까지는 제거할 수 없는 단점이 있다.

3) NEBNext® Ultra™ II 라이브러리 키트로 만들어진 non-UDG 라이브러리의 경우 특이적인 사후변성 패턴을 잘 활용해 많은 자리에 대해 유전자형을 결정할 수 있다. 참조유전체와 동일한 가닥의 DNA를 시퀀싱한 양성 서열(positive strand) 리드들의 경우 표준유전체 기준 G-to-A 사후변성을 3’ 말단에 높은 비율로 가지고 C-to-T 사후변성을 5’말단 첫번째와 두번째 염기 정도에 갖게 되므로, 5’ 말단을 trimBam으로 마스킹한 후 C/T SNP를 결정지을 수 있다. 반대로 참조유전체와 상보적인 가닥의 DNA를 시퀀싱한 음성 서열(negative strand) 리드의 경우 표준유전체 기준으로 G-to-A 변환에 상보적인 C-to-T 변환을 갖고 있으므로, 반대로 5’ 말단을 마스킹한 후 G/A SNP를 결정지을 수 있다. C/T와 G/A SNP을 제외한 나머지 SNP들은 모든 리드들을 이용하여 마스킹하지 않고 유전자형을 결정한다. 5’ 말단 5 bp 마스킹을 위해서는 trimBam 프로그램의 “-L 5” 옵션을 사용하며, 양성 서열을 추출하기 위해서는 samtools view -F 16, 음성 서열을 추출하기 위해서는 samtools view -f 16을 사용할 수 있다.

4) Single-strand 라이브러리의 경우 모든 리드가 C-to-T 사후변성 패턴을 지니고 있다. 그러므로 양성 서열은 C/T SNP들에 영향을 받고 음성 서열들은 G/A SNP들에 영향을 받는다. NEBNext® Ultra™ II 라이브러리와 비슷하게 양성 서열과 음성 서열을 나누어 준 후, 양성 서열로부터는 G/A SNP들을, 음성 서열로부터는 C/T SNP들을 결정하고 나머지 SNP들은 전체 리드들로부터 결정하는 방법을 통해 사후변성의 영향을 피할 수 있다. pileupCaller에서는 --singleStrandMode라는 옵션을 통해 별도로 리드를 나누지 않고도 이런 전략으로 유전자형을 결정할 수 있다.

다음 과정으로 만들어지는 유전자형에 대한 정보를 담고 있는 파일은 eigenstrat 형식이라 불리며, 세 개의 파일이 하나의 묶음을 이루게 된다. SNP의 염색체 상 위치와 대립유전자 정보를 담고 있는 SNP 파일은 .snp 확장자를 가지며, 하나의 행이 하나의 SNP 좌위에 대응된다. 개체의 정보를 담고 있는 IND 파일은 .ind 확장자를 가지며, 개체의 이름, 성별과 집단에 대한 정보를 담으며, 하나의 행이 하나의 개체에 대응된다. 개체가 지니는 참조유전체의 유전자형 개수 정보를 행렬로 나타낸 GENO 파일은 .geno 확장자를 가지며, 행은 하나의 SNP 좌위에 대한 여러 개체들의 유전자형을, 열은 하나의 개체의 모든 SNP에 대한 유전자형을 나타낸다. 본 연구에서 사용한 pseudohaploid 유전자형의 경우 이형접합 좌위를 가질 수 없으므로 하나의 SNP 좌위에 대해 GENO 파일 내에서는 참조유전체가 가지는 유전자형을 동형접합으로 가지는 2, 참조유전체의 유전자형을 갖지 않는 0, 또는 결측 값을 의미하는 9의 세가지 값 중 하나만을 갖게 되며, 추후 분석에서도 이형접합 좌위가 존재하지 않음을 유념하고 분석을 진행해야 한다. SNP, IND, 그리고 GENO 세 파일의 자세한 형식과 관계는 (그림 6)에 자세히 도식화하였다.

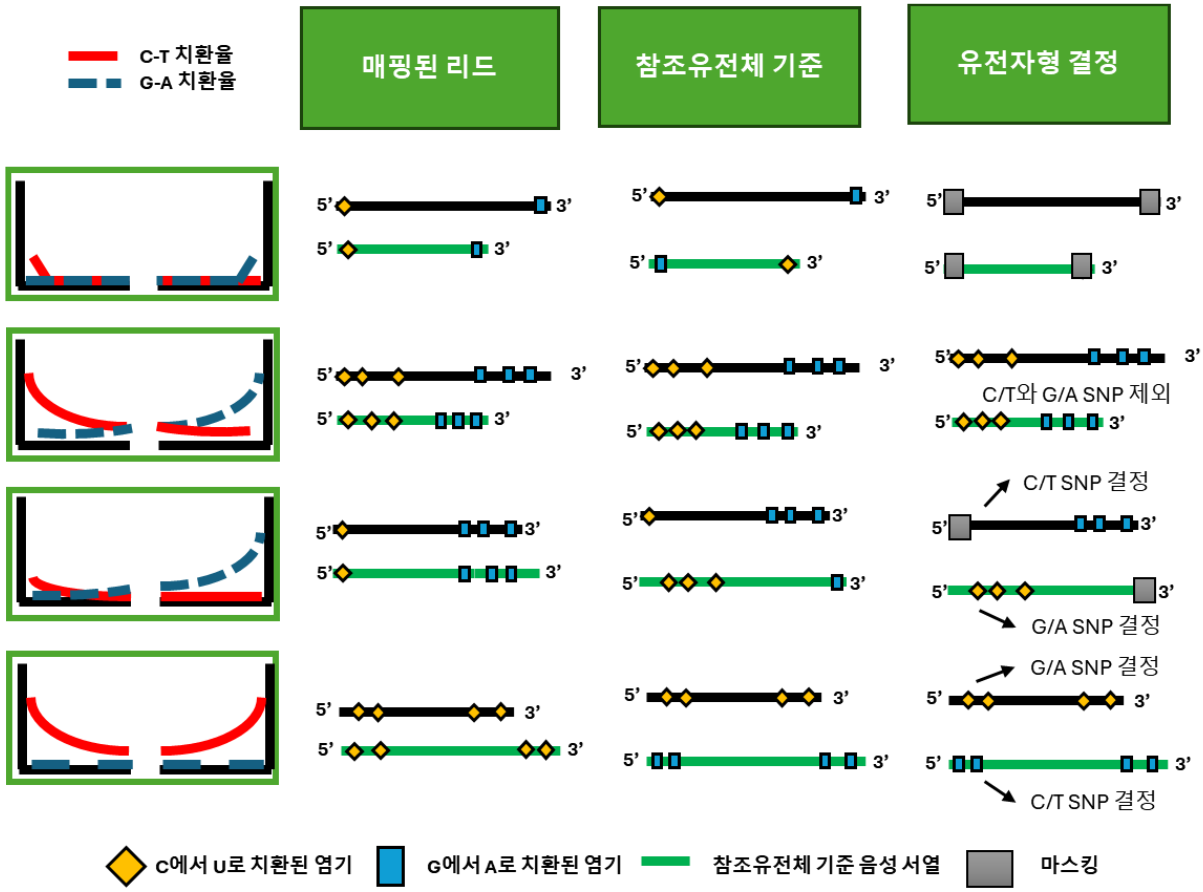

그림 5. 라이브러리 사후 변성 패턴에 따른 유전자형 결정 과정 도식화. 위에서부터 아래까지 차례대로 partial-UDG 처리를 한 라이브러리, double-strand non-UDG 라이브러리, NEBNext® Ultra™ II 라이브러리 키트로 만들어진 non-UDG 라이브러리, 그리고 single-strand 라이브러리의 유전자형 결정 전략을 도식화한 결과이다. 각 라이브러리 제작 과정에 따른 자세한 마스킹 전략은 본문에 기술하였다.

실제 유전자형

	rs950122	rs113171913
개체1	C/C	C/C
개체2	G/C	C/T
개체3	G/G	C/C

IND 파일

개체명	성별	집단명
개체1	M	집단1
개체2	F	집단2
개체3	M	집단 3

SNP 파일

SNP 이름	염색체	유전적 위치 (M)	위치 (bp)	참조유전체 SNP	대립유전자 SNP
rs950122	1	0.022720	846864	G	C
rs113171913	1	0.023436	869303	C	T

GENO 파일

개체 1	개체 2	개체 3
0	1	2
2	1	2

그림 6. Eigenstrat 포맷의 도식화. 실제 유전자형이 어떻게 eigenstrat 포맷으로 표현되는지를 도식화한 결과이다. 주황색은 SNP rs950122에 대해 대응되는 정보를, 파란색은 개체2에 대해 대응되는 정보를 나타낸 것이다.

유전적 성별 결정

NGS 데이터의 경우 성별의 결정은 매우 쉬운 문제이다. 남성은 X 염색체와 Y 염색체 한 쌍씩, 여성은 X염색체 두 쌍을 갖기 때문에 참조유전체에 매핑 된 리드의 평균 커버리지를 상염색체에 매핑된 리드의 평균 커버리지와 비교를 함으로써 성별을 결정할 수가 있다. 유의점으로는 앞서 MAPQ 수치가 30이상인 리드들만 필터링을 했는데, Y 염색체의 다양한 반복 서열 구간을 커버하는 MAPQ가 30 이하인 리드들이 제거되기 때문에 MAPQ 30으로 필터링한 후의 Y염색체의 커버리지는 예상보다 떨어질 수 있다(그림 7A). 1240K SNP를 포함하는 리드들의 경우 반복 서열 주변이 아닌 영역들을 커버하기 때문에 Y 염색체의 커버리지를 X염색체나 상염색체와 공정하게 비교가 가능해진다. 실제로 데이터의 Y염색체의 커버리지와 X 염색체의 커버리지를 상염색체의 평균 커버리지로 나눈 후 비교하면 성별에 따른 뚜렷한 차이를 확인할 수 있다(그림 7B, 표 1).

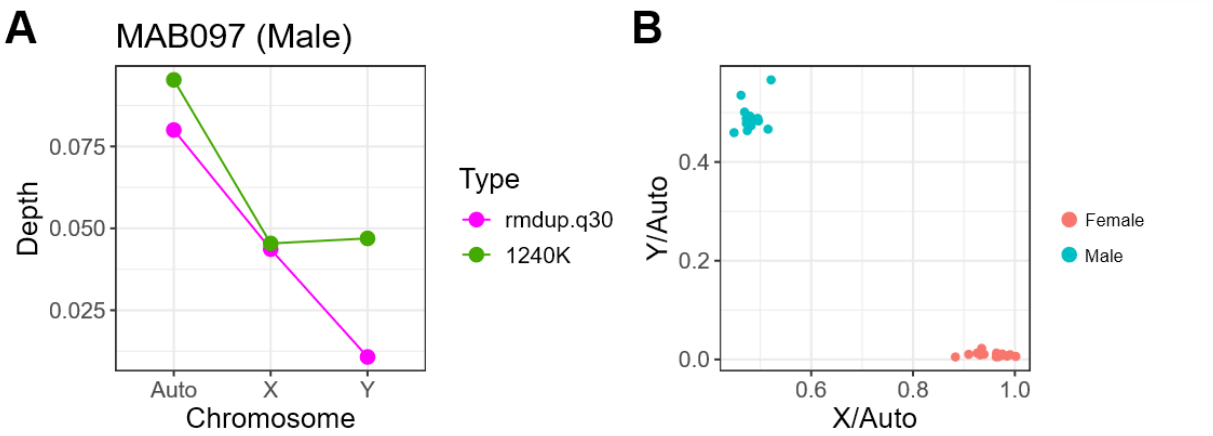

그림 7. 상염색체와 성염색체 커버리지를 이용한 성별 결정. A. MAB097 개체에 대한 매핑과 전처리 과정에 따른 상염색체와 성염색체의 평균 커버리지 변화. rmdup.q30: MAPQ 점수가 30 이하인 리드를 제거한 후, 1240K: 1240K SNP 자리를 커버하는 리드들만 남긴 이후. **B.** 성염색체와 상염색체의 비율을 비교하여 성별을 구분한 도식도.

표 1. 라이브러리 시퀀싱 결과별 상염색체, 성염색체 커버리지 및 성별 동정 결과. 라이브러리별로 커버리지를 구하고 성별을 동정한 결과. Autosome은 상염색체의 커버리지, X는 X 염색체의 커버리지, Y는 Y 염색체의 커버리지.

개체 ID	Library	Autosome	X	Y	Sex
MAB097	Non-UDG	0.1123	0.0534	0.0542	M
	Full-UDG	0.1894	0.0901	0.0917	M
	Partial-UDG	0.0953	0.0454	0.0469	M
MAB098	Non-UDG	0.1527	0.0783	0.0709	M
	Full-UDG	0.3181	0.1575	0.1553	M
	Partial-UDG	0.0474	0.0222	0.0238	M
MAB099	Non-UDG	0.0862	0.0403	0.0417	M
	Full-UDG	0.0585	0.0280	0.0288	M
	Partial-UDG	0.0151	0.0079	0.0086	M
MAB100	Non-UDG	0.0608	0.0298	0.0285	M
	Full-UDG	0.1762	0.0836	0.0817	M
	Partial-UDG	0.0409	0.0195	0.0197	M
MAB102	Non-UDG	0.0940	0.0909	0.0005	F
	Full-UDG	0.0410	0.0395	0.0002	F
	Partial-UDG	0.0568	0.0547	0.0007	F
MAB103	Non-UDG	0.0954	0.0456	0.0449	M
	Full-UDG	0.0650	0.0307	0.0317	M
	Partial-UDG	0.0428	0.0198	0.0229	M
MAB104	Non-UDG	0.1184	0.0576	0.0561	M
	Full-UDG	0.0793	0.0356	0.0364	M
	Partial-UDG	0.2485	0.1174	0.1188	M
MAB105	Non-UDG	0.1257	0.1229	0.0011	F
	Full-UDG	0.1083	0.1067	0.0007	F
	Partial-UDG	0.1233	0.1190	0.0011	F
MAB106	Non-UDG	0.2312	0.2170	0.0021	F
	Full-UDG	0.1271	0.1122	0.0006	F
	Partial-UDG	0.1065	0.0969	0.0011	F
MAB107	Non-UDG	0.1375	0.1367	0.0013	F
	Full-UDG	0.1261	0.1167	0.0016	F
	Partial-UDG	0.0732	0.0682	0.0007	F
MAB108	Non-UDG	0.0754	0.0750	0.0004	F
	Full-UDG	0.0193	0.0188	0.0002	F
	Partial-UDG	0.0260	0.0243	0.0006	F

현대인 DNA 오염도 추정

NGS 기반 고DNA 분석법은 PCR 기반의 기존 고DNA 분석법과 대조되어 오염도의 직접 추정이 가능하다. 오염도의 수량적 추정은 사람 유전체에서 반수체(haploid)로 존재하는 DNA의 경우 시퀀싱 오류와 사후변성으로 인한 염기 치환을 제외하면 여러 리드에서 오직 하나의 염기만이 관찰되어야 한다는 점을 활용하여 계산된다. 이 때 사용되는 DNA 영역은 미토콘드리아의 DNA와 남성의 X 염색체 영역 중 재조합이 일어나지 않는 영역이다. 만약 오염이 일어난다면 집단에서 다형성(polymorphism)을 갖는 것으로 알려진 영역에서 여러 다른 염기들이 관찰될 것이며, 이런 영역에서의 다형성의 증가 정도를 수치화하여 오염도를 추정할 수 있다. 대표적으로 미토콘드리아 DNA를 기반으로 오염도를 추정하는 Schmutzi 프로그램(Renaud et al. 2015)과 남성의 X 염색체 영역을 기반으로 오염도를 추정하는 ANGSD 프로그램의 contamination 모듈(Korneliussen et al. 2014)이 있다. 미토콘드리아 DNA 기반 오염도 추정의 경우 핵염색체와 미토콘드리아 DNA의 비율이 추출 부위에 따라 다르기 때문에 실제 핵염색체 DNA 오염의 비율과 다른 값을 가질 수 있다. 그에 반해 ANGSD 프로그램의 contamination 모듈은 핵염색체 오염 비율을 직접 추정이 가능하나 X 염색체를 두 개 갖는 여성에 대해서는 핵염색체 오염의 비율을 추정하지 못한다. 본 연구에서는 schmutzi 프로그램과 ANGSD 프로그램의 contamination 모듈 v.0.937을 활용하였다(표 2). 미토콘드리아 DNA 오염도를 추정하기 위해 먼저 미토콘드리아 DNA에 매핑된 리드들만을 필터링해 준 후, samtools fillmd 기능으로 미토콘드리아 참조유전체 서열 기준 불일치와 결실을 표시한 BAM 파일의 MD 태그값을 올바르게 고쳐주었다. 리드 수가 30,000개 이상 넘어가는 경우 Schmutzi 프로그램의 원활한 실행을 위해 samtools view -s 옵션으로 30,000 개 리드를 무작위로 추출한 BAM을 만들어 그 BAM에 대해 작업해주었다. Schmutzi 프로그램의 경우 contDeam.pl 스크립트를 먼저 돌린 후, schmutzi.pl 스크립트를 돌려주었다. 두 스크립트 모두 초기값으로 리드의 말단으로부터 사후 변성이 높은 염기의 개수를 제공하는 --lengthDeam 옵션에 대해 non-UDG 라이브러리의 경우 20, partial-UDG와 full-UDG 라이브러리의 경우 2의 값을 주었으며, contDeam.pl 옵션의 경우 --library 옵션에 라이브러리의 종류에 맞게 double-strand 라이브러리에 맞게 double값을 주었다. schmutzi.pl 코드를 돌릴 때에는 --notusepredC 옵션을 주어 낮은 오염도의 시료에 대해 오염도 추정을 잘 할 수 있도록 해주었다. Non-UDG의 NEBNext® Ultra™ II 라이브러리 키트의 경우 contDeam.pl 코드의 Bam2Prof 코드가 5’의 C-to-T 치환의 부재로 제대로 된 파일을 만들지 못해 작동하지 않는 경우들이 존재하며, 이를 피하기 위해 bam2prof -double -5p ${of1}_nolen.endo.5p.prof -3p ${of1}_nolen.endo.3p.prof -length 20 명령어를 사용하여 endogeneous DNA의 deamination rate 대신 평균 deamination rate를 기반으로 인풋 파일을 만들었으며, contDeam.pl의 코드 부분 중 아래 코드의 부분을 주석처리하여 bam2prof 프로그램을 contDeam.pl이 돌리지 않도록 설정해 준 후 오염도를 추정하였다.

```
my $cmdBam2Prof = $bam2prof." -length $lengthDeam -endo -".$library." -5p ".$outputPrefix.".endo.5p.prof -
3p ".$outputPrefix.".endo.3p.prof $inbam";
runcmd($cmdBam2Prof);
```

표 2. 라이브러리 시퀀싱 결과별 오염도 추정 결과.

			ANGSD X contamination 모듈					Schmutzi (MT)		
개체	UDG	성별	nSNP	nFlank	rSNP	rFlank	오염도	MT 커버리지	오염도 (기본)	오염도 (변형)
	non-UDG	M	51	459	0.000	0.001	-0.003±0.003	27.84	0.98±0.01	0.01±0.01
MAB097	full-UDG	M	225	2025	0.002	0.000	0.005±0.005	79.12	NA	0.01±0.01
	partial-UDG	M	106	954	0.000	0.000	NA	41.53	0.01±0.01	0.01±0.01
	non-UDG	M	82	738	0.000	0.001	-0.004±0.002	11.87	NA	0.01±0.01
MAB098	full-UDG	M	527	4743	0.001	0.000	0.001±0.002	34.38	NA	0.01±0.01
	partial-UDG	M	56	504	0.000	0.002	-0.006± 0.006	6.54	NA	0.01±0.01
	non-UDG	M	3	27	0.000	0.021	-0.072±0.160	8.89	NA	0.01±0.01
MAB099	full-UDG	M	18	162	0.000	0.003	-0.010±0.011	22.91	NA	0.005±0.005
	partial-UDG	M	9	81	0.000	0.000	NA	8.08	NA	0.01±0.01
	non-UDG	M	24	216	0.000	0.003	-0.006±0.007	8.04	NA	0.01±0.01
MAB100	full-UDG	M	195	1755	0.000	0.001	-0.001±0.001	27.91	NA	0.005±0.005
	partial-UDG	M	19	171	0.000	0.000	NA	6.30	NA	0.01±0.01
	non-UDG	F						14.03	0.99±0.005	0.01±0.01
MAB102	full-UDG	F						8.58	NA	0.01±0.01
	partial-UDG	F						11.49	NA	0.01±0.01
	non-UDG	M	27	243	0.018	0.011	0.017±0.047	11.42	NA	0.01±0.01
MAB103	full-UDG	M	21	189	0.024	0.006	0.057±0.057	11.13	NA	0.01±0.01
	partial-UDG	M	17	153	0.000	0.000	NA	6.40	NA	0.01±0.01
	non-UDG	M	57	513	0.017	0.005	0.038±0.036	20.59	0.01±0.01	0.01±0.01
MAB104	full-UDG	M	36	324	0.000	0.000	NA	23.95	NA	0.01±0.01
	partial-UDG	M	384	3456	0.002	0.001	0.004±0.004	59.00	0.01±0.01	0.01±0.01
	non-UDG	F						15.29	NA	0.01±0.01
MAB105	full-UDG	F						17.37	NA	0.01±0.01
	partial-UDG	F						30.09	NA	0.01±0.01
	non-UDG	F						18.89	0.99±0.005	0.01±0.01
MAB106	full-UDG	F						16.92	NA	0.01±0.01
	partial-UDG	F						19.10	0.01±0.01	0.01±0.01
	non-UDG	F						23.09	NA	0.01±0.01
MAB107	full-UDG	F						34.71	NA	0.005±0.005
	partial-UDG	F						17.35	NA	0.01±0.01
	non-UDG	F						18.89	NA	0.01±0.01
MAB108	full-UDG	F						8.80	NA	0.01±0.01
	partial-UDG	F						12.29	NA	0.01±0.01

미토콘드리아 및 Y 하플로그룹 결정

사람 참조유전체에는 미토콘드리아 DNA와 Y 염색체 서열이 있기 때문에 NGS 데이터로부터 하플로그룹 정보를 쉽게 얻을 수 있다. 사람의 하플로그룹 정보는 집단의 이동과 구성, 성별에 따른 계보와 친족 추정에 유용하게 사용될 수 있다. 미토콘드리아 하플로그룹의 경우 Schmutzi의 endoCaller 프로그램을 이용하여 시료 별 미토콘드리아 consensus sequence를 추정하였고, 이로부터 얻은 fasta 형식의 파일로부터 HaploGrep v2.1.20 (Weissensteiner et al. 2016)을 이용하여 미토콘드리아 하플로그룹에 할당하였다(표 3). Y 염색체의 경우 앞서 1240K SNP 기준 유전자형을 결정한 방식으로 유전자형을 결정하되, ISOGG 데이터베이스의 Y 염색체 SNP 13,508 개에 대해 유전자형을 결정하였으며, pileupCaller에서 랜덤하게 유전자형을 결정하는 --randomHaploid 옵션 대신 단수체인 Y 염색체의 특성을 고려하여 다수의 리드들에서 관찰되는 SNP를 유전자형으로 결정하는 --majorityCall 옵션을 사용하였다. 얻은 유전자형으로부터 yHaplo 프로그램(Poznik 2016) (https://github.com/alexhbnr/yhaplo)을 사용하여 Y 하플로그룹을 할당했으며, --ancStopThresh 10 옵션을 주어 고DNA 자료에 적합한 수준의 결측치를 허용해주었다(표 3).

표 3. 미토콘드리아와 Y 하플로그룹 추정 결과.

개체	MT	Y
MAB097	D4j+16311	Q1a2 (Q-L57, Q-M346)
MAB098	F2a1	J (J-M304)
MAB099	U4b1a4	Q1a2a (Q-L475, Q-L53)
MAB100	K2b1a1	J2a2 (J-L581)
MAB102	D4j+(16286)	-
MAB103	G2a5	J (J-CTS3732, J-M304)
MAB104	C1e	Q1a2a1c (Q-L334, Q-L330)
MAB105	D4j12	-
MAB106	D4f	-
MAB107	K2b1a1	-
MAB108	H2b	-

친족 결정

NGS 데이터로부터 개체들 간 친족 관계의 추정을 쉽게 진행할 수 있다. 가까운 생물학적 친족의 경우 과거의 공통 조상으로부터 받은 DNA를 공유하게 되는 공동자손 동일성(Identity-by-descent; IBD)의 공유 양상을 분석함으로써 두 사람 간 근연관계를 추정할 수 있다. 한 사람은 대립 유전자를 두 개 갖게 되는데, 두 사람으로부터 대립 유전자를 각각 하나씩 샘플링 했을 때 두 대립 유전자가 같은 조상으로부터 기원했을 확률을 혈연 계수(coefficient of kinship; 개체 i와 j에 대해 F_{ij})이라고 정의한다. 이 정의에 근거하면 쌍둥이 혹은 내 자신과의 F_{ij} 값은 0.5, 부모-자손과 형제-자매는 F_{ij} 값이 0.25, 조손관계, 삼촌 조카 관계와 이부/이복 형제는 F_{ij} 값이 0.125가 된다. 이 혈연계수를 pseudohaploid 데이터로부터 추정하는 간단한 방법론 중 하나가 pairwise mismatch rate (PMR) 방법론이다(Kennett et al. 2017). PMR의 정의는 두 pseudohaploid 유전자형이 일치하지 않을 확률이며, 두 개체 사이에 일치하지 않는 SNP의 비율로 추정한다. 두 유전자형이 IBD일 확률 F_{ij}에 대한 사전 정보를 갖고 있다면 PMR의 이론적인 값을 계산할 수 있으며, 역으로 PMR 값을 안다면 두

유전자형이 IBD일 확률에 대한 정보를 바탕으로 친족 관계를 예측할 수 있다. 두 개체의 pseudohaploid 유전자형으로부터 각각 하나의 SNP를 뽑았을 때 정의상 pseudohaploid 유전자형 자체가 하나의 개체로부터 대립유전자 하나를 랜덤으로 샘플링을 한 것이므로 두 SNP는 F_{ij}의 확률로 IBD인 대립유전자이다. 두 SNP가 IBD이면 정의상 같은 대립유전자를 가지기 때문에, 다음과 같은 조건부 확률을 만족한다.

$$P(\text{두 SNP가 다름} \mid \text{두 SNP가 IBD}) = 0$$

하지만 설령 두 SNP가 IBD가 아니라고 하더라도 SNP 좌위 i의 한 대립유전자 SNP의 집단 비율이 p이고 하디-바인베르크 법칙을 따른다고 가정할 때 $1 - 2p_i(1-p_i)$의 확률로 두 SNP는 같을 수 있다. 실제 PMR 값을 계산할 때에는 여러 SNP 좌위들에 대해 평균적으로 IBD가 아닐 때 샘플링한 두 SNP가 다를 확률을 생각해 볼 수 있다. 즉, N개의 SNP 좌위에 대해서 다음과 같은 식이 만족되며, 이를 PMR의 기본값(baseline)을 지칭하도록 b라는 기호로 나타내도록 하겠다.

$$P(\text{두 SNP가 다름} \mid \text{두 SNP가 IBD가 아님}) = \sum_{i=1}^{N} 2p_i(1-p_i) \equiv b$$

이제 조건부확률을 이용하여 PMR과 F_{ij}의 이론적 관계를 구할 수 있으며, 다음과 같은 수식을 만족하게 된다.

$$PMR = P(\text{두 SNP가 다름}) =$$
$$P(\text{두 SNP가 다름} \mid \text{두 SNP가 IBD})\, P(\text{두 SNP가 IBD}) +$$
$$P(\text{두 SNP가 다름} \mid \text{두 SNP가 IBD가 아님})\, P(\text{두 SNP가 IBD가 아님}) =$$
$$(1 - F_{ij})b$$

즉, b와 PMR이 주어졌을 때 F_{ij}값을 구할 수 있고, F_{ij}값으로부터 두 개체 간 친족관계를 유추할 수 있다. b의 값은 모든 대립유전자 빈도를 정확히 아는 경우 이론식을 활용하여 계산하는 것이 가능하나, 샘플의 작은 크기와 결측치를 갖는 SNP 좌위들로 인해 각각 SNP의 대립유전자 빈도를 추정하여 b를 계산하는데 한계가 있어 샘플들 간의 PMR 값으로부터 역으로 b를 계산하는 전략을 취한다. 두 가지 방법으로 유추가 가능한데, 동일 개체 사이의 PMR을 구한 후, F_{ij}값이 0.5임을 이용해 동일 개체의 PMR 값의 절반을 b로 정하는 방법과 개체쌍의 대부분은 친족관계가 아닐 것이라는 가정 하에 PMR의 최빈값을 b로 정하는 방법이 있다.

PMR은 유전적 거리를 통해 친족을 결정하는 방법론으로, IBD 구간 자체를 추정하여 친족을 결정하는 KIN (Popli et al. 2023)또는 ancIBD (Ringbauer et al. 2024) 방법론이나 유전자형 패턴을 분류하여 관계를 결정하는 IBSrelate (Waples et al. 2019) 방법론에 비해 세부적인 친족 계열을 구분하는 능력은 제한적이며, 3차 친족보다 먼 관계를 추정하는데 한계를 지닌다. 또한 PMR은 SNP 간 연관비평형으로 발생하는 상관구조를 활용하지 않고 단순 평균을 내는 통계량이므로 Chromopainter/fineSTRUCTURE (Lawson et al. 2012)와 같이 하플로타입 기반 정보를 활용하는 방법론에 비해 미세한 집단 구조를 구분 짓는 데에는 한계를 지닌다. 그럼에도 불구하고 PMR은 그 적용이 간편하고 요구 데이터의 조건이 낮다는 장점에 있어, 정밀한 분석에 앞서 초기 단계에서 개체 간 유전적 연관성을 확인하는 용도로 널리 사용된다.

실제 데이터로 추정한 PMR 값을 두가지 방법으로 가시화해보았다. 시베트 하이르한 고DNA 데이터를 비교해본 결과 기본 PMR 값으로부터 중복된 개체로부터의 PMR, 1차 친족의 PMR, 그리고 2차 친족의 PMR 값에 맞게 PMR 값들이 분포하고 있음을 확인할 수가 있다(그림 8A). 또한, PMR을 기반으로 한 덴드로그램에 맞게 히트맵을 그려본 결과, 같은 개체로부터 온 라이브러리들이 먼저 뭉쳐지고, 그 후 친족관계가 가까운 개체들이 가깝게 나타나는 패턴을 뚜렷하게 관찰할 수가

있다(그림 8B). 본 분석을 통해 1차 친족 1쌍(MAB100과 MAB107)과 2차 친족 1쌍(MAB097과 MAB099)을 규명할 수 있었다.

결론

본 논문에서는 고DNA의 NGS 분석에 필수적으로 고려되어야 하는 요소들을 실제 고유전체 데이터를 기반으로 분석해 보았다. 고DNA의 NGS 분석은 일반적인 NGS 분석과는 달리 파편화된 길이, 낮은 시퀀싱 커버리지와 사후 변성에 의한 염기 치환의 특징을 적극적으로 고려해야 한다. 고DNA NGS 데이터를 바탕으로 실제 고DNA 여부의 검정, 고DNA의 현대인 DNA 오염도 추정, 고대인의 성별과 미토콘드리아와 Y 하플로그룹, 그리고 고대인 간 친족관계 추정을 진행할 수 있었다. 본 논문은 고DNA 분석의 정형화된 분석 기준과 방법론 일부에 대해 설명하였으며, 논문을 바탕으로 고DNA에 대한 관심과 분석 방법에 대한 가이드라인을 제공하고자 하였다.

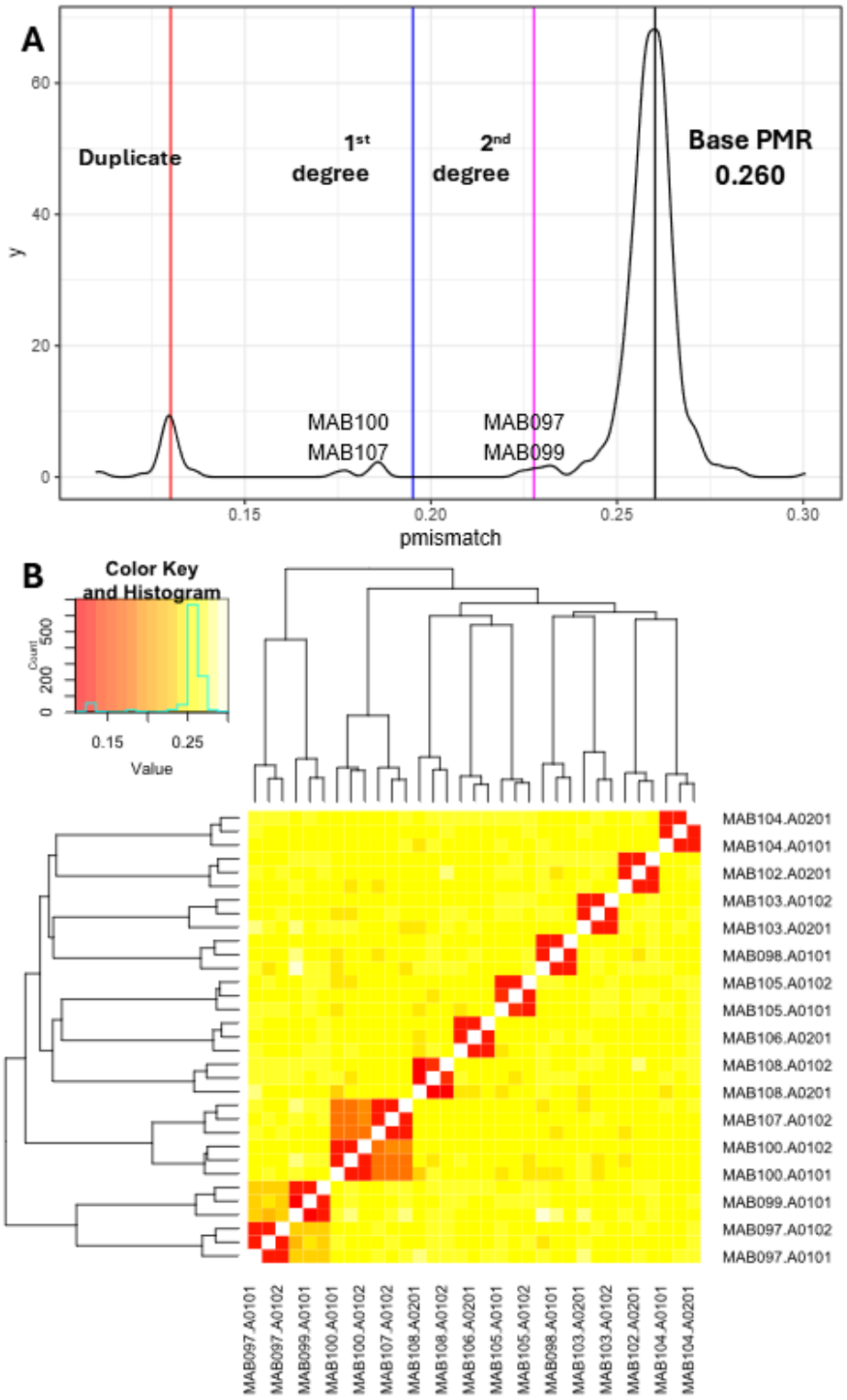

그림 8. 실제 데이터를 바탕으로 친족을 추정한 그림. A. PMR의 밀도 분포를 나타낸 도식. 친족관계가 없는 사람들 간 PMR 값의 최빈값을 집단의 평균 PMR 값으로 잡은 후, 동일 개체 간 PMR, 1차 친족 간 PMR, 그리고 2차 친족 간 PMR의 이론적인 값을 세로선으로 나타내주었다. **B.** PMR를 기반으로 한 덴도그램과 히트맵 그림. 붉은색일수록 PMR 값이 작음을 나타낸다.

부록

본 부록에서는 유닉스 계열의 컴퓨터를 활용하여 돌릴 수 있는 코드와 필요한 파일의 형식들을 기재하였다. 따로 언급이 있지 않는 이상 모든 코드는 배쉬(Bash) 문법으로 작성되었다. 모든 코드는 하나의 fastq 시퀀싱 결과에 대한 코드이나 필요에 따라 여러 시퀀싱 결과에 대해 병렬적으로 돌릴 수 있도록 구성할 수 있다. 또한 주석을 #을 이용해 달아주어 코드 중간 중간에 설명을 제공하였다.

리스트 파일의 양식

본 분석에서 사용되는 리스트 파일은 분석하는 샘플에 대한 기본적인 정보를 담고 있다. 리스트 파일은 여섯 개의 열로 이루어진 표의 형식이며, 파일에는 헤더를 포함하지 않아야 하고, 표의 구분자로 공백이나 탭(tab)을 사용해야 한다. (표 4)는 동일 개체로부터 온 세 라이브러리에 대해 작성한 리스트 파일이다.

fastq 파일이 있는 경로의 경우에는 SE 시퀀싱의 경우 fastq 파일 하나의 경로를, PE 시퀀싱의 경우 리드1과 리드2 파일 각각을 쉼표로 구분하여 모두 입력해주어야 한다. 라이브러리의 종류로는 ssLib (Single-strand library), non-UDG, non-UDG_NEB, partial-UDG, full-UDG의 다섯 가지 중 하나를 입력해야 하며, 종에는 추후 서술할 참조유전체 목록 파일에 있는 종의 이름을 입력해주어야 한다.

표 4. 리스트 파일의 예시.

IID	LID	RID	fastq 파일이 있는 경로	라이브러리 종류	종
MAB097	MAB097.A0101	MAB097.A0101.SG1.1	/fastq/파일까지의/절대경로/MAB097.A0101.SG1.1_1.fastq.gz,/fastq/파일까지의/절대경로/MAB097.A0101.SG1.1_2.fastq.gz	non-UDG_NEB	Human
MAB097	MAB097.A0102	MAB097.A0102.SG1.1	/fastq/파일까지의/절대경로/MAB097.A0102.SG1.1_1.fastq.gz,/fastq/파일까지의/절대경로/MAB097.A0102.SG1.1_2.fastq.gz	full-UDG	Human
MAB097	MAB097.A0201	MAB097.A0201.SG1.1	/fastq/파일까지의/절대경로/MAB097.A0201.SG1.1_1.fastq.gz,/fastq/파일까지의/절대경로/MAB097.A0201.SG1.1_2.fastq.gz	partial-UDG	Human

참조유전체 목록 파일과 양식

본 분석에서는 종에 따른 참조유전체의 경로를 제공하기 위해 참조유전체 목록 파일을 만들어 사용한다. 두 개의 열로 이루어져 있으며, 리스트 파일과 마찬가지로 헤더를 포함하지 않고, 표의 구분자로 공백이나 탭을 사용해야 한다.

표 5. 참조유전체 목록 파일의 예시.

종	참조유전체 파일이 있는 경로
Human	/home/References/Human/hs37d5.fa
Horse	/home/References/Horse/equCab3.fa
Cattle	/home/References/Cattle/ARS-UCD1.2.chrY.fa
Sheep	/home/References/Sheep/oviAri4.fa
Goat	/home/References/Goat/Saanen_v1_MT.fa

참조유전체 인덱싱(Index)하기

분석을 진행하기 전, 참조유전체 fasta 파일을 BWA와 samtools가 쉽게 접근할 수 있도록 인덱싱을 해주어야 한다. 특히 BWA를 이용한 매핑에는 필수적으로 인덱싱을 해주어야 한다.

코드 1. 참조유전체 인덱싱 코드.

```
path="참조유전체/파일의/경로.fa"


bwa index -a bwtsw "${path}" # Create BWA FM-index
samtools faidx "${path}"  # Create samtools index
```

1240K bed 파일

1240K 유전자형 결정을 위해 참조유전체 기준으로 SNP의 염색체 상 위치 정보를 담고 있는 bed 포맷의 파일이 필요하다. 본 분석에서는 (표6)과 같은 형식의 bed 파일을 사용하였으며, 가장 위 두 줄 만 표시하였다. SNP에 대한 정보이기 때문에 0기준 시작 위치와 1 기준 끝 위치의 차이는 1이다.

표 6. bed 파일의 예시.

염색체	0 기준 시작 위치	1 기준 끝 위치	이름	점수	양성/음성
1	752565	752566	test	1000	+
1	776545	776546	test	1000	+

작업 환경 폴더 만들어주기

이후 과정의 코드들은 정해진 폴더 구조를 따르도록 되어 있다. 아래 코드는 작업물들이 만들어질 폴더를 만드는 코드이다.

코드 2. 작업 환경 폴더를 만들어주는 예시 코드.

```
mkdir Shiveet_Khairkhan; cd Shiveet_Khairkhan


mkdir -p DataProcessing; mkdir -p genotypes; mkdir -p analysis

cd DataProcessing
```

코드 3. main_utils.sh로 저장한다. 각각의 단계에서 반복적으로 리스트 파일을 읽거나 변수 지정이 올바르게 되었는지를 확인하는 명령어들이 반복적으로 사용이 된다. 이런 명령어 함수들을 한 곳에 모아둔 후, source 라는 명령어를 사용하여 여러 코드에서 같은 함수를 사용하였다.

```
#!/bin/bash
# 리스트 파일(listf)과 작업할 행(inum)을 기반으로 변수들을 읽어오고 화면에 출력해준다.
read_listfile_into_variables() {
    read -r iid lid rid tfqs libtype spp < <(awk -v inum="${inum}" 'inum == NR {print}' "${listf}")
    if [[ -z "${iid}" || -z "$(Skourtanioti et al.)" || -z "${rid}" || -z "${tfqs}" || -z "${libtype}" ||
-z "${spp}" ]]; then
      printf "Error: Missing values in list file for task ID '$inum'. Please check your list file.\n" >&2
      exit 1
    fi
    cat << EOF
Individual ID: ${iid}
Library ID: $(Skourtanioti et al.)
Run ID: ${rid}
Files to be processed: ${tfqs}
Library type: ${libtype}
Species list: ${spp}
EOF
}
# libtype 변수가 정해진 항목 안에 제대로 정의되었는지 확인한다.
check_librarytype() {
    if [[ "${libtype}" =~ ^(ssLib|non-UDG|non-UDG_NEB|partial-UDG|full-UDG|Modern)$ ]]; then
        echo "Library type is " ${libtype}
    else
        echo "Error: ${libtype} does not match category, exiting."
        echo "Error: Categories are: ssLib|non-UDG|non-UDG_NEB|partial-UDG|full-UDG|Modern."
        exit 1
    fi
}
# tfqs 변수를 기반으로 stval 변수를 SE 또는 PE 로 설정한다.
set_stval() { ## Take a list of FastQ files and decide if they are SE or PE
    stval=($(echo ${tfqs} | sed s/","/"\n"/g | awk '$1 ~ /_R2/ || $1 ~ /_2.fastq.gz/ || $1 ~ /_2.fq.gz/' |
wc -l | awk '{if ($1 > 0) print "PE"; else print "SE"}'))
    printf "This library was "${stval}" sequenced.\n"
}
```

어댑터 제거하기

코드 4. 어댑터 제거를 위한 코드 파일. wrap_adapterremoval.sh 로 저장해준다.

```
#!/bin/bash
source /path/to/main_utils.sh  # [코드 3]
pt1=$1 # 작업 경로
listf=$2   ## 리스트 파일
inum=$3    ## 리스트 파일에서 작업할 시퀀싱 자료의 행
optstr="--gzip --threads 8 --trimns --trimqualities "
    optstr+="--adapter1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC "
    optstr+="--adapter2 AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTA "
    optstr+="--minlength 35 --minquality 20 --minadapteroverlap 1"
main() {
    read_listfile_into_variables; check_fastq_sanity
    mkdir -p "${pt1}${iid}/FastQ/${rid}" && cd "${pt1}${iid}/FastQ/${rid}"
    set_stval ;  set_fastq_input ;  run_adapterremoval
}
check_fastq_sanity() {
    while read fq; do
        if [[ "$fq" =~ \.fastq\.gz$ || "$fq" =~ \.fq\.gz$ ]]; then
            if [ -f "$fq" ]; then
                echo "fastq ${fq} has correct suffix"
            else
                echo "fastq ${fq} does not exist!" >&2 && exit 1
            fi
        else
            echo "Line does not end with fastq.gz or fq.gz." >&2 && exit 1
        fi
    done < <(echo ${tfqs} | sed s/","/"\n"/g  )
}
set_fastq_input() {
    fq1s=""; while read fq; do
        if [[ "$fq1s" == "" ]]; then fq1s+=${fq}; else fq1s+=" "${fq}; fi
    done < <( echo ${tfqs} \
        | sed s/","/"\n"/g \
        | awk '$1 !~ /_R2/ && $1 !~ /_2.fastq.gz/ && $1 !~ /_2.fq.gz/' )
    if [[ "$stval" == "PE" ]]; then
        fq2s=""; while read fq; do
            if [[ "$fq2s" == "" ]]; then fq2s+=${fq}; else fq2s+=" "${fq}; fi
        done< <( echo ${tfqs} \
        | sed s/","/"\n"/g \
        | awk '$1 ~ /_R2/ || $1 ~ /_2.fastq.gz/ || $1 ~ /_2.fq.gz/')
    fi
    printf "Fastq1s:  "${fq1s}", Fastq2s: "${fq2s}"\n"
}
run_adapterremoval() {
    fqout=${rid}".L35.fq"
    if [[ "$stval" == "SE" ]]; then
        AdapterRemoval --file1 ${fq1s} --basename ${fqout} ${optstr}
    else
        AdapterRemoval --file1 ${fq1s} --file2 ${fq2s} --basename ${fqout} ${optstr} --collapse
        zcat ${fqout}.collapsed.gz ${fqout}.collapsed.truncated.gz ${fqout}.singleton.truncated.gz \
        | gzip > ${fqout}.combined.gz
    fi
}
main
```

코드 5. 어댑터 제거를 위한 코드, 명령창에 입력하여 돌려준다.

```
scf1="./wrap_adapterremoval.sh" # 위에 만들어준 코드의 경로
pt1=($(pwd)"/")
listf="리스트/파일/경로.list"; inum=1 # 리스트 파일의 행 중 하나
chmod +x "${scf1}"  # 실행 권한 부여
"${scf1} ${pt1} ${listf} ${inum}" # 코드 실행
```

매핑 작업

코드 6. wrap_mapping_ancient.sh로 저장해준다.

```
#!/bin/bash
source /path/to/main_utils.sh  # [코드 3]
pt1=$1
listf=$2
inum=$3
reff=$4
main() {
    mkdir -p ${pt1}${iid}/BAM/ && cd ${pt1}${iid}/BAM/
    read_listfile_into_variables ; check_librarytype; set_seedval; set_stval
    set_input_truncated_fastq_based_on_stval
    run_mapping_job
}
set_seedval() {
    seedval="32"; if [[ "${libtype}" == "ssLib" ]] || [[ "${libtype}" == "non-UDG" ]] || [[ "${libtype}"
== "non-UDG_NEB" ]] ; then seedval="9999"; fi
}
set_input_truncated_fastq_based_on_stval() {
    ## Input FastQ (after running AdapterRemoval)
    if [[ "${stval}" == "SE" ]]; then
        fqin=${pt1}${iid}"/FastQ/"${rid}"/${rid}.L35.fq.truncated.gz"
    else
        fqin=${pt1}${iid}"/FastQ/"${rid}"/"${rid}".L35.fq.combined.gz"
    fi
    if [ ! -f "${fqin}" ]; then
        echo "${fqin} required, but doesn't exist! Did you run adapterremoval?" >&2 && exit 1
    fi
    printf "${fqin} is used for mapping\n"
}
run_mapping_job() {
    printf "Mapping on each reference\n"
    snum=0
    for sp in $(echo ${spp} | sed s/","/" "/g); do
        let snum+=1
        printf "2."${snum}" Read Mapping on "${sp}"\n"
        ref=($(awk -v sp="$sp" '{if ($1 == sp) print $2}' ${reff}))  ## Take the reference genome .fa file
        sid=${rid}"."${sp}
        ## Make directory for each run id + reference
        mkdir -p ${sid}
        ## Setup output BAM file prefix
        of1="./"${sid}"/"${sid}".L35.mapped"
        ## Define read group
        RG="@RG\tID:"${rid}"\tSM:"$(Skourtanioti et al.)"\tLB:"${rid}"\tPL:illumina"
        ## Run BWA aln (-l 9999 for non-UDG, -l 32 for UDG-half)
        bwa aln -t 8 -n 0.01 -l ${seedval} -f ${of1}.sai ${ref} ${fqin}
        ## Run BWA samse and filter out unmapped reads
        bwa samse -r ${RG} ${ref} ${of1}.sai ${fqin} | samtools view -h -F 0x0004 -o ${of1}.0.bam -
        ## Sort the output file and index
        samtools sort -@ 2 -m 4G ${of1}.0.bam -o ${of1}.bam
        samtools index ${of1}.bam
        ## Remove temporary files
        rm ${of1}.0.bam &&  rm ${of1}.sai
        run_dedup
        ## Apply quality filter (-q30)
        samtools view -bh -q30 -o ${of1}.rmdup.q30.bam ${of1}.rmdup.bam
        samtools index ${of1}.rmdup.q30.bam
    done
    printf "\n\nMapping is totally complete. \n\n"
}




run_dedup() {
    dedup="java -Xmx8192m -jar /opt/ohpc/pub/apps/dedup/0.12.8/DeDup-0.12.8.jar"
```

```
    ## Remove duplicates using dedup
    ${dedup} -i ${of1}.bam -m -o ./${sid}/
    mv ${of1}_rmdup.bam ${of1}.rmdup.bam
    samtools index ${of1}.rmdup.bam
    rm ${of1}.dedup.json

    mv ${of1}.hist ${of1}.rmdup.hist
    mv ${of1}.log ${of1}.rmdup.log
}

main
```

코드 7. 실제 명령어는 다음과 같이 돌려준다.

```
scf1="./wrap_mapping_ancient.sh" # 위에 만들어준 코드의 경로
pt1=($(pwd)"/")
listf="리스트/파일/경로.list"
inum=1 # 리스트 파일의 행 중 하나
reff="참조유전체/목록/파일.list"
chmod +x "${scf1}"  # 실행 권한 부여

"${scf1} ${pt1} ${listf} ${inum} ${reff}" # 코드 실행
```

mapDamage

코드 8. wrap_mapdamage.sh로 저장한다.

```
#!/bin/bash
source /path/to/main_utils.sh  # [코드 3]
pt1=$1 # work directory
listf=$2   ## A list of samples to be processed (LID, run, FastQs, type)
inum=$3
reff=$4
nrmax=100000

main() {
    read_listfile_into_variables
    cd ${pt1}${iid}/mapDamage/
    for sp in $(echo ${spp} | sed s/","/" "/g); do
        ref=($(awk -v sp="$sp" '{if ($1 == sp) print $2}' ${reff}))  ## Take the reference genome.fa file
        sid=${rid}"."${sp}
        ibam=($(realpath ${pt1}${iid}/BAM/${sid}/${sid}.L35.mapped.rmdup.q30.bam))
        ## Run mapDamage
        mapDamage -i ${ibam} -r ${ref} \
          -d ./${sid} --merge-reference-sequences -t ${sid} -n ${nrmax} --no-stat
    done
    printf "\n\nmapDamage is complete!\n\n"
}
main
```

코드 9. 실제 명령어는 다음과 같이 돌려준다.

```
scf1="./wrap_mapdamage.sh" # 위에 만들어준 코드의 경로
pt1=($(pwd)"/")
listf="리스트/파일/경로.list"
inum=1 # 리스트 파일의 행 중 하나
reff="참조유전체/목록/파일.list"
chmod +x "${scf1}"  # 실행 권한 부여
"${scf1} ${pt1} ${listf} ${inum} ${reff}" # 코드 실행
```

오염도 추정

코드 10. wrap_schmutzi.sh로 저장한다.

```
#!/bin/bash
```

```
source /path/to/main_utils.sh  # [코드 3]
pt1=$1 # work directory
listf=$2   ## A list of samples to be processed (LID, run, FastQs, type)
inum=$3

main() {
    read_listfile_into_variables && check_librarytype && set_strandtype
    ibam=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.L35.mapped.rmdup.q30.bam  ## Input BAM file
    set_deam_len
    if [[ ! -f "${ibam}" ]]; then
        echo "Error: Input BAM file '$ibam' does not exist!" >&2 && exit 1
    fi
    printf "Separating Mitochondrial reads\n\n\n"
    MTref1="/home/References/Human/hg19_MT.fasta"
    MTref2="/home/References/Human/hg19_MT_500.fasta"
    ctgn="NC_012920.1"
    nrmax=50000  ## Use max 50,000 reads

    rg="@RG\tID:"${rid}.${spp}"\tSM:"${iid}"\tLB:"${rid}"\tPL:illumina"
    of1=${rid}.${spp}.circmapper
    bam1=${of1}.rmdup.q30.MD.small.bam
    tn1="temp1_"${rid}.${spp}

    mkdir -p ${pt1}${iid}/circularMT/${rid}.${spp} ; cd ${pt1}${iid}/circularMT/${rid}.${spp}
    samtools view -h ${ibam} MT \
    | awk -v ctgn="$ctgn" '{OFS="\t"} {if ($1 == "@SQ" && $2 ~ /MT$/) print $1,"SN:"ctgn,$3;
    else if ($1 == "@SQ"); else if ($1 ~ /^@/) print $0; else { $3=ctgn; print $0 } }' \
    | samtools view -bh -o ${of1}.rmdup.q30.bam -
    samtools index ${of1}.rmdup.q30.bam
    samtools fillmd -b ${of1}.rmdup.q30.bam ${MTref1} > ${of1}.rmdup.q30.MD.bam
    samtools index ${of1}.rmdup.q30.MD.bam

    ## Downsample reads to run Schmutzi easily (capped to 30,000 reads)
    nr1=$(samtools view -c ${of1}.rmdup.q30.MD.bam)
    pr1=$(echo ${nr1} | awk -v nrmax="$nrmax" '{printf "%.3f\n", nrmax/$1}')
    if [ "$nr1" -le "$nrmax" ]; then
        cp ${of1}.rmdup.q30.MD.bam ${of1}.rmdup.q30.MD.small.bam
        samtools index ${of1}.rmdup.q30.MD.small.bam
    else
        samtools view -bh -s ${pr1} -o ${of1}.rmdup.q30.MD.small.bam ${of1}.rmdup.q30.MD.bam
        samtools index ${of1}.rmdup.q30.MD.small.bam
    fi

    printf "Starting schmutzi\n\n\n"
    cd ${pt1}; mkdir -p  ${iid}/schmutzi/${rid}.${spp} && cd ${iid}/schmutzi/${rid}.${spp}

    ## Run contDeam (Schmutzi step 1)
    contDeam.pl --library ${strandtype} \
        --lengthDeam ${lenDeam} \
        --out ./${of1}_nolen ${MTref1} \
        ${pt1}${iid}/circularMT/${rid}.${spp}/${bam1}

    ## Run schmutzi (Schmutzi step 2)
    schmutzi.pl \
        -t 8  \
        --notusepredC \
        --lengthDeam ${lenDeam} --ref ${MTref1} \
        ./${of1}_nolen /opt/ohpc/pub/apps/schmutzi/share/schmutzi/alleleFreqMT/197/freqs \
        ${pt1}${iid}/circularMT/${rid}.${spp}/${bam1}


   printf "Starting schmutzi without prior contamination estimate \n\n\n"

    ## Move into Schmutzi directory
    cd ${pt1}; mkdir -p  ${iid}/schmutzi_mean/${rid}.${spp} && cd ${iid}/schmutzi_mean/${rid}.${spp}
```

```
    ## Run contDeam (Schmutzi step 1)

    bam2prof -double -5p ./${of1}_nolen.endo.5p.prof -3p ./${of1}_nolen.endo.3p.prof \
     -length 20 ${pt1}${iid}/circularMT/${rid}.${spp}/${bam1}

    contDeam.noest.pl --library ${strandtype} \
        --lengthDeam ${lenDeam} \
        --out ./${of1}_nolen ${MTref1} \
        ${pt1}${iid}/circularMT/${rid}.${spp}/${bam1}

    ## Run schmutzi (Schmutzi step 2)
    schmutzi.pl \
        -t 8  \
        --notusepredC \
        --lengthDeam ${lenDeam} --ref ${MTref1} \
        ./${of1}_nolen /opt/ohpc/pub/apps/schmutzi/share/schmutzi/alleleFreqMT/197/freqs \
        ${pt1}${iid}/circularMT/${rid}.${spp}/${bam1}

}
set_strandtype() {
    if [[ ${libtype} == "ssLib" ]]; then
        strandtype="single"  # single or double
    else
        strandtype="double"
    fi
}
set_deam_len() {
    lenDeam=2   ## deamination
    if [[ "${libtype}" == "non-UDG" || "${libtype}" == "non-UDG_NEB" ]]; then
         lenDeam=20
    fi
}
main
```

코드 11. wrap_angsd_xcont.sh로 저장한다.

```
#!/bin/bash
source /path/to/main_utils.sh  # [코드 3]
pt1=$1
listf=$2
inum=$3
main() {
    read_listfile_into_variables
    bam=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.L35.mapped.rmdup.q30.bam
    of1=${pt1}${iid}/Xcont/${rid}.${spp}/${rid}.${spp}.angsdCounts
    mkdir -p ${pt1}${iid}/Xcont/${rid}.${spp}

    # Generate ANGSD input files
    angsd -i $(Mallick et al.) -r X:5000000-154900000 -doCounts 1 -iCounts 1 -minMapQ 30 -minQ 30 -out
${of1}

    # Run contamination level estimation
    /opt/ohpc/pub/apps/angsd/misc/contamination -a ${of1}.icnts.gz \
    -h /opt/ohpc/pub/apps/angsd/RES/HapMapChrX.gz 2> ${of1}
}

main
```

코드 12. 실제 명령어는 다음과 같이 돌려준다.

```
scf1="./wrap_schmutzi.sh" # 위에 만들어준 schmutzi 코드의 경로
scf2="./wrap_angsd_xcont.sh" # 위에 만들어준 angsd의 X contamination 모듈 코드의 경로
pt1=($(pwd)"/")
listf="리스트/파일/경로.list"
```

```
inum=1 # 리스트 파일의 행 중 하나
chmod +x "${scf1}"  # 실행 권한 부여
chmod +x "${scf2}"  # 실행 권한 부여

"${scf1} ${pt1} ${listf} ${inum}" # 코드 실행
"${scf2} ${pt1} ${listf} ${inum}"
```

유전자형 결정

유전자형 결정을 하기 위해서 만들어지는 eigenstrat 파일의 집단명을 지정해주는popfile(표 7)이랑 리드의 염기를 마스킹해주는 정도를 지정해주는maskfile(표 8)이 필요하다. 두 파일 모두 헤더가 없으며 두 개의 열로 되어 있어야 한다.

표 7. Popfile의 예시.

라이브러리의 RID	Eigenstrat 포맷에서의 집단 명
MAB097.A0101.SG1.1	Shiveet_Khairkhan_nonUDG
MAB097.A0102.SG1.1	Shiveet_Khairkhan_fullUDG
MAB097.A0201.SG1.1	Shiveet_Khairkhan_partialUDG

표 8. Maskfile 목록의 예시.

라이브러리의 RID	마스킹할 염기의 개수
MAB097.A0101.SG1.1	20
MAB097.A0102.SG1.1	0
MAB097.A0201.SG1.1	2

코드 13. 리드 중 1240K SNP 좌위를 커버하는 리드들을 추출하는 코드. wrap_extract_1240K.sh로 저장해준다.

```
#!/bin/bash
pt1=$1 # work directory
listf=$2   ## A list of samples to be processed (LID, run, FastQs, type)
inum=$3
reff=$4
bedf=$5
set -euo pipefail
source /path/to/main_utils.sh  # [코드 3]
main() {
    if [ -z "${bedf}" ] || [ ! -f "${bedf}" ]; then
        echo "ERROR: bed file is missing or does not exist." >&2
        exit 1
    fi
    read_listfile_into_variables
    ###################################################################
    ## Stage 1. Extract 1240K reads.
    ibam=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.L35.mapped.rmdup.q30.bam
    of1=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.1240K.rmdup.q30

    ## Extract 1240K reads by providing bed file.
    samtools view -bh -q30 -L ${bedf} -o ${of1}.1.bam ${ibam}
    samtools index ${of1}.1.bam
    ## Extract MT reads by MT tag.
    samtools view -bh -q30 -o ${of1}.2.bam ${ibam} MT
    samtools index ${of1}.2.bam
    samtools merge -f -c -p ${of1}.bam ${of1}.1.bam ${of1}.2.bam # I set -f so it overwrites.
    samtools index ${of1}.bam
```

```
    rm ${of1}.1.*; rm ${of1}.2.*
    ###################################################################
    ## Stage 2. Calculate 1240K depth.

    ibam=${of1}.bam
    mkdir -p ${pt1}${iid}/coverage/${rid}.${spp}
    samtools depth -q30 -Q30 -aa -b ${bedf} ${ibam} |\
    awk -v lid=${rid}.${spp} 'BEGIN { a = 0; x = 0; y = 0;sa = 0; sx = 0; sy = 0} $1 != "X" && $1 != "Y"
{a = a + $3 ; sa = sa + 1 } $1 == "X" {x = x + $3 ; sx = sx + 1} $1 == "Y" {y = y + $3 ; sy = sy + 1} END
{print lid FS a/sa FS x/sx FS y/sy}'\
     > ${pt1}${iid}/coverage/${rid}.${spp}/${rid}.${spp}.txt
}

main
```

코드 14. wrap_masksplitbam.sh으로 저장해준다.

```
#!/bin/bash
pt1=$1
listf=$2
inum=$3
maskf=$4
source /path/to/main_utils.sh  # [코드 3]
set -euo pipefail

main() {
    if [ -z "${maskf}" ] || [ ! -f "${maskf}" ]; then
        echo "ERROR: maskf file is missing or does not exist." >&2
        exit 1
    fi
    read_listfile_into_variables && check_librarytype
    nmask=$(awk -v rid="${rid}" '$1 == rid {print $2}' ${maskf} )
    nmask=${nmask:-0}
    inbam=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.1240K.rmdup.q30.bam
    if [ ${libtype} == "non-UDG_NEB" ]; then
        if [ -n "${nmask}" ] && [ "${nmask}" -gt 0 ]; then
            echo "Number of masked bases set and is ${nmask}"
        else
            echo "Error: nmask:${nmask} not set or 0, but non-UDG_NEB... aborting..." >&2
            exit 1
        fi

        obam=$(echo ${inbam} | sed s/".q30.bam"/".q30.lm"${nmask}".bam"/g)
        # Create positive strand/ negative strand bam files.
        obamps=$(echo ${obam} | sed s/"q30.lm${nmask}.bam"/"q30.lm${nmask}.ps.bam"/g)
        obamns=$(echo ${obam} | sed s/"q30.lm${nmask}.bam"/"q30.lm${nmask}.ns.bam"/g)

        if [ "$inbam" == "$obam" ] || [ "$inbam" == "$obamps" ] || [ "$inbam" == "$obamns" ]; then
            echo "Error: inbam matches another variable."; exit 1
        elif [ "$obam" == "$obamps" ] || [ "$obam" == "$obamns" ]; then
            echo "Error: obam matches another variable."; exit 1
        elif [ "$obamps" == "$obamns" ]; then
            echo "Error: obamps matches obamns." ; exit 1
        else
            echo "All variables are unique."
        fi

        # We can give i option to mask by positive strand direction
        # But we don't since samtools aligned reads(0 and 16 flags) are already reference sequence based.
(flag 16 is reverse complemented reads)
        bam trimBam ${inbam} ${obam} -L ${nmask}
        samtools index ${obam}

        samtools view -bh -F 16 -o ${obamps} ${obam}
```

```
        samtools view -bh -f 16 -o ${obamns} ${obam}
    elif [ ${nmask}  -gt 0 ]; then
        obam=($(echo ${inbam} | sed s/".q30.bam"/".q30.m"${nmask}".bam"/g))
        if [ "$inbam" == "$obam" ]; then
            echo "inbam matches another variable."; exit 1
        else
            echo "All variables are unique."
        fi
        bam trimBam ${inbam} ${obam} ${nmask}
        samtools index ${obam}
    elif [ ${libtype} == "partial-UDG" ]; then
        echo "Warning: Not masking due to nmask set to 0 or unspecified"
        echo "Warning: Are you sure you want to avoid masking for ${rid}.${spp}?"
    fi
}

main
```

코드 15. wrap_pileupcaller_eigenstrat.sh로 저장해준다.

```
#!/bin/bash
set -euo pipefail

pt1=$1 # work directory
listf=$2   ## A list of samples to be processed (LID, run, FastQs, type)
inum=$3
reff=$4
bedf=$5
maskf=$6
popfile=$7
mode=$8
folder=$9
workprefix=${10}
snpf1=${11}
posf=${12}
filtersex=${13}
source /path/to/main_utils.sh  # [코드 3]

main() {

    sanity_check_pileupcaller_options &&  read_listfile_into_variables
    # Set reference. Modify code so that it can be used with other ways.
    for sp in $(echo ${spp} | sed s/","/" "/g); do
        ref=($(awk -v sp="$sp" '{if ($1 == sp) print $2}' ${reff}))  ## Take the reference genome .fa file
    done
    check_librarytype && sanity_check_popfile
    nmask=$(awk -v rid="${rid}" '$1 == rid {print $2}' ${maskf} )
    nmask=${nmask:-0}

    set_input_output_based_on_mask_and_library
    printf "Eigenstrat sample ID is ${outprefix}\n"
    printf "bam prefix is ${bamprefix}\n"
    mkdir -p ${pt1}${iid}/${folder}/${rid}.${spp}/
    of1=${pt1}${iid}/${folder}/${rid}.${spp}/${outprefix}
    tn1=${pt1}${iid}/${folder}/${rid}.${spp}/"temp1_"${outprefix}


    ## Create IND and SNP file.
    get_sample_sex_based_on_coverage
    if [[ "${filtersex}" == ${sex}  ]]; then
        echo "Sample has same sex with filter sex, aborting genotype call."
        exit 0
    fi
    echo -e "${outprefix}\t${sex}\t${pop}" > ${of1}.ind
    cp ${snpf1} ${of1}.snp
```

```
        run_pileupcaller_based_on_libtype
    rm ${pt1}${iid}/${folder}/${rid}.${spp}/temp1_*
}
sanity_check_pileupcaller_options() {
    if [[ "${mode}" == "randomHaploid" || "${mode}" == "majorityCall" ]]; then
        echo "Mode is valid: ${mode}"
    else
        echo "Error: Invalid mode value. It should be either 'randomHaploid' or 'majorityCall'."
        exit 1
    fi
    if [ -z "${bedf}" ] || [ ! -f "${bedf}" ]; then
        echo "ERROR: bed file is missing or does not exist." >&2
        exit 1
    fi
}
sanity_check_popfile() {
    if [ -f ${popfile} ]; then
        pop=$(awk -v rid="${rid}" '$1 == rid {print $2}' ${popfile} )
        if [ -z "${pop}" ]; then
            echo "Warning: Population for rid=${rid} not found in ${popfile}. Using ${rid} as population."
            pop=${rid}  # Default to rid if population is not found
        else
            echo "${pop} will be given for population."
        fi
    else
        pop=${rid}
        echo "Warning: no pop file given, pop will be same with ${rid}"
    fi
}
set_input_output_based_on_mask_and_library() {
    bamprefix=${rid}.${spp}.${workprefix}

    if [[ ${nmask} -eq 0 ]]; then
        echo "Masking is 0"
        outprefix=${rid}.${spp}
    elif [[ "${libtype}" == "non-UDG_NEB" ]]; then
        echo "Masking is not 0 and nmask is ${nmask}, only left-mask due to NEB treatment."
        outprefix=${rid}.${spp}.lm${nmask}
        bamprefix=${bamprefix}.lm${nmask}
    else
        echo "Masking is not 0 and nmask is ${nmask}"
        outprefix=${rid}.${spp}.m${nmask}
        bamprefix=${bamprefix}.m${nmask}
    fi
}
get_sample_sex_based_on_coverage() {
    sex=$(cat  ${pt1}${iid}/coverage/${rid}.${spp}/${rid}.${spp}.txt  |  awk  '{if  ($3/($2+0.000001)  >  0.7)
print "F";
        else if ($3/($2+0.000001) < 0.3) print "U"; else print "M"}')
    if [[ -z "$sex" ]]; then
        echo     "Error:     The     sex     variable     is     empty,     please     check     the     file
${pt1}${iid}/coverage/${rid}.${spp}/${rid}.${spp}.txt "
        exit 1  # Exit with a non-zero status indicating an error
    fi
    printf "sample sex is ${sex}\n"
}


run_pileup_on_bam() {
    local inputbam="${1}"
    local tempfile="${2}"
    local output="${3}"
    samtools mpileup -B -R -q30 -Q30 -l ${posf} -f ${ref} ${inputbam} > "${tempfile}" && \
    cut -f 1 "${tempfile}" | sed 's/X/23/g' | sed 's/Y/24/g' | paste - "${tempfile}" |\
    cut -f 1,3- > "${output}.pileup" && \
    rm "${tempfile}"
```

```
}

run_pileupcaller_based_on_libtype() {
    printf "Starting samtools pileup\n"
    pileuppref=${pt1}${iid}/${folder}/${rid}.${spp}/${rid}.${spp}

    bam1=${pt1}${iid}/BAM/${rid}.${spp}/${rid}.${spp}.${workprefix}.bam

    run_pileup_on_bam "${bam1}" "${tn1}_1" "${pileuppref}" &
    if [[ ${libtype} == "non-UDG_NEB" ]]; then

        bam2=${pt1}${iid}/BAM/${rid}.${spp}/${bamprefix}.ps.bam
        run_pileup_on_bam "${bam2}" "${tn1}_2" "${pileuppref}.lm${nmask}.ps" &
        bam3=${pt1}${iid}/BAM/${rid}.${spp}/${bamprefix}.ns.bam
        run_pileup_on_bam "${bam3}" "${tn1}_3" "${pileuppref}.lm${nmask}.ns" &
        wait
        printf "Running pileupcaller for ${libtype} library\n"
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
         -e "${tn1}"_1 < ${pileuppref}.pileup &
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
        -e "${tn1}"_2 < ${pileuppref}.lm${nmask}.ps.pileup &
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
        -e "${tn1}"_3 < ${pileuppref}.lm${nmask}.ns.pileup &
        wait
        printf "Use masked positive strand for C <-> T, masked negative strand for G <-> A, and all other
SNPs for unmasked file for library ${libtype}\n"
        paste ${snpf1} ${tn1}_1.geno ${tn1}_2.geno ${tn1}_3.geno | \
        awk '{if ($5$6 == "CT" || $5$6 == "TC") print $8; else if ($5$6 == "GA" || $5$6 == "AG") print $9;
         else print $7}' > ${of1}.geno
    elif [[ ${libtype} == "ssLib" ]]; then
        wait
        printf "Running pileupcaller for ${libtype} library\n"
        pileupCaller --${mode} --singleStrandMode --sampleNames "${outprefix}" -f "${snpf1}" \
        -e "${tn1}"_1 < ${pileuppref}.pileup &
        wait
        printf "Used --singlestrandmode in pileupCaller for ${libtype}\n"
        cp ${tn1}_1.geno ${of1}.geno
    elif [[ ${nmask} -gt 0 ]]; then
        bam2=${pt1}${iid}/BAM/${rid}.${spp}/${bamprefix}.bam
        run_pileup_on_bam "${bam2}" "${tn1}_2" "${pileuppref}.m${nmask}" &
        wait
        printf "Running pileupcaller for ${libtype} library\n"
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
        -e "${tn1}"_1 < ${pileuppref}.pileup &
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
        -e "${tn1}"_2 < ${pileuppref}.m${nmask}.pileup &
        wait
        printf "Masked for transitions, unmasked for transversions for library type ${libtype}\n"
        paste ${snpf1} ${tn1}_1.geno ${tn1}_2.geno | \
        awk '{if ($5$6 == "CT" || $5$6 == "TC"|| $5$6 == "GA" || $5$6 == "AG") print $8; \
       else print $7}' > ${of1}.geno




    else
        wait
        printf "Running pileupcaller for ${libtype} library\n"
        pileupCaller --${mode} --sampleNames "${outprefix}" -f "${snpf1}" \
      -e "${tn1}"_1 < ${pileuppref}.pileup &
        wait
        if [[ ${libtype} == "non-UDG" ]]; then
            echo "Using only transversion SNPs from geno for ${libtype}"
            paste ${snpf1} ${tn1}_1.geno | \
            awk '{if ($5$6 == "CT" || $5$6 == "TC"|| $5$6 == "GA" || $5$6 == "AG") print 9; \
            else print $7}' > ${of1}.geno
```

```
        elif [[ ${libtype} == "full-UDG" ]]; then
            echo "Using all SNPs from geno for ${libtype}"
            cp ${tn1}_1.geno ${of1}.geno
        else
            echo "Warning: please check library type ${libtype}, Using all SNPs from geno."
            cp ${tn1}_1.geno ${of1}.geno
        fi
    fi
}

main
```

코드 16. 위에 저장한 코드들을 다음과 같이 순차적으로 돌려준다. autosnpf1는 1240K SNP에 대해 eigenstrat 포맷의 .snp 형식의 파일이며, autoposf는 autosnpf의 2열(염색체)과 4열(SNP 위치)로 이루어진 두 열로 된 파일이다.

```
bedf="pos/to/bed"
maskf="pos/to/masklist"
popfile="pos/to/poplist"
autosnpf1="/home/References/Human/SNPCapBEDs/1240K.snp"
autoposf="/home/References/Human/SNPCapBEDs/1240K.pos.list.txt"

scf1=./wrap_extract_1240K.sh
scf2=./wrap_masksplitbam.sh
scf3=./wrap_pileupcaller_eigenstrat.sh

"${scf1} ${pt1} ${listf} ${inum} ${reff} ${bedf}"
"${scf2} ${pt1} ${listf} ${inum} ${maskf}"
"${scf3}  ${pt1}  ${listf}  ${inum}  ${reff}  ${bedf}  ${maskf}  ${popfile}  randomHaploid  genotypes
1240K.rmdup.q30 ${autosnpf1} ${autoposf} NA"
```

하플로그룹 결정

코드 17. wrap_ycall.sh로 저장해준다.

```
#!/bin/bash
pt1=$1
listf=$2
inum=$3
reff=$4
maskf=$5
folder=$6
workprefix=$7
filtersex=$8
source /path/to/main_utils.sh  # [코드 3]
py="/opt/ohpc/pub/apps/yhaplo/2016-v1/callHaplogroups.py"
read_listfile_into_variables && check_librarytype
nmask=$(awk -v rid="${rid}" '$1 == rid {print $2}' ${maskf} ); nmask=${nmask:-0}
bamprefix=${rid}.${spp}.${workprefix}
sex=$(cat ${pt1}${iid}/coverage/${rid}.${spp}/${rid}.${spp}.txt \
     | awk '{if ($3/($2+0.000001) > 0.7) print "F";
    else if ($3/($2+0.000001) < 0.3) print "U"; else print "M"}')

if [[ "${filtersex}" == ${sex}  ]]; then
    echo "Sample has same sex with filter sex, aborting Yhaplo call."
    exit 0
fi
if [[ ${nmask} -eq 0 ]]; then
    echo "Masking is 0"
    outprefix=${rid}.${spp}
elif [[ "${libtype}" == "non-UDG_NEB" ]]; then
    echo "Masking is not 0 and nmask is ${nmask}, only left-mask due to NEB treatment."
    outprefix=${rid}.${spp}.lm${nmask}
```

```
else
    echo "Masking is not 0 and nmask is ${nmask}"
    outprefix=${rid}.${spp}.m${nmask}
fi
echo "Eigenstrat sample ID is ${outprefix}"

of1=${pt1}${iid}/${folder}/${rid}.${spp}/${outprefix}

if [ -f "${of1}.geno" ] && [ -f "${of1}.ind" ]; then
    echo "Files with prefix ${of1} are used."
else
    echo "No existing '${of1}.geno' or '${of1}.ind' files in ${folder}, exiting."
    exit 1
fi
## Change EIGENSTRAT format data into yHaplo input format
./EIGENSTRAT_to_yhaplo.py ${of1}

echo "Running yhaplo..."
python2 ${py} --ancStopThresh 10 -i ${of1}.genos.txt -o ${pt1}${iid}/${folder}/${rid}.${spp}/${outprefix}
-c -hp -ds -dsd -as -asd
cat ${pt1}${iid}/${folder}/${rid}.${spp}/${outprefix}/haplogroups.${outprefix}.txt | sort -k1,1 | awk
'{OFS="\t"} {if ($2 == $3) print $1,$4"("$2")"; else print $1,$4"("$2","$3")"}' > ${of1}.result.txt
echo "yhaplo ${version} results appended to ${of1}.result.txt"
```

코드 18. EIGENSTRAT_to_yhaplo.py로 저장한다.

```
#! /usr/bin/env python
import re, sys, os, gzip
arg = sys.argv
## Read in arguments
inFile = arg[1]
def column(mat, i):
    return [row[i] for row in mat]
r1 = os.getcwd() + "/"; os.chdir(r1)
## Import .geno, .snp and .ind files
maps = [line.strip().split() for line in open(inFile + ".snp", "r").readlines()]
ids = [line.strip().split()[0] for line in open(inFile + ".ind", "r").readlines()]
genos = [line.strip() for line in open(inFile + ".geno", "r").readlines()]
ninds = len(ids)
nsnps = len(genos)
a_refs = [val for val in column(maps, -2)]
a_alts = [val for val in column(maps, -1)]
hv = "ID\t" + '\t'.join(column(maps, 3))
F1 = open(inFile + ".genos.txt", "w")
F1.writelines(hv + "\n")
for i in range(ninds):
    tgv = [val for val in column(genos, i)]
    tgv2 = [a_refs[j] if v == "2" else a_alts[j] if v == "0" else "." for j,v in enumerate(tgv)]
    F1.writelines(ids[i] + "\t" + '\t'.join(tgv2) + "\n")

F1.close()
```

코드 19. wrap_mtcall.sh로 저장해준다.

```
#!/bin/bash
pt1=$1
listf=$2
inum=$3
reff=$4
source /path/to/main_utils.sh  # [코드 3]
read_listfile_into_variables && check_librarytype

MTref1="/home/References/Human/hg19_MT.fasta"
of1=${rid}.${spp}.circmapper
bam1=${pt1}${iid}/circularMT/${rid}.${spp}/${of1}.rmdup.q30.MD.small.bam
```

```
mkdir -p ${pt1}${iid}/haplogrep2/${rid}.${spp}/

faout1=${pt1}${iid}/haplogrep2/${rid}.${spp}/${rid}.${spp}.circmapper.MTendo.schmutzi.nolen.fasta
faout2=${pt1}${iid}/haplogrep2/${rid}.${spp}/${rid}.${spp}.circmapper.MTcont.schmutzi.nolen.fasta

>${faout1}
>${faout2}

logf1=${pt1}${iid}/schmutzi/${rid}.${spp}/${rid}.${spp}.circmapper_nolen_final_endo.log ## log file for
endogenous seq
logf2=${pt1}${iid}/schmutzi/${rid}.${spp}/${rid}.${spp}.circmapper_nolen_final_cont.log ## log file for
contaminant seq

if [ -f ${logf1} ] && [ -f ${logf2} ]; then
    echo "Contamination estimate exists."
else
    echo "Contamination estimate does not exist, running endoCaller."
    faout1=${pt1}${iid}/haplogrep2/${rid}.${spp}/${rid}.${spp}.circmapper.MTnoest.schmutzi.nolen.fasta
    >${faout1}
    faout2=""
    logf1=${pt1}${iid}/schmutzi/${rid}.${spp}/${rid}.${spp}.circmapper_noest_endo.log
    logf2=""
    endoCaller -cont 0 -seq ${pt1}${iid}/schmutzi/${rid}.${spp}/${rid}.${spp}.circmapper_noest_endo.fa -
log ${logf1} ${MTref1} ${bam1}
fi

for Q in 10 20 30; do hv=">"${rid}".q"${Q}
    if [ -f "$logf1" ]; then
        echo "reading ${logf1}... "
        echo $(Jones et al.) >> ${faout1}; /opt/ohpc/pub/apps/schmutzi/src/log2fasta -q ${Q} ${logf1} \
       | tail -n +2 >> ${faout1}
    fi
    if [ -f "$logf2" ]; then
        echo "reading ${logf2}... "
        echo $(Jones et al.) >> ${faout2}; /opt/ohpc/pub/apps/schmutzi/src/log2fasta -q ${Q} ${logf2} \
       | tail -n +2 >> ${faout2}
    fi
done

for fa in ${faout1} ${faout2}; do
    if [ -f "$fa" ]; then
        echo "deciding Haplogroup of ${fa}..."
        java -jar /opt/ohpc/pub/apps/haplogrep/haplogrep-2.1.20.jar --in ${fa} --format fasta --out
${fa}_1
        cat ${fa}_1 | sed s/'"'/''/g | awk '{OFS="\t"} {print $1,$2,$3,$4,$5}' > ${fa}.call.txt
    fi
done
```

코드 20. 위에 저장한 코드들을 다음과 같이 순차적으로 돌려준다.

```
bedf="path/to/bed"
maskf="path/to/maskfile"
popfile="path/to/popfile"
ysnpf1_2016="/opt/ohpc/pub/apps/yhaplo/2016-v1/input/ISOGG_chrY_cleanup_170802.snp"
yposf_2016="/opt/ohpc/pub/apps/yhaplo/2016-v1/input/ISOGG_chrY_cleanup_170802_hs37d5.list"

scf1=./wrap_pileupcaller_eigenstrat.sh
scf2=./wrap_ycall.sh
scf3=./wrap_mtcall.sh

"${scf1} ${pt1} ${listf} ${inum} ${reff} ${bedf} ${maskf} ${popfile} majorityCall Yhap_2016
1240K.rmdup.q30 ${ysnpf1_2016} ${yposf_2016} F"
"${scf2} ${pt1} ${listf} ${inum} ${reff} ${maskf} Yhap_2016 1240K.rmdup.q30 F"
"${scf3} ${pt1} ${listf} ${inum} ${reff}"
```

PMR 계산

코드 21. Calculate_PMR_matrix.R로 저장한다.

```
library(argparser, quietly=TRUE)

# Create a parser
p <- arg_parser("Calculate PMR using matrix multiplicaiton")

# Add command line arguments
p <- add_argument(p, "--geno", help="genotype file name, genotype should be delimited with space")
p <- add_argument(p, "--ind", help="ind file name, individual should match genotype")
p <- add_argument(p, "--outfile", help="output file name")
p <- add_argument(p, "--pseudohaploid", help="pseudohaploid (TRUE) or diploid (FALSE)", default=F)

# Parse the command line arguments
argv <- parse_args(p)

# Input files
gfile=argv$geno
ifile=argv$ind
of1=argv$outfile
pseudohaploid=argv$pseudohaploid

# Import library
library(data.table)

geno <- as.matrix(fread(gfile,header=F,data.table=FALSE))
ind <- fread(ifile,header=F)
colnames(ind) <- c("iid","sex","pid")

# memory size of matrix
print(paste0("The size of genotype matrix : ",as.integer(object.size(geno)/1e6)," Mb") )

# convert genotype matrix
tic("Matrix multiplication")
if (pseudohaploid) {
    # first, calculate nmismatch (shifted later)
    geno <- geno - 1
    geno[geno %% 8 == 0] <- 0 # 9-1 as missing
    count_matrix <- t(geno) %*% geno
    # second, calculate nsnps by each individual pairs
    geno[geno == -1] <- 1
    nsnp_matrix <- t(geno) %*% geno


   # Third, combine it to calculate pmr
```

```
    pmr_matrix <- (-(1/2)*(count_matrix) + (1/2)*(nsnp_matrix))/(nsnp_matrix)
    nmis_matrix <- -(1/2)*(count_matrix) + (1/2)*(nsnp_matrix)

} else  {
    # first, calculate nmismatch (shifted later)
    geno_c <- geno - 1
    geno_c[geno_c %% 8 == 0] <- 0 # 9-1 as missing

    count_matrix <- t(geno_c) %*% geno_c

    # second, calculate nsnps by each individual pairs
    geno[geno == 0|geno == 2] <- 1
    geno[geno == 9] <- 0

    nsnp_matrix <- t(geno) %*% geno

    # Third, combine it to calculate pmr
    pmr_matrix <- (-(1/2)*(count_matrix) + (1/2)*(nsnp_matrix))/(nsnp_matrix)
    nmis_matrix <- -(1/2)*(count_matrix) + (1/2)*(nsnp_matrix)

}

# convert to data table
iid1_vec <- c(); iid2_vec <- c(); pid1_vec <- c(); pid2_vec <- c()
nsnp_vec <- c(); nmis_vec <- c(); pmis_vec <- c()

for (i in 1:(nrow(ind)-1))  {
    for (j in (i+1):nrow(ind))  {
        # unique indivs pairs
        iid1 <- as.character(ind$iid[i]) ; pid1 <- as.character(ind$pid[i])
        iid2 <- as.character(ind$iid[j]) ; pid2 <- as.character(ind$pid[j])
        # add to vector
        iid1_vec <- c(iid1_vec,iid1); iid2_vec <- c(iid2_vec,iid2); pid1_vec <- c(pid1_vec,pid1)
        pid2_vec <- c(pid2_vec,pid2)
        nsnp_vec <- c(nsnp_vec,nsnp_matrix[i,j]); nmis_vec <- c(nmis_vec,nmis_matrix[i,j]);
        pmis_vec <- c(pmis_vec,pmr_matrix[i,j])
    }
}
df <- data.frame(cbind(iid1_vec,pid1_vec,iid2_vec,pid2_vec,nsnp_vec,nmis_vec,pmis_vec))
colnames(df) <- c("IID1","PID1","IID2","PID2","nsnp","nmismatch","pmismatch")

df[df$pmismatch==NaN,6] <- NA
df[df$pmismatch==NaN,7] <- NA

write.table(df, file=of1, quote=F, sep="\t", row.names=F)
```

코드 22. 다음 코드로 실행을 한다.

```
## Check pairwise mismatch rate.
of1="/eigenstrat/prefix"
outputname="name_of_output"

cat ${of1}.geno |  sed 's/\(.\)/\1 /g' | sed 's/ $//' > ${outputname}.spacegeno

Rscript ./Calculate_PMR_matrix.R -g ${outputname}.spacegeno -i ${of1}.ind -o ${outputname}_pmrres.txt"
```

사사

이 논문은 국립문화유산연구원(구 국립문화재연구원) 연구용역과제(계약번호 20232600514-00)의 지원을 받아 작성하였음.

참고문헌

Allentoft ME, Sikora M, Sjögren K-G, Rasmussen S, Rasmussen M, Stenderup J, Damgaard PB, Schroeder H, Ahlström T, Vinner L et al. 2015. Population genomics of bronze age eurasia. Nature. 522(7555):167-172.

Bos KI, Harkins KM, Herbig A, Coscolla M, Weber N, Comas I, Forrest SA, Bryant JM, Harris SR, Schuenemann VJ et al. 2014. Pre-columbian mycobacterial genomes reveal seals as a source of new world human tuberculosis. Nature. 514(7523):494-497.

Bos KI, Schuenemann VJ, Golding GB, Burbano HA, Waglechner N, Coombes BK, McPhee JB, DeWitte SN, Meyer M, Schmedes S et al. 2011. A draft genome of yersinia pestis from victims of the black death. Nature. 478(7370):506-510.

Chintalapati M, Patterson N, Moorjani P. 2022. The spatiotemporal patterns of major human admixture events during the european holocene. eLife. 11:e77625.

da Fonseca RR, Smith BD, Wales N, Cappellini E, Skoglund P, Fumagalli M, Samaniego JA, Carøe C, Ávila-Arcos MC, Hufnagel DE et al. 2015. The origin and evolution of maize in the southwestern united states. Nature Plants. 1(1):14003.

Dabney J, Meyer M. 2019. Extraction of highly degraded DNA from ancient bones and teeth. In: Shapiro B, Barlow A, Heintzman PD, Hofreiter M, Paijmans JLA, Soares AER, editors. Ancient DNA: Methods and protocols. New York, NY: Springer New York. p. 25-29.

Dabney J, Meyer M, Pääbo S. 2013. Ancient DNA damage. Cold Spring Harbor Perspectives in Biology. 5(7).

Daly KG, Maisano Delser P, Mullin VE, Scheu A, Mattiangeli V, Teasdale MD, Hare AJ, Burger J, Verdugo MP, Collins MJ et al. 2018. Ancient goat genomes reveal mosaic domestication in the fertile crescent. Science. 361(6397):85-88.

Damgaard PdB, Marchi N, Rasmussen S, Peyrot M, Renaud G, Korneliussen T, Moreno-Mayar JV, Pedersen MW, Goldberg A, Usmanova E et al. 2018. 137 ancient human genomes from across the eurasian steppes. Nature. 557(7705):369-374.

Fu Q, Hajdinjak M, Moldovan OT, Constantin S, Mallick S, Skoglund P, Patterson N, Rohland N, Lazaridis I, Nickel B et al. 2015. An early modern human from romania with a recent neanderthal ancestor. Nature. 524(7564):216-219.

Gaunitz C, Fages A, Hanghøj K, Albrechtsen A, Khan N, Schubert M, Seguin-Orlando A, Owens IJ, Felkel S, Bignon-Lau O et al. 2018. Ancient genomes revisit the ancestry of domestic and przewalski's horses. Science. 360(6384):111-114.

Green RE, Krause J, Briggs AW, Maricic T, Stenzel U, Kircher M, Patterson N, Li H, Zhai W, Fritz MH-Y et al. 2010. A draft sequence of the neandertal genome. Science. 328(5979):710-722.

Haak W, Lazaridis I, Patterson N, Rohland N, Mallick S, Llamas B, Brandt G, Nordenfelt S, Harney E, Stewardson K et al. 2015. Massive migration from the steppe was a source for indo-european languages in europe. Nature. 522(7555):207-211.

Han E, Sinsheimer JS, Novembre J. 2013. Characterizing bias in population genetic inferences from low-coverage sequencing data. Molecular Biology and Evolution. 31(3):723-735.

Hellenthal G, Busby GBJ, Band G, Wilson JF, Capelli C, Falush D, Myers S. 2014. A genetic atlas of human admixture history. Science. 343(6172):747-751.

Jeong C, Wang K, Wilkin S, Taylor WTT, Miller BK, Bemmann JH, Stahl R, Chiovelli C, Knolle F, Ulziibayar S et al. 2020. A dynamic 6,000-year genetic history of eurasia's eastern steppe. Cell. 183(4):890-904.e829.

Jones ER, Gonzalez-Fortes G, Connell S, Siska V, Eriksson A, Martiniano R, McLaughlin RL, Gallego Llorente M, Cassidy LM, Gamba C et al. 2015. Upper palaeolithic genomes reveal deep roots of modern eurasians. Nature Communications. 6(1):8912.

Jónsson H, Ginolhac A, Schubert M, Johnson PLF, Orlando L. 2013. Mapdamage2.0: Fast approximate bayesian estimates of ancient DNA damage parameters. Bioinformatics. 29(13):1682-1684.

Kennett DJ, Plog S, George RJ, Culleton BJ, Watson AS, Skoglund P, Rohland N, Mallick S, Stewardson K, Kistler L et al. 2017. Archaeogenomic evidence reveals prehistoric matrilineal dynasty. Nature Communications. 8(1):14115.

Kim K, Kim D, Hanotte O, Lee C, Kim H, Jeong C. 2023. Inference of admixture origins in indigenous african cattle. Molecular Biology and Evolution. 40(12).

Korneliussen TS, Albrechtsen A, Nielsen R. 2014. Angsd: Analysis of next generation sequencing data. BMC Bioinformatics. 15(1):356.

Lawson DJ, Hellenthal G, Myers S, Falush D. 2012. Inference of population structure using dense haplotype data. PLOS Genetics. 8(1):e1002453.

Li H, Durbin R. 2009. Fast and accurate short read alignment with burrows–wheeler transform. Bioinformatics. 25(14):1754-1760.

Li H, Durbin R. 2011. Inference of human population history from individual whole-genome sequences. Nature. 475(7357):493-496.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP. 2009. The sequence alignment/map format and samtools. Bioinformatics. 25(16):2078-2079.

Loh P-R, Lipson M, Patterson N, Moorjani P, Pickrell JK, Reich D, Berger B. 2013. Inferring admixture histories of human populations using linkage disequilibrium. Genetics. 193(4):1233-1254.

Maddison WP. 1997. Gene trees in species trees. Systematic Biology. 46(3):523-536.

Mallick S, Li H, Lipson M, Mathieson I, Gymrek M, Racimo F, Zhao M, Chennagiri N, Nordenfelt S, Tandon A et al. 2016. The simons genome diversity project: 300 genomes from 142 diverse populations. Nature. 538(7624):201-206.

Mallick S, Micco A, Mah M, Ringbauer H, Lazaridis I, Olalde I, Patterson N, Reich D. 2024. The allen ancient DNA resource (aadr) a curated compendium of ancient human genomes. Scientific Data. 11(1):182.

Mascher M, Schuenemann VJ, Davidovich U, Marom N, Himmelbach A, Hübner S, Korol A, David M, Reiter E, Riehl S et al. 2016. Genomic analysis of 6,000-year-old cultivated grain illuminates the domestication history of barley. Nature Genetics. 48(9):1089-1093.

Mathieson I, Lazaridis I, Rohland N, Mallick S, Patterson N, Roodenberg SA, Harney E, Stewardson K, Fernandes D, Novak M et al. 2015. Genome-wide patterns of selection in 230 ancient eurasians. Nature. 528(7583):499-503.

McColl H, Racimo F, Vinner L, Demeter F, Gakuhari T, Moreno-Mayar JV, van Driem G, Gram Wilken U, Seguin-Orlando A, de la Fuente Castro C et al. 2018. The prehistoric peopling of southeast asia. Science. 361(6397):88-92.

Meyer M, Arsuaga J-L, de Filippo C, Nagel S, Aximu-Petri A, Nickel B, Martínez I, Gracia A, de Castro JMB, Carbonell E et al. 2016. Nuclear DNA sequences from the middle pleistocene sima de los huesos hominins. Nature. 531(7595):504-507.

Meyer M, Kircher M. 2010. Illumina sequencing library preparation for highly multiplexed target capture and sequencing. Cold Spring Harbor Protocols. 2010(6):pdb.prot5448.

Miller W, Drautz DI, Ratan A, Pusey B, Qi J, Lesk AM, Tomsho LP, Packard MD, Zhao F, Sher A et al. 2008. Sequencing the nuclear genome of the extinct woolly mammoth. Nature. 456(7220):387-390.

Mittnik A, Wang C-C, Pfrengle S, Daubaras M, Zariņa G, Hallgren F, Allmäe R, Khartanovich V, Moiseyev V, Tõrv M et al. 2018. The genetic prehistory of the baltic sea region. Nature Communications. 9(1):442.

Moreno-Mayar JV, Potter BA, Vinner L, Steinrücken M, Rasmussen S, Terhorst J, Kamm JA, Albrechtsen A, Malaspinas A-S, Sikora M et al. 2018. Terminal pleistocene alaskan genome reveals first founding population of native americans. Nature. 553(7687):203-207.

Palkopoulou E, Mallick S, Skoglund P, Enk J, Rohland N, Li H, Omrak A, Vartanyan S, Poinar H, Götherström A et al. 2015. Complete genomes reveal signatures of demographic and genetic declines in the woolly mammoth. Current Biology. 25(10):1395-1400.

Patterson N, Moorjani P, Luo Y, Mallick S, Rohland N, Zhan Y, Genschoreck T, Webster T, Reich D. 2012. Ancient admixture in human history. Genetics. 192(3):1065-1093.

Peltzer A, Jäger G, Herbig A, Seitz A, Kniep C, Krause J, Nieselt K. 2016. Eager: Efficient ancient genome reconstruction. Genome Biology. 17(1):60.

Popli D, Peyrégne S, Peter BM. 2023. Kin: A method to infer relatedness from low-coverage ancient DNA. Genome Biology. 24(1):10.

Poznik GD. 2016. Identifying y-chromosome haplogroups in arbitrarily large samples of sequenced or genotyped men. bioRxiv.088716.

Prüfer K, Stenzel U, Hofreiter M, Pääbo S, Kelso J, Green RE. 2010. Computational challenges in the analysis of ancient DNA. Genome Biology. 11(5):R47.

Raghavan M, DeGiorgio M, Albrechtsen A, Moltke I, Skoglund P, Korneliussen TS, Grønnow B, Appelt M, Gulløv HC, Friesen TM et al. 2014. The genetic prehistory of the new world arctic. Science. 345(6200):1255832.

Rasmussen M, Guo X, Wang Y, Lohmueller KE, Rasmussen S, Albrechtsen A, Skotte L, Lindgreen S, Metspalu M, Jombart T et al. 2011. An aboriginal australian genome reveals separate human dispersals into asia. Science. 334(6052):94-98.

Rasmussen M, Li Y, Lindgreen S, Pedersen JS, Albrechtsen A, Moltke I, Metspalu M, Metspalu E, Kivisild T, Gupta R et al. 2010. Ancient human genome sequence of an extinct palaeo-eskimo. Nature. 463(7282):757-762.

Renaud G, Slon V, Duggan AT, Kelso J. 2015. Schmutzi: Estimation of contamination and endogenous mitochondrial consensus calling for ancient DNA. Genome Biology. 16(1):224.

Ringbauer H, Huang Y, Akbari A, Mallick S, Olalde I, Patterson N, Reich D. 2024. Accurate detection of identity-by-descent segments in human ancient DNA. Nature Genetics. 56(1):143-151.

Rohland N, Glocke I, Aximu-Petri A, Meyer M. 2018. Extraction of highly degraded DNA from ancient bones, teeth and sediments for high-throughput sequencing. Nature Protocols. 13(11):2447-2461.

Sawyer S, Krause J, Guschanski K, Savolainen V, Pääbo S. 2012. Temporal patterns of nucleotide misincorporations and DNA fragmentation in ancient DNA. PLOS ONE. 7(3):e34131.

Scally A, Dutheil JY, Hillier LW, Jordan GE, Goodhead I, Herrero J, Hobolth A, Lappalainen T, Mailund T, Marques-Bonet T et al. 2012. Insights into hominid evolution from the gorilla genome sequence. Nature. 483(7388):169-175.

Schiffels S, Durbin R. 2014. Inferring human population size and separation history from multiple genome sequences. Nature Genetics. 46(8):919-925.

Schubert M, Lindgreen S, Orlando L. 2016. Adapterremoval v2: Rapid adapter trimming, identification, and read merging. BMC Research Notes. 9(1):88.

Sikora M, Pitulko VV, Sousa VC, Allentoft ME, Vinner L, Rasmussen S, Margaryan A, de Barros Damgaard P, de la Fuente C, Renaud G et al. 2019. The population history of northeastern siberia since the pleistocene. Nature. 570(7760):182-188.

Skoglund P, Thompson JC, Prendergast ME, Mittnik A, Sirak K, Hajdinjak M, Salie T, Rohland N, Mallick S, Peltzer A et al. 2017. Reconstructing prehistoric african population structure. Cell. 171(1):59-71.e21.

Skourtanioti E, Ringbauer H, Gnecchi Ruscone GA, Bianco RA, Burri M, Freund C, Furtwängler A, Gomes Martins NF, Knolle F, Neumann GU et al. 2023. Ancient DNA reveals admixture history and endogamy in the prehistoric aegean. Nature Ecology & Evolution. 7(2):290-303.

Soubrier J, Gower G, Chen K, Richards SM, Llamas B, Mitchell KJ, Ho SYW, Kosintsev P, Lee MSY, Baryshnikov G et al. 2016. Early cave art and ancient DNA record the origin of european bison. Nature Communications. 7(1):13158.

Verdugo MP, Mullin VE, Scheu A, Mattiangeli V, Daly KG, Maisano Delser P, Hare AJ, Burger J, Collins MJ, Kehati R et al. 2019.

Ancient cattle genomics, origins, and rapid turnover in the fertile crescent. Science. 365(6449):173-176.
Wang M-S, Murray GGR, Mann D, Groves P, Vershinina AO, Supple MA, Kapp JD, Corbett-Detig R, Crump SE, Stirling I et al. 2022. A polar bear paleogenome reveals extensive ancient gene flow from polar bears into brown bears. Nature Ecology & Evolution. 6(7):936-944.
Waples RK, Albrechtsen A, Moltke I. 2019. Allele frequency-free inference of close familial relationships from genotypes or low-depth sequencing data. Molecular Ecology. 28(1):35-48.
Weissensteiner H, Pacher D, Kloss-Brandstätter A, Forer L, Specht G, Bandelt H-J, Kronenberg F, Salas A, Schönherr S. 2016. Haplogrep 2: Mitochondrial haplogroup classification in the era of high-throughput sequencing. Nucleic Acids Research. 44(W1):W58-W63.
Yang MA, Fan X, Sun B, Chen C, Lang J, Ko Y-C, Tsang C-h, Chiu H, Wang T, Bao Q et al. 2020. Ancient DNA indicates human population shifts and admixture in northern and southern china. Science. 369(6501):282-288.

영문초록

Title: A guide to processing, quality control, and kinship estimation of genome-wide sequencing data from ancient samples

Abstract: Next-generation sequencing (NGS)-based analysis of ancient biological tissues from post-mortem samples has become increasingly popular and expanding in the field of evolutionary genetics due to its utility. However, analyses based on high-quality modern DNA extracts that do not consider the unique characteristics of ancient DNA are prone to biases, thus requiring extreme caution. In this paper, we present principles and practical applications of ancient DNA processing and quality control, using sequencing data obtained from the femur of 11 ancient individuals from the Shiveet Khairkhan site of Mongol as an example case. We describe the manipulation of NGS based ancient DNA analysis including preprocessing steps such as quality control, as well as methods for estimating kinship relationships among ancient individuals. By doing so, we aim to enhance the understanding of analysis strategies tailored to ancient DNA NGS data.

Authors: [1,2]Hyoungmin Moon, [3]Eun Min Cho, [3]Suyeon Kim, [3]Soyeong Kang, [1,2*]Choongwon Jeong

Affiliation: [1]School of Biological Sciences, Seoul National University, Seoul 08826, Republic of Korea

[2]Institute for Data Innovation in Science, Seoul National University, Seoul 08826, Republic of Korea

[3]National Research Institute of Cultural Heritage, Daejeon 34122, Republic of Korea

Corresponding author: *