# 出芽酵母单核苷酸突变特征

生信 2001 张子栋 梁国相

# 分工

- 张子栋
  - 代码实现
  - 项目展示
  - 实验报告
- 梁国相
  - 思路与模型建立
  - 代码校对
  - R 语言绘图

# 1. 基因序列和非基因序列突变率的差异

- 实验思路
  - 使用 Java 通过 MyBatis 操作数据完成数据处理和统计量计算
  - 通过 R 绘图

# 数据预处理

- 数据导入
- 通过 Navicat 导入向导导入数据
- 其中，saccharomyces_cerevisiae_R64-1-1_20110208.gff 文件仅提取第 1，3，4，5 列为有效信息

对象 | t_budding_yeast_snps @bio (local) - 表 | t_saccharomyces_cerevisiae @bio (lo...

开始事务 | 文本 | 筛选 | 排序 | 导入 | 导出

| chromosome_id | chromosome_pos | type |
|---|---|---|
| chrI | 20 | CACC |
| chrI | 22 | ACCC |
| chrI | 41 | ACCC |
| chrI | 63 | TCAC |
| chrI | 75 | CACC |
| chrI | 79 | ACAC |
| chrI | 81 | ACCT |
| chrI | 82 | CTCA |
| chrI | 84 | GAAC |
| chrI | 90 | ATAT |
| chrI | 96 | CTAC |
| chrI | 101 | GAGT |
| chrI | 103 | CTTT |
| chrI | 104 | ATTT |
| chrI | 109 | GCTT |
| chrI | 114 | TCCA |
| chrI | 115 | CACA |
| chrI | 119 | TCCT |
| chrI | 136 | GATC |
| chrI | 157 | CTCC |
| chrI | 169 | AGTC |
| chrI | 171 | CTCA |
| chrI | 177 | GCCA |

# 数据预处理

- 创建实体类
- 实体类 Information 用于存储序列信息
- 实体类 Mutation 用于存储变异信息

```java
/**
 * @author ZidongZh
 * @date 2022/3/3
 */
public class Mutation {
    private String chromosomeId;
    private Integer chromosomePos;
    private String type;

    /**
     * constructor
     *
     * @param chromosomeId   染色体号
    * @param chromosomePos 染色体位置
    * @param type           变异类型
    */
    public Mutation(
            String chromosomeId,
            Integer chromosomePos,
            String type) {
        this.chromosomeId = chromosomeId;
        this.chromosomePos = chromosomePos;
        this.type = type;
    }
```

```java
/**
 * @author ZidongZh
 * @date 2022/3/3
 */
public class Information {
    private String chromosomeId;
    private String type;
    private Integer startPos;
    private Integer endPos;
```

# 数据预处理

- 处理重叠基因序列

```java
/**
 * 合并重叠基因区间
 *
 * @param information 原始数据
 */
void connect(List<Information> information) {
    int i = 0;
    int j = 1;
    Information previous = null;
    Information next = null;
    while (j + 1 < information.size()) {
        previous = information.get(i);
        next = information.get(j);
        if (previous.getEndPos() >= next.getStartPos() && previous.getEndPos() <= next.getEndPos()) {
            previous.setEndPos(next.getEndPos());
            information.remove(j);
            information.set(i, previous);
        } else {
            i++;
            j++;
        }
    }
}
```

# 数据预处理

- 获取非基因序列

```java
/**
 * @param genes 基因信息
 * @return 非基因序列
 */
List<Information> getNonGenes(List<Information> genes, List<Information> chromosomes) {
    int pos = 0;

    List<Information> nonGenes = new ArrayList<>();
    Information nonGeneHead = new Information(chromosomes.get(0).getChromosomeId(), "nonGene", 1,
genes.get(0).getStartPos() - 1);
    nonGenes.add(nonGeneHead);
    for (int i = 0; i < chromosomes.size(); i++) {
        for (int j = 0; j < genes.size(); j++) {
            if (genes.get(j).getChromosomeId().equals(chromosomes.get(i).getChromosomeId()) &&
                    genes.get(j + 1).getChromosomeId().equals(chromosomes.get(i).getChromosomeId())) {
                Information nonGene = new Information(genes.get(j).getChromosomeId(), "nonGene",
genes.get(j).getEndPos() + 1, genes.get(j + 1).getStartPos() - 1);
                nonGenes.add(nonGene);
                pos = j + 2;
            }
        }
        if (null != genes.get(pos)) {
            Information nonGeneHead1 = new Information(genes.get(pos).getChromosomeId(), "nonGene", 1,
genes.get(pos).getStartPos() - 1);
            nonGenes.add(nonGeneHead1);
        }
    }
    return nonGenes;
}
```

# 数据预处理

- 获取非基因序列

```java
/**
 * @param genes 基因信息
 * @return 非基因序列
 */
List<Information> getNonGenes(List<Information> genes, List<Information> chromosomes) {
    int pos = 0;

    List<Information> nonGenes = new ArrayList<>();
    Information nonGeneHead = new Information(chromosomes.get(0).getChromosomeId(), "nonGene", 1,
genes.get(0).getStartPos() - 1);
    nonGenes.add(nonGeneHead);
    for (int i = 0; i < chromosomes.size(); i++) {
        for (int j = 0; j < genes.size(); j++) {
            if (genes.get(j).getChromosomeId().equals(chromosomes.get(i).getChromosomeId()) &&
                    genes.get(j + 1).getChromosomeId().equals(chromosomes.get(i).getChromosomeId())) {
                Information nonGene = new Information(genes.get(j).getChromosomeId(), "nonGene",
genes.get(j).getEndPos() + 1, genes.get(j + 1).getStartPos() - 1);
                nonGenes.add(nonGene);
                pos = j + 2;
            }
        }
        if (null != genes.get(pos)) {
            Information nonGeneHead1 = new Information(genes.get(pos).getChromosomeId(), "nonGene", 1,
genes.get(pos).getStartPos() - 1);
            nonGenes.add(nonGeneHead1);
        }
    }
    return nonGenes;
}
```

# 数据处理

- 计算统计数
- 在 Information 类中添加属性 mutationNum 和 mutationRate

```java
/**
 * @author ZidongZh
 * @date 2022/3/3
 */
public class Information {
    private String chromosomeId;
    private String type;
    private Integer startPos;
    private Integer endPos;
    private Integer mutationNum;
    private Double mutationRate;
```

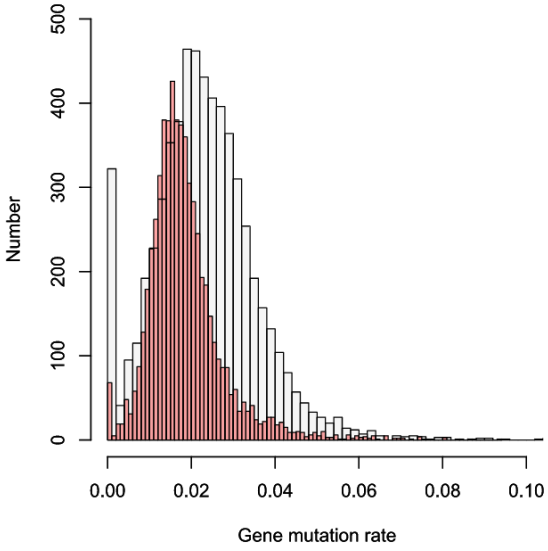
# 数据处理

- 计算统计数
- 计算变异率

```
//计算变异数与变异率
for (int i = 0; i < allMutation.size(); i++) {
    for (int j = 0; j < genes.size(); j++) {
        if (allMutation.get(i).getChromosomeId().equals(genes.get(j).getChromosomeId()) &&
allMutation.get(i).getChromosomePos() >= genes.get(j).getStartPos() &&
            allMutation.get(i).getChromosomePos() <= genes.get(j).getEndPos()) {
            genes.get(j).setMutationNum(genes.get(j).getMutationNum() + 1);
            genes.get(j).setMutationRate((double) genes.get(j).getMutationNum() / (double)
(genes.get(j).getEndPos() - genes.get(j).getStartPos() + 1));
        }
    }
    for (int j = 0; j < nonGenes.size(); j++) {
        if (allMutation.get(i).getChromosomeId().equals(nonGenes.get(j).getChromosomeId()) &&
allMutation.get(i).getChromosomePos() >= nonGenes.get(j).getStartPos() &&
            allMutation.get(i).getChromosomePos() <= nonGenes.get(j).getEndPos()) {
            nonGenes.get(j).setMutationNum(nonGenes.get(j).getMutationNum() + 1);
            nonGenes.get(j).setMutationRate((double) nonGenes.get(j).getMutationNum() / (double)
(nonGenes.get(j).getEndPos() - nonGenes.get(j).getStartPos() + 1));
        }
    }
}
```

# 数据可视化

- 使用 R　hist() 函数




```
> geneMutationRate = read.table(file.choose(), header = F, sep = ",")
> geneMutationRate = as.matrix(geneMutationRate)
> nonGeneMutationRate = read.table(file.choose(), header = F, sep = ",")
> nonGeneMutationRate = as.matrix(nonGeneMutationRate)
> hist(geneMutationRate, xlab="Gene mutation rate",ylab="Number",col=rgb(255, 0, 0, 90,
maxColorValue=255), breaks = 100, xlim = c(0.0,0.1), ylim = c(0,500), main = NULL)
> par(new = T)
> hist(nonGeneMutationRate, xlab="Gene mutation rate",ylab="Number",col=rgb(2, 0, 0, 10,
maxColorValue=255), breaks = 100, xlim = c(0.0,0.1), ylim = c(0,500), main = NULL)
```

# 假设检验

主要算法:

```
//t检验
double t = 0.0;
t = (geneMutRateMean - nonGeneMutRateMean) / (sqrt((((genes.size() - 1) * geneVariance) +
((nonGenes.size() - 1) * nonGeneVariance)) / (genes.size() + nonGenes.size() - 2)) * sqrt((1
/ (double) genes.size()) + (1 / (double) nonGenes.size()))));
System.out.println("t = " + t);
//u 检验
double u = 0.0;
u = (geneMutRateMean - nonGeneMutRateMean) / (sqrt((geneVariance / genes.size()) +
(nonGeneVariance / nonGenes.size()))));
System.out.println("u = " + u);
```

输出结果:

```
geneMutRateMean = 0.018604969719609406
geneVariance  = 8.597880264141603E-5
nonGeneMutRateMean = 0.023312667924136225
nonGeneVariance = 1.7675824048238737E-4
t = -22.612916116843113
u = -22.610982585720944
```

# 使用 R 内置 t.test()验证结果

```
> t.test(geneMutationRate,nonGeneMutationRate)

    Welch Two Sample t-test

data:  geneMutationRate and nonGeneMutationRate
t = -22.611, df = 10825, p-value < 2.2e-16
alternative hypothesis: true difference in means is
not equal to 0
95 percent confidence interval:
 -0.005115816 -0.004299580
sample estimates:
 mean of x  mean of y
0.01860497 0.02331267
```

结果与 Java 基本一致

# U 检验

1. 提出:
   - $H_0: \mu_1 = \mu_2$ , 出芽酵母基因区间突变率与非基因区间突变率没有显著差异
   - $H_A: \mu_1 \neq \mu_2$
2. 假定 $H_0$ 成立
3. 选取显著水平 $\alpha = 0.01$
4. 统计量:

$$s_{\overline{x_1}-\overline{x_2}} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

$$u = \frac{(\overline{x_1} - \overline{x_2})}{s_{\overline{x_1}-\overline{x_2}}}$$

   设计程序计算得出 `u = -22.610982585720944`

$$|u| > 2.58,\ P < 0.01$$

5. 推断: 在 0.01 显著水平上, 拒绝 $H_0$, 接受 $H_A$
   认为出芽酵母基因区间突变率与非基因区间突变率有显著差异

# T 检验

1. 提出:
   - $H_0: \mu_1 = \mu_2$ , 出芽酵母基因区间突变率与非基因区间突变率没有显著差异
   - $H_A: \mu_1 \neq \mu_2$
2. 假定 $H_0$ 成立
3. 选取显著水平 $\alpha = 0.01$
4. 统计量:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

   设计程序计算得出 `t = -22.612916116843113`

$$|t| > 2.576,\ P < 0.01$$

5. 推断: 在 0.01 显著水平上, 拒绝 $H_0$, 接受 $H_A$
   认为出芽酵母基因区间突变率与非基因区间突变率有显著差异

# 2.突变方向的差异性

- 修改实体类

```
public class Information {
    private String chromosomeId;
    private String type;
    private Integer startPos;
    private Integer endPos;
    private Integer mutationNum;
    private Double mutationRate;
    private Integer AGNum;
    private Integer ACNum;
    private Integer ATNum;
    private Integer GANum;
    private Integer GCNum;
    private Integer GTNum;
    private Integer CANum;
    private Integer CGNum;
    private Integer CTNum;
    private Integer TANum;
    private Integer TGNum;
    private Integer TCNum;
```

# 算法设计

- 获取基因突变方向及突变数

```
void classify(Information gene, Mutation mutation) {
    if (mutation.getChromosomeId().equals(gene.getChromosomeId()) &&
            mutation.getChromosomePos() >= gene.getStartPos() &&
            mutation.getChromosomePos() <= gene.getEndPos()) {
        if (mutation.getType().startsWith("AT")) {
            gene.setATNum(gene.getATNum() + 1);
        } else if (mutation.getType().startsWith("AG")) {
            gene.setAGNum(gene.getAGNum() + 1);
        } else if (mutation.getType().startsWith("AC")) {
            gene.setACNum(gene.getACNum() + 1);
        } else if (mutation.getType().startsWith("TA")) {
            gene.setTANum(gene.getTANum() + 1);
        } else if (mutation.getType().startsWith("TG")) {
            gene.setTGNum(gene.getTGNum() + 1);
        } else if (mutation.getType().startsWith("TC")) {
            gene.setTCNum(gene.getTCNum() + 1);
        } else if (mutation.getType().startsWith("GA")) {
            gene.setGANum(gene.getGANum() + 1);
        } else if (mutation.getType().startsWith("GT")) {
            gene.setGTNum(gene.getGTNum() + 1);
        } else if (mutation.getType().startsWith("GC")) {
            gene.setGCNum(gene.getGCNum() + 1);
        } else if (mutation.getType().startsWith("CA")) {
            gene.setCANum(gene.getCANum() + 1);
        } else if (mutation.getType().startsWith("CT")) {
            gene.setCTNum(gene.getCTNum() + 1);
        } else if (mutation.getType().startsWith("CG")) {
            gene.setCGNum(gene.getCGNum() + 1);
        }
    }
}
```

# 算法设计

- t 检验算法

```java
/**
 * t检验
 *
 * @param genes     基因序列数据
 * @param nonGenes 非基因序列数据
 * @param flag      需要计算的突变类型“AG”表示由 A 突变为 G
 * @return
 */
double tTest(List<Information> genes, List<Information> nonGenes, String flag) {

    List<Double> geneRate = rate(genes, flag);
    List<Double> nonGeneRate = rate(nonGenes, flag);
    double geneMutRateMean = 0.0;
    double nonGeneMutRateMean = 0.0;
    double geneSum = 0.0;
    double nonGeneSum = 0.0;
    double geneVariance = 0.0;
    double nonGeneVariance = 0.0;
    double t = 0.0;

    geneMutRateMean = mean(geneRate);
    nonGeneMutRateMean = mean(nonGeneRate);
    geneSum = (double) genes.size();
    nonGeneSum = (double) nonGenes.size();
    geneVariance = variance(geneRate, geneMutRateMean);
    nonGeneVariance = variance(nonGeneRate, nonGeneMutRateMean);

    t = (geneMutRateMean - nonGeneMutRateMean) / (sqrt((((geneSum - 1) *
geneVariance) + ((nonGeneSum - 1) * nonGeneVariance)) / (geneSum + nonGeneSum -
2)) * sqrt((1 / geneSum) + (1 / nonGeneSum)));
    System.out.println("gene mean is " + geneMutRateMean);
    System.out.println("non gene mean is " + nonGeneMutRateMean);
    return t;
}
```

# 算法设计

- u 检验算法

```java
/**
 * u检验
 *
 * @param genes     基因序列数据
 * @param nonGenes 非基因序列数据
 * @param flag      需要计算的突变类型 "AG" 表示由 A 突变为 G
 * @return
 */
double uTest(List<Information> genes, List<Information> nonGenes, String flag) {
    List<Double> geneRate = rate(genes, flag);
    List<Double> nonGeneRate = rate(nonGenes, flag);
    double geneMutRateMean = 0.0;
    double nonGeneMutRateMean = 0.0;
    double geneSum = 0.0;
    double nonGeneSum = 0.0;
    double geneVariance = 0.0;
    double nonGeneVariance = 0.0;
    double u = 0.0;

    geneMutRateMean = mean(geneRate);
    nonGeneMutRateMean = mean(nonGeneRate);
    geneSum = (double) genes.size();
    nonGeneSum = (double) nonGenes.size();
    geneVariance = variance(geneRate, geneMutRateMean);
    nonGeneVariance = variance(nonGeneRate, nonGeneMutRateMean);

    u = (geneMutRateMean - nonGeneMutRateMean) / (sqrt((geneVariance / geneSum) +
(nonGeneVariance / nonGeneSum)));
    return u;
}
```

# 辅助算法

- 以下方法在 t/u 检验方法中被调用

1. 计算突变率
2. 通过此方法可以获取突变率集合

```java
/**
 * 由原始数据计算突变率
 *
 * @param information 输入的序列
 * @param flag        需要计算的突变类型“AG”表示由 A 突变为 G
 * @return 返回突变率集合
 */
List<Double> rate(List<Information> information, String flag) {
    int num = 0;
    int length = 0;
    List<Double> rates = new ArrayList<>();
    for (int i = 0; i < information.size(); i++) {
        length = information.get(i).getEndPos() -
information.get(i).getStartPos() + 1;
        if ("AT".equals(flag)) {
            num = information.get(i).getATNum();
        } else if ("AG".equals(flag)) {
            num = information.get(i).getAGNum();
        } else if ("AC".equals(flag)) {
            num = information.get(i).getACNum();
        } else if ("TA".equals(flag)) {
            num = information.get(i).getTANum();
        } else if ("TG".equals(flag)) {
            num = information.get(i).getTGNum();
        } else if ("TC".equals(flag)) {
            num = information.get(i).getTCNum();
        } else if ("GA".equals(flag)) {
            num = information.get(i).getGANum();
        } else if ("GT".equals(flag)) {
            num = information.get(i).getGTNum();
        } else if ("GC".equals(flag)) {
            num = information.get(i).getGCNum();
        } else if ("CA".equals(flag)) {
            num = information.get(i).getACNum();
        } else if ("CT".equals(flag)) {
            num = information.get(i).getCTNum();
        } else if ("CG".equals(flag)) {
            num = information.get(i).getCGNum();
        }
        rates.add((double) num / (double) length);
    }
    return rates;
}
```

# 辅助算法

- 计算突变率均值

```
/**
 * 计算突变率均值
 *
 * @param rates 突变率集合
 * @return 返回均值
 */
double mean(List<Double> rates) {
    double sum = 0.0;
    for (int i = 0; i < rates.size(); i++) {
        //Double是引用数据类型 此处 if 语句用于防止自动装箱时产生误差
       if (rates.get(i) > 0.0) {
            sum += rates.get(i);
        }
    }
    return sum / rates.size();
}
```

# 辅助算法

- 计算方差

```
/**
 * 计算方差
 *
 * @param rates 突变率集合
 * @param mean  突变率均值
 * @return 返回方差
 */
double variance(List<Double> rates, double mean) {
    double sum = 0.0;
    for (int i = 0; i < rates.size(); i++) {
        //Double是引用数据类型 此处 if 语句用于防止自动装箱时产生误差
       if (rates.get(i) > 0.0) {
            sum += pow((rates.get(i) - mean), (double) 2);
        }
    }
    return sum / (rates.size() - 1);
}
```

# 假设检验

- 主程序

```
@Test
public void mutationTest() throws IOException {
    InputStream inputStream = Resources.getResourceAsStream("mybatis-config.xml");
    SqlSessionFactoryBuilder sqlSessionFactoryBuilder = new SqlSessionFactoryBuilder();
    SqlSessionFactory sqlSessionFactory = sqlSessionFactoryBuilder.build(inputStream);
    SqlSession sqlSession = sqlSessionFactory.openSession();

    InformationMapper informationMapper = sqlSession.getMapper(InformationMapper.class);
    MutationMapper mutationMapper = sqlSession.getMapper(MutationMapper.class);

    List<Information> genes = informationMapper.getGenes();
    List<Information> nonGenes = informationMapper.getNonGenes();
```

# 假设检验

- 主程序

续

```
    System.out.println("------------------------------------");
    System.out.println("AC: t is " + tTest(genes, nonGenes, "AC"));
    System.out.println("------------------------------------");
    System.out.println("AT: t is " + tTest(genes, nonGenes, "AT"));
    System.out.println("------------------------------------");
    System.out.println("GA: t is " + tTest(genes, nonGenes, "GA"));
    System.out.println("------------------------------------");
    System.out.println("GC: t is " + tTest(genes, nonGenes, "GC"));
    System.out.println("------------------------------------");
    System.out.println("GT: t is " + tTest(genes, nonGenes, "GT"));
    System.out.println("------------------------------------");
    System.out.println("CA: t is " + tTest(genes, nonGenes, "CA"));
    System.out.println("------------------------------------");
    System.out.println("CG: t is " + tTest(genes, nonGenes, "CG"));
    System.out.println("------------------------------------");
    System.out.println("CT: t is " + tTest(genes, nonGenes, "CT"));
    System.out.println("------------------------------------");
    System.out.println("TA: t is " + tTest(genes, nonGenes, "TA"));
    System.out.println();
    System.out.println();
    System.out.println("------------------------------------");
    System.out.println("AC: u is " + uTest(genes, nonGenes, "AC"));
    System.out.println("------------------------------------");
    System.out.println("AT: u is " + uTest(genes, nonGenes, "AT"));
    System.out.println("------------------------------------");
    System.out.println("GA: u is " + uTest(genes, nonGenes, "GA"));
    System.out.println("------------------------------------");
    System.out.println("GC: u is " + uTest(genes, nonGenes, "GC"));
    System.out.println("------------------------------------");
    System.out.println("GT: u is " + uTest(genes, nonGenes, "GT"));
    System.out.println("------------------------------------");
    System.out.println("CA: u is " + uTest(genes, nonGenes, "CA"));
    System.out.println("------------------------------------");
    System.out.println("CG: u is " + uTest(genes, nonGenes, "CG"));
    System.out.println("------------------------------------");
    System.out.println("CT: u is " + uTest(genes, nonGenes, "CT"));
    System.out.println("------------------------------------");
    System.out.println("TA: u is " + uTest(genes, nonGenes, "TA"));
}
```

# 输出结果

t test:

```
-----------------------------------
AC: t is -12.737652036387919
-----------------------------------
AT: t is -21.419035715474898
-----------------------------------
GA: t is -2.4649230471397994
-----------------------------------
GC: t is -8.191668558555735
-----------------------------------
GT: t is -13.803584880399859
-----------------------------------
CA: t is -12.737652036387919
-----------------------------------
CG: t is -5.413376607132111
-----------------------------------
CT: t is -5.337783238152486
-----------------------------------
TA: t is -19.556575936619616
```

u test:

```
-----------------------------------
AC: u is -12.737652036387919
-----------------------------------
AT: u is -21.419035715474898
-----------------------------------
GA: u is -2.4649230471397994
-----------------------------------
GC: u is -8.191668558555735
-----------------------------------
GT: u is -13.80358488039986
-----------------------------------
CA: u is -12.737652036387919
-----------------------------------
CG: u is -5.413376607132111
-----------------------------------
CT: u is -5.337783238152486
-----------------------------------
TA: u is -19.556575936619616
```

# 生物学意义

- 根据假设检验结果:
  - 基因序列和非基因序列突变率存在显著差异
  - 不同突变方向在基因序列和非基因序列之间存在显著差异
  - 不同突变方向之间部分存在显著差异

- 源代码
  - GitHub: Bluuur/BiostaticsProject1: 生物统计学课程项目1 (github.com)
  - 同步至 Gitee: BiostaticsProject1: 生物统计学课程项目1 (gitee.com)
- 实验报告
  - GitHub: MarkdownNotes/Biostatics at main · Bluuur/MarkdownNotes (github.com)
  - 同步至 Gitee: Biostatics · blur/MarkdownNotes - 码云 - 开源中国 (gitee.com)