Kratki pregled značaja analize DNA sekvenci u biologiji, genetici i biomedicini

Analiza DNA sekvenci ima ključnu ulogu u suvremenoj biologiji, genetici i biomedicini. Od otkrića strukture DNA do razvoja naprednih tehnologija sekvenciranja, ovo područje značajno je pridonijelo našem razumijevanju života na molekularnoj razini. DNA sekvence sadrže genetske informacije koje određuju biološke procese, nasljeđivanje i evoluciju.

U biologiji, analiza DNA omogućila je otkrivanje gena odgovornih za različite fenotipske karakteristike, mapiranje genoma i istraživanje evolucijskih odnosa među vrstama. Genetika koristi ove metode za proučavanje nasljednih bolesti, identifikaciju genetskih mutacija i razumijevanje kompleksnih nasljednih obrazaca. U biomedicini, analiza DNA sekvenci revolucionirala je dijagnostiku, omogućila personaliziranu medicinu te razvoj ciljanih terapija za mnoge bolesti.

Razvojem tehnologija poput masivnog paralelnog sekvenciranja (NGS) i računalnih metoda analize, postalo je moguće analizirati velike količine podataka u kratkom vremenu. Ova interdisciplinarna sinergija matematike, statistike i bioinformatike omogućila je nove uvide u biološke procese i ubrzala znanstvena otkrića.

Kroz ove primjene, analiza DNA sekvenci ne samo da produbljuje naše temeljno razumijevanje biologije, već ima i izravan utjecaj na poboljšanje kvalitete života.

Razvoj metoda analize DNA od tradicionalnih laboratorijskih tehnika do modernih računalnih pristupa

Razvoj metoda analize DNA prošao je kroz nekoliko ključnih faza, od tradicionalnih laboratorijskih tehnika do sofisticiranih računalnih pristupa koji se koriste danas. U ranim fazama molekularne biologije, istraživanje DNA oslanjalo se na klasične laboratorijske metode poput izolacije DNA i elektroforeze. Ove metode omogućile su znanstvenicima proučavanje osnovnih karakteristika genetskog materijala, ali su bile ograničene u pogledu brzine i obujma analize.

Jedan od prvih revolucionarnih koraka u analizi DNA bila je metoda Sangerovog sekvenciranja razvijena 1977. godine. Ova tehnika omogućila je sekvenciranje relativno kratkih DNA fragmenata i postala je zlatni standard u genetičkim istraživanjima tijekom nekoliko desetljeća. Iako precizna, metoda Sangerovog sekvenciranja bila je spora i skupa, što je ograničavalo njezinu primjenu na veće genomske projekte.

S razvojem tehnologije početkom 21. stoljeća, pojavile su se metode masivnog paralelnog sekvenciranja (NGS), koje su donijele revoluciju u analizu DNA. Tehnologije poput Illumina sekvenciranja omogućile su sekvenciranje cijelih genoma u razumnom vremenskom roku i po pristupačnoj cijeni. Ove metode povećale su brzinu i obujam analize, omogućujući znanstvenicima da generiraju ogromne količine podataka o genomima organizama.

Razvojem modernih računalnih pristupa, analize DNA proširile su se izvan samog sekvenciranja. Bioinformatički alati i algoritmi omogućili su obradu, poravnanje i analizu velikih sekvencijskih podataka. Matematički modeli, statističke analize i strojarno učenje sve se više koriste za identifikaciju uzoraka, predviđanje funkcija gena i razumijevanje genetskih mreža.

Ova evolucija od tradicionalnih laboratorijskih tehnika do računalnih pristupa ne samo da je ubrzala analize DNA, već je omogućila rješavanje kompleksnih bioloških problema na način koji prije nije bio moguć. Očekuje se da će daljnji napredak u tehnologiji i računalnim znanostima dodatno unaprijediti mogućnosti analize DNA u budućnosti.

Uloga matematičkih metoda u obradi i analizi velikih količina podataka generiranih sekvenciranjem

Razvoj tehnologija masivnog sekvenciranja (NGS) rezultirao je eksponencijalnim rastom podataka vezanih uz DNA sekvence. Obrada i analiza ovako velikih količina podataka postala je nemoguća bez primjene naprednih matematičkih metoda koje omogućuju efikasnu obradu, interpretaciju i vizualizaciju.

Jedna od ključnih primjena matematičkih metoda u analizi DNA je teorija informacija. Shannonova entropija koristi se za mjerenje raznolikosti unutar sekvenci, omogućujući identifikaciju regija visoke konzervacije ili varijabilnosti. Kompresijski algoritmi dodatno olakšavaju analizu složenosti genoma, omogućujući poredbu različitih organizama na temelju njihove genetske kompleksnosti.

Statistički modeli, kao što su Markovljevi lanci i skriveni Markovljevi modeli (HMM), koriste se za predikciju funkcionalnih regija DNA, poput promotorskih regija, introna i egzona. Ovi modeli omogućuju identifikaciju uzoraka u sekvencama i olakšavaju proces anotacije genoma.

Matematički pristupi također su temelj algoritama za poravnanje sekvenci, poput Needleman-Wunsch i Smith-Waterman algoritama, koji koriste dinamičko programiranje za optimalno poravnanje nukleotidnih nizova. Ovi algoritmi čine osnovu za mnoge bioinformatičke alate poput BLAST-a i ClustalW, koji su ključni za analizu homologije između sekvenci.

Primjena metoda poput klaster analize i glavnih komponenata (PCA) omogućuje smanjenje dimenzionalnosti podataka i identifikaciju skrivenih obrazaca unutar velikih datasetova. Ove metode često se koriste za identifikaciju genetskih markera i klasifikaciju uzoraka na temelju genetskih profila.

Dodatno, strojarno učenje i umjetna inteligencija sve više dobivaju na značaju u analizi DNA podataka. Algoritmi poput neuronskih mreža, podržanih vektorskih strojeva (SVM) i random forest metoda omogućuju predikciju funkcija gena, identifikaciju mutacija povezanih s bolestima i otkrivanje kompleksnih genetskih mreža.

Matematičke metode ne samo da ubrzavaju analizu DNA podataka, već omogućuju detaljniju interpretaciju bioloških procesa. Njihova interdisciplinarna primjena spaja biologiju, statistiku, računalne znanosti i matematiku, stvarajući temelje za nove znanstvene spoznaje i praktične primjene u biomedicini i genetskim istraživanjima.

Reprezentacija DNA sekvenci

Analiza DNA sekvenci počinje prikladnom reprezentacijom sekvenci kako bi se omogućila njihova matematička i računalna obrada. DNA sekvence, koje se sastoje od četiri osnovne baze (adenin - A, timin - T, gvanin - G i citozin - C), mogu se kodirati na različite načine, uključujući numeričke ili simboličke nizove, matrice i vektore.

Kodiranje DNA sekvenci (A, T, G, C) u numeričke ili simboličke nizove

Jednostavan pristup kodiranju DNA sekvenci uključuje pretvorbu baza u numeričke ili simboličke nizove. Na primjer, DNA sekvenca `ATCGGTA` može se kodirati koristeći fiksnu numeričku vrijednost za svaku bazu:

$$\text{A} = 1, \quad \text{T} = 2, \quad \text{C} = 3, \quad \text{G} = 4$$

Time se sekvenca `ATCGGTA` pretvara u niz:

$$[1, 2, 3, 4, 4, 2, 1]$$

U R-u, ovo se može implementirati na sljedeći način:

```
# Definicija sekvence
dna_sequence <- "ATCGGTA"

# Kodiranje baza u numeričke vrijednosti
dna_to_numeric <- function(sequence) {
bases <- unlist(strsplit(sequence, ""))
numeric_values <- match(bases, c("A", "T", "C", "G"))
return(numeric_values)
}

# Primjena funkcije
numeric_sequence <- dna_to_numeric(dna_sequence)
print(numeric_sequence)
```

Drugačiji pristup uključuje binarno kodiranje, gdje svaka baza ima vlastiti binarni niz:

$$\text{A} = [1, 0, 0, 0], \quad \text{T} = [0, 1, 0, 0], \quad \text{C} = [0, 0, 1, 0], \quad \text{G} = [0, 0, 0, 1]$$

Sekvenca `ATCG` tada postaje:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Primjena binarnog kodiranja u R-u:

```
# Binarno kodiranje DNA baza
binary_encoding <- function(sequence) {
bases <- unlist(strsplit(sequence, ""))
encoding <- sapply(bases, function(base) {
switch(base,
"A" = c(1, 0, 0, 0),
"T" = c(0, 1, 0, 0),
"C" = c(0, 0, 1, 0),
"G" = c(0, 0, 0, 1))
})
return(t(encoding))
}

# Primjena funkcije
binary_sequence <- binary_encoding(dna_sequence)
print(binary_sequence)
```

Korištenje matrica i vektora za predstavljanje sekvenci

Kodiranje DNA sekvenci u matricama omogućuje lakšu primjenu matematičkih operacija poput poravnanja sekvenci, analize frekvencija baza ili vizualizacije obrazaca.

Jedan primjer je reprezentacija sekvence koristeći učestalost baza u kliznim prozorima određene duljine (k-mer analiza). Za sekvencu `ATCGATCG` i klizni prozor duljine 3 ($k = 3$), dobivamo:

$$\text{Prozori} = \{\text{ATC}, \text{TCG}, \text{CGA}, \text{GAT}, \text{ATC}\}$$

Matrica učestalosti za svaki prozor može se zapisati kao:

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Primjer u R-u:

```
# Funkcija za generiranje k-mer prozora
k_mer_windows <- function(sequence, k) {
n <- nchar(sequence)
windows <- sapply(1:(n - k + 1), function(i) substr(sequence, i, i + k - 1))
return(windows)
}

# Funkcija za matricu učestalosti baza
frequency_matrix <- function(windows) {
k <- nchar(windows[1])
frequency <- matrix(0, nrow = k, ncol = 4)
colnames(frequency) <- c("A", "T", "C", "G")

for (i in 1:k) {
bases <- substr(windows, i, i)
for (base in c("A", "T", "C", "G")) {
frequency[i, base] <- sum(bases == base)
}
}
return(frequency)
}

# Generiranje prozora i matrice
windows <- k_mer_windows(dna_sequence, 3)
freq_matrix <- frequency_matrix(windows)
print(freq_matrix)
```

Zaključak

Reprezentacija DNA sekvenci u numeričkim i simboličkim nizovima, kao i u matricama, čini temelj za većinu bioinformatičkih analiza. Matematičke metode i implementacije u alatima poput R-a omogućuju efikasnu manipulaciju i analizu ovih podataka, pružajući uvid u strukturu i funkciju DNA.

Teorija informacija u analizi DNA

Teorija informacija pruža moćne alate za analizu DNA sekvenci. Kroz matematičke mjere poput entropije i kompresije, moguće je kvantificirati raznolikost sekvenci, detektirati uzorke i procijeniti složenost genetskog materijala. Ove metode omogućuju dublje razumijevanje genetske strukture i funkcije.

Shannonova entropija za analizu raznolikosti sekvenci

Shannonova entropija, definirana unutar teorije informacija, mjeri nesigurnost ili nepredvidivost skupa podataka. U kontekstu DNA sekvenci, entropija kvantificira raznolikost baza (A, T, C, G) u određenom segmentu.

Matematička formula za Shannonovu entropiju je:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

gdje je: - H: Shannonova entropija, - p_i: vjerojatnost pojavljivanja baze i (A, T, C ili G), - n: broj različitih baza.

Primjer: Razmotrimo DNA sekvencu `ATCGATCG`. Frekvencije baza su:

$$p_A = 2/8, \quad p_T = 2/8, \quad p_C = 2/8, \quad p_G = 2/8$$

Entropija se izračunava kao:

$$H = -(4 \cdot (2/8) \cdot \log_2(2/8)) = 2$$

Ova vrijednost označava maksimalnu raznolikost, jer su sve baze jednako zastupljene.

Primjena Shannonove entropije u R-u:

```
# Funkcija za Shannonovu entropiju
shannon_entropy <- function(sequence) {
bases <- unlist(strsplit(sequence, ""))
base_freq <- table(bases) / length(bases)
entropy <- -sum(base_freq * log2(base_freq))
return(entropy)
}

# Primjer
dna_sequence <- "ATCGATCG"
entropy <- shannon_entropy(dna_sequence)
print(entropy)
```

Kompresijski algoritmi za procjenu složenosti sekvenci

Kompresijski algoritmi koriste se za procjenu složenosti DNA sekvenci. Princip je jednostavan: što se više sekvenca može komprimirati, to je manje složena. Nasuprot tome, sekvence s visokom složenošću, poput nasumičnih nizova, teško se komprimiraju.

Kompresijska omjer (C) definiran je kao:

$$C = \frac{L_{compressed}}{L_{original}}$$

gdje je: - $L_{compressed}$: duljina sekvence nakon kompresije, - $L_{original}$: početna duljina sekvence.

Sekvence s visokim stupnjem ponavljanja (npr. `AAAAAA`) imaju $C \ll 1$, dok nasumične sekvence imaju $C \approx 1$.

Primjena u R-u koristi ugrađeni algoritam za kompresiju:

```
# Funkcija za kompresijski omjer
compression_ratio <- function(sequence) {
original_length <- nchar(sequence)
compressed_length <- nchar(memCompress(charToRaw(sequence), type = "gzip"))
ratio <- compressed_length / original_length
return(ratio)
}

# Primjer
dna_sequence <- "ATCGATCGATCG"
ratio <- compression_ratio(dna_sequence)
print(ratio)
```

Primjer: Za sekvencu `ATCGATCGATCG`, koja ima ponavljanje, C će biti manji od 1, dok će za `AGCTTACG` omjer biti bliži 1 zbog manje ponavljanja.

Zaključak

Shannonova entropija i kompresijski algoritmi ključni su alati za analizu raznolikosti i složenosti DNA sekvenci. Entropija pruža kvantitativnu mjeru nesigurnosti u sekvenci, dok kompresijski omjeri otkrivaju obrasce i ponavljanja. Njihova primjena doprinosi boljem razumijevanju genetske informacije i strukture.

Statistički modeli

Statistički modeli pružaju snažne alate za analizu DNA sekvenci, omogućujući modeliranje složenih obrazaca poput prijelaza između nukleotida i identifikacije funkcionalnih regija. Ova sekcija pokriva primjenu Markovljevih lanaca i skrivenih Markovljevih modela (HMM) u analizi DNA.

Markovljevi lanci za modeliranje nukleotidnih prijelaza

Markovljevi lanci su stohastički modeli koji se koriste za opisivanje prijelaza između stanja, gdje buduće stanje ovisi samo o trenutnom stanju, a ne o prethodnim stanjima. U kontekstu DNA, stanja predstavljaju nukleotide (`A`, `T`, `C`, `G`), a prijelazne vjerojatnosti modeliraju prijelaze između nukleotida.

Prijelazna matrica P za Markovljev lanac definirana je kao:

$$P = \begin{bmatrix} P_{AA} & P_{AT} & P_{AC} & P_{AG} \\ P_{TA} & P_{TT} & P_{TC} & P_{TG} \\ P_{CA} & P_{CT} & P_{CC} & P_{CG} \\ P_{GA} & P_{GT} & P_{GC} & P_{GG} \end{bmatrix}$$

gdje P_{ij} označava vjerojatnost prijelaza iz stanja i u stanje j.

Primjer: Pretpostavimo DNA sekvencu `ATCGAT`. Izračun prijelaznih vjerojatnosti: - $P_{AT} = \frac{\text{Broj prijelaza iz } A \text{ u } T}{\text{Ukupan broj prijelaza iz } A}$.

Primjena Markovljevih lanaca u R-u:

```
# Funkcija za generiranje prijelazne matrice
generate_transition_matrix <- function(sequence) {
bases <- c("A", "T", "C", "G")
transition_matrix <- matrix(0, nrow = 4, ncol = 4, dimnames = list(bases, bases))

seq_split <- unlist(strsplit(sequence, ""))
for (i in 1:(length(seq_split) - 1)) {
current <- seq_split[i]
next <- seq_split[i + 1]
transition_matrix[current, next] <- transition_matrix[current, next] + 1
}

row_sums <- rowSums(transition_matrix)
transition_matrix <- sweep(transition_matrix, 1, row_sums, FUN = "/")
return(transition_matrix)
```

```
}

# Primjer
dna_sequence <- "ATCGATCG"
transition_matrix <- generate_transition_matrix(dna_sequence)
print(transition_matrix)
```

Markovljevi lanci omogućuju analizu obrazaca u prijelazima između nukleotida, što može otkriti važne biološke informacije o DNA sekvenci.

Skupni modeli (Hidden Markov Models) za identifikaciju gena i regija regulatorne DNA

Skriveni Markovljevi modeli (HMM) proširuju Markovljeve lance dodavanjem skrivenih stanja koja nisu izravno promatrana. HMM se često koristi za identificiranje gena, introna, egzona i regulatornih regija u DNA sekvencama.

HMM je definiran s tri glavna skupa parametara: 1. Skup skrivenih stanja (S): npr. introni, egzoni. 2. Emisijske vjerojatnosti (B): vjerojatnosti generiranja određene baze (`A`, `T`, `C`, `G`) iz svakog stanja. 3. Prijelazne vjerojatnosti (A): prijelazi između skrivenih stanja.

Viterbijev algoritam često se koristi za pronalaženje najvjerojatnijeg niza skrivenih stanja za zadanu DNA sekvencu.

Primjena HMM-a u R-u koristeći paket `HMM`:

```
# Instalacija paketa HMM
# install.packages("HMM")

library(HMM)

# Definicija modela
states <- c("Exon", "Intron")
symbols <- c("A", "T", "C", "G")

# Prijelazne i emisijske vjerojatnosti
trans_probs <- matrix(c(0.9, 0.1, 0.1, 0.9), nrow = 2, byrow = TRUE)
emission_probs <- matrix(c(0.3, 0.3, 0.2, 0.2, 0.2, 0.2, 0.3, 0.3), nrow = 2, byrow = TRUE)

# Kreiranje HMM-a
hmm <- initHMM(states, symbols, transProbs = trans_probs, emissionProbs = emission_probs)

# Primjer DNA sekvence
dna_sequence <- "ATCGATCG"
seq_split <- unlist(strsplit(dna_sequence, ""))

# Viterbijev algoritam
viterbi_path <- viterbi(hmm, seq_split)
print(viterbi_path)
```

U ovom primjeru, HMM omogućuje klasifikaciju baza kao dijelova egzona ili introna na temelju prijelaznih i emisijskih vjerojatnosti.

Zaključak

Statistički modeli poput Markovljevih lanaca i skrivenih Markovljevih modela pružaju moćne alate za analizu DNA sekvenci. Dok Markovljevi lanci modeliraju prijelaze između nukleotida, HMM omogućuju identifikaciju skrivenih stanja poput gena ili regulatornih regija, čime značajno doprinose razumijevanju strukture i funkcije genoma.

Lokalno i globalno poravnanje sekvenci

Poravnanje DNA sekvenci ključni je korak u analizi genetskog materijala, omogućujući identificiranje sličnosti između nizova. Lokalno i globalno poravnanje koriste algoritme dinamičkog programiranja za optimizaciju poravnanja. U ovom poglavlju fokusiramo se na algoritme Needleman-Wunsch (globalno poravnanje) i Smith-Waterman (lokalno poravnanje).

Dinamičko programiranje za poravnanje sekvenci

Algoritmi dinamičkog programiranja koriste matricu rezultata kako bi izračunali optimalno poravnanje između dviju sekvenci. Osnovna formula za dinamičko programiranje je:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{podudaranje ili nepodudaranje} \\ F(i-1,j) + d & \text{umetanje} \\ F(i,j-1) + d & \text{brisanje} \end{cases}$$

gdje je: - $F(i,j)$: optimalan rezultat za i-tu bazu prve sekvence i j-tu bazu druge sekvence, - $s(x_i, y_j)$: rezultat za podudaranje $(+1)$ ili nepodudaranje (-1), - d: penal za umetanje ili brisanje.

Razlika između algoritama: - **Needleman-Wunsch**: Izračunava globalno poravnanje cijelih sekvenci. - **Smith-Waterman**: Izračunava lokalno poravnanje, optimizirajući podnizove.

Needleman-Wunsch algoritam (Globalno poravnanje)

Needleman-Wunsch algoritam izračunava optimalno poravnanje između cijelih sekvenci. Postupak uključuje: 1. Inicijalizaciju matrice rezultata. 2. Rekurzivno popunjavanje matrice koristeći gore navedenu formulu. 3. Rekonstrukciju puta za optimalno poravnanje.

Primjena u R-u:

```
# Funkcija za Needleman-Wunsch algoritam
needleman_wunsch <- function(seq1, seq2, match = 1, mismatch = -1, gap = -2) {
n <- nchar(seq1)
m <- nchar(seq2)
seq1 <- unlist(strsplit(seq1, ""))
seq2 <- unlist(strsplit(seq2, ""))

# Inicijalizacija matrice
score_matrix <- matrix(0, nrow = n + 1, ncol = m + 1)
for (i in 1:(n + 1)) score_matrix[i, 1] <- (i - 1) * gap
for (j in 1:(m + 1)) score_matrix[1, j] <- (j - 1) * gap

# Popunjavanje matrice
for (i in 2:(n + 1)) {
for (j in 2:(m + 1)) {
match_score <- ifelse(seq1[i - 1] == seq2[j - 1], match, mismatch)
score_matrix[i, j] <- max(
score_matrix[i - 1, j - 1] + match_score,
score_matrix[i - 1, j] + gap,
score_matrix[i, j - 1] + gap
)
}
}

return(score_matrix)
}

# Primjer
seq1 <- "GATTACA"
seq2 <- "GCATGCU"
nw_matrix <- needleman_wunsch(seq1, seq2)
print(nw_matrix)
```

Smith-Waterman algoritam (Lokalno poravnanje)

Smith-Waterman algoritam koristi istu osnovnu logiku kao Needleman-Wunsch, ali uvodi dodatni korak:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{podudaranje ili nepodudaranje} \\ F(i-1,j) + d & \text{umetanje} \\ F(i,j-1) + d & \text{brisanje} \\ 0 & \text{započinjanje novog poravnanja} \end{cases}$$

Dodavanje nule omogućuje algoritmu da započne lokalno poravnanje u bilo kojoj točki matrice.

Primjena u R-u:

```
# Funkcija za Smith-Waterman algoritam
smith_waterman <- function(seq1, seq2, match = 1, mismatch = -1, gap = -2) {
n <- nchar(seq1)
m <- nchar(seq2)
seq1 <- unlist(strsplit(seq1, ""))
seq2 <- unlist(strsplit(seq2, ""))

# Inicijalizacija matrice
score_matrix <- matrix(0, nrow = n + 1, ncol = m + 1)

# Popunjavanje matrice
max_score <- 0
for (i in 2:(n + 1)) {
for (j in 2:(m + 1)) {
match_score <- ifelse(seq1[i - 1] == seq2[j - 1], match, mismatch)
score_matrix[i, j] <- max(
0,
score_matrix[i - 1, j - 1] + match_score,
score_matrix[i - 1, j] + gap,
score_matrix[i, j - 1] + gap
)
max_score <- max(max_score, score_matrix[i, j])
}
}

return(list(matrix = score_matrix, max_score = max_score))
}

# Primjer
seq1 <- "GATTACA"
seq2 <- "GCATGCU"
sw_result <- smith_waterman(seq1, seq2)
print(sw_result$matrix)
print(paste("Maksimalni rezultat:", sw_result$max_score))
```

Zaključak

Needleman-Wunsch algoritam omogućuje globalno poravnanje cijelih sekvenci, dok Smith-Waterman optimizira lokalne podnizove. Oba algoritma koriste dinamičko programiranje kako bi osigurali optimalna poravnanja, pružajući ključne alate za analizu DNA i proteinskih sekvenci.

Heuristički pristupi poravnavanju sekvenci

Heuristički pristupi poravnavanju sekvenci koriste približne metode za brzo i učinkovito poravnanje velikih sekvenci. Jedan od najpoznatijih algoritama u ovom području je BLAST (Basic Local Alignment Search Tool), koji omogućuje brzo lokalno poravnanje DNA, RNA ili proteinskih sekvenci s bazama podataka.

Matematička pozadina BLAST algoritma

BLAST koristi heuristički pristup kako bi identificirao regije s visokim stupnjem sličnosti između upitne sekvence i ciljne baze podataka. Proces uključuje sljedeće korake:

1. **Razbijanje upitne sekvence na riječi**: - Upitna sekvenca razdvaja se u riječi fiksne duljine k, gdje je k unaprijed definirana duljina (obično $k = 3$ za proteine i $k = 11$ za DNA). - Riječi se generiraju pomičnim prozorom preko sekvence.

2. **Pretraživanje podudaranja u bazi podataka**: - Riječi iz upitne sekvence uspoređuju se s riječima iz baze podataka koristeći unaprijed izračunate rezultate (scoring matrices) poput PAM ili BLOSUM za proteine.

3. **Proširenje podudaranja**: - Kada se pronađe podudaranje riječi, algoritam proširuje lokalno poravnanje u oba smjera dok rezultat ostaje iznad praga (S).

4. **Evaluacija značajnosti**: - BLAST koristi E-vrijednost (expected value) za procjenu značajnosti poravnanja. Matematički, E-vrijednost se definira kao:

$$E = Kmne^{-\lambda S}$$

gdje su: - K i λ: parametri specifični za bazu podataka, - m, n: duljine upitne i ciljne sekvence, - S: rezultat poravnanja.

Primjena BLAST algoritma u R-u

U R-u se BLAST može koristiti integracijom s vanjskim alatima poput `blastn` ili `blastp`. Primjer korištenja BLAST-a:

1. Instalirajte BLAST alat iz NCBI-a. 2. Kreirajte lokalnu bazu podataka. 3. Pokrenite BLAST pretragu.

```
# Korisnička funkcija za pokretanje BLAST-a
run_blast <- function(query_file, db_file, output_file) {
blast_cmd <- sprintf("blastn -query %s -db %s -out %s -outfmt 6",
query_file, db_file, output_file)
system(blast_cmd)
}

# Primjer korištenja
query_file <- "query.fasta"   # Upitna sekvenca
db_file <- "database"        # Lokalna baza podataka
output_file <- "blast_output.txt"  # Izlazni rezultat

# Pokretanje BLAST-a
run_blast(query_file, db_file, output_file)

# Čitanje rezultata
blast_results <- read.table(output_file, header = FALSE)
print(blast_results)
```

Matematičke osnove ocjenjivanja rezultata

BLAST koristi ocjenjivanje za evaluaciju podudaranja između sekvenci. Glavni elementi uključuju:

- **Rezultat poravnanja (S)**:

$$S = \sum_{i=1}^{L} s(x_i, y_i)$$

gdje je: - L: duljina poravnanja, - $s(x_i, y_i)$: rezultat zamjene za par (x_i, y_i), preuzet iz matrice rezultata (npr. BLOSUM62).

- **E-vrijednost (E)**: Kao što je ranije navedeno, E-vrijednost ovisi o S i duljinama sekvenci.

Primjer implementacije ocjenjivanja u R-u:

```
# Funkcija za izračun rezultata poravnanja
alignment_score <- function(seq1, seq2, match = 1, mismatch = -1, gap = -2) {
```

```
n <- min(nchar(seq1), nchar(seq2))
seq1 <- unlist(strsplit(seq1, ""))
seq2 <- unlist(strsplit(seq2, ""))

score <- 0
for (i in 1:n) {
if (seq1[i] == seq2[i]) {
score <- score + match
} else {
score <- score + mismatch
}
}
return(score)
}

# Primjer
seq1 <- "GATTACA"
seq2 <- "GCATGCU"
score <- alignment_score(seq1, seq2)
print(score)
```

Prednosti i ograničenja BLAST-a

Prednosti:

- Izuzetno brzo poravnanje sekvenci, što ga čini idealnim za pretrage velikih baza podataka.
- Značajna fleksibilnost u ocjenjivanju (koristeći prilagođene matrice rezultata).

Ograničenja:

- BLAST se oslanja na heuristiku, što može rezultirati propuštanjem nekih optimalnih poravnanja.
- Pogodan je za sekvence s visokim stupnjem sličnosti, ali manje za udaljenije homologije.

Zaključak

BLAST je jedan od najvažnijih alata u bioinformatici zbog svoje brzine i fleksibilnosti. Njegova matematička osnova omogućuje učinkovito poravnanje velikih sekvenci, dok E-vrijednost pruža kvantitativnu procjenu značajnosti poravnanja. Kombinacija heurističkih metoda i matematičkih modela čini ga ključnim za analizu genetskih podataka.

Evaluacija poravnanja sekvenci

Evaluacija poravnanja sekvenci važan je korak u bioinformatici jer omogućuje procjenu značajnosti rezultata poravnanja. Statistički testovi pružaju mjeru vjerojatnosti da je poravnanje rezultat biološke sličnosti, a ne slučajnosti. U ovom poglavlju objašnjavamo ključne metrike i testove koji se koriste za evaluaciju poravnanja sekvenci.

Statistički modeli za evaluaciju poravnanja

Kada poravnamo dvije sekvence, cilj je odrediti je li rezultat statistički značajan. Dvije ključne metrike u evaluaciji poravnanja su rezultat poravnanja (S) i E-vrijednost.

1. Rezultat poravnanja (S): Rezultat poravnanja računa se na temelju bodovne matrice (npr. BLOSUM za proteine) i penala za umetanja i brisanja:

$$S = \sum_{i=1}^{L} s(x_i, y_i) + \text{penali za umetanja/brisanja}$$

gdje je: - L: duljina poravnanja, - $s(x_i, y_i)$: rezultat podudaranja ili nepodudaranja za bazu x_i i y_i.

2. E-vrijednost (Expected value): E-vrijednost procjenjuje broj očekivanih poravnanja jednake ili veće kvalitete S koja bi nastala slučajno. Definira se kao:

$$E = Kmne^{-\lambda S}$$

gdje su: - K: parametar specifičan za bazu podataka, - m, n: duljine upitne i ciljne sekvence, - λ: skalirajući faktor za bodovnu matricu, - S: rezultat poravnanja.

Niža E-vrijednost znači da je poravnanje značajnije.

Statistički testovi za procjenu značajnosti

Statistički testovi omogućuju usporedbu rezultata poravnanja s distribucijom očekivanih rezultata. Ključni pristupi uključuju:

1. Permutacijski testovi: Permutacijski testovi slučajno permutiraju jednu od sekvenci više puta kako bi generirali distribuciju rezultata slučajnih poravnanja. P-vrijednost izračunava se kao:

$$p = \frac{\text{Broj rezultata } \geq S_{\text{observed}}}{\text{Ukupan broj permutacija}}$$

2. Z-vrijednost: Z-vrijednost standardizira rezultat poravnanja S koristeći srednju vrijednost i standardnu devijaciju slučajne distribucije rezultata:

$$Z = \frac{S - \mu}{\sigma}$$

gdje su: - μ: srednja vrijednost slučajnih rezultata, - σ: standardna devijacija.

3. Monte Carlo simulacije: Monte Carlo simulacije koriste generiranje velikog broja slučajnih poravnanja kako bi se procijenila distribucija rezultata.

Primjena u R-u

1. Permutacijski test za evaluaciju poravnanja Sljedeći R kod implementira permutacijski test za poravnanje dviju sekvenci:

```
# Funkcija za izračun rezultata poravnanja
alignment_score <- function(seq1, seq2, match = 1, mismatch = -1, gap = -2) {
n <- min(nchar(seq1), nchar(seq2))
seq1 <- unlist(strsplit(seq1, ""))
seq2 <- unlist(strsplit(seq2, ""))
score <- 0
for (i in 1:n) {
if (seq1[i] == seq2[i]) {
score <- score + match
} else {
score <- score + mismatch
}
}
return(score)
}

# Permutacijski test
permutation_test <- function(seq1, seq2, n_permutations = 1000) {
observed_score <- alignment_score(seq1, seq2)
random_scores <- numeric(n_permutations)
for (i in 1:n_permutations) {
permuted_seq2 <- sample(unlist(strsplit(seq2, "")), replace = FALSE)
permuted_seq2 <- paste0(permuted_seq2, collapse = "")
random_scores[i] <- alignment_score(seq1, permuted_seq2)
}
p_value <- sum(random_scores >= observed_score) / n_permutations
return(list(observed_score = observed_score, p_value = p_value))
```

```
}

# Primjer
seq1 <- "GATTACA"
seq2 <- "GCATGCU"
result <- permutation_test(seq1, seq2)
print(result)
```

2. Procjena E-vrijednosti Sljedeći kod izračunava E-vrijednost:

```
calculate_e_value <- function(score, m, n, lambda = 0.1, K = 0.5) {
e_value <- K * m * n * exp(-lambda * score)
return(e_value)
}

# Primjer
e_value <- calculate_e_value(score = 10, m = 100, n = 200)
print(e_value)
```

Zaključak

Statistički testovi, poput permutacijskih testova, Z-vrijednosti i procjene E-vrijednosti, ključni su za razumijevanje značajnosti poravnanja sekvenci. Kombinacija ovih testova osigurava pouzdane rezultate u bioinformatičkoj analizi i omogućuje razlikovanje biološki značajnih poravnanja od slučajnih podudaranja.

Matematički modeli evolucije

Matematički modeli evolucije koriste se za analizu genetskih promjena kroz vrijeme i za rekonstrukciju filogenetskih odnosa među organizmima. U ovom poglavlju obrađuju se substitucijski modeli i Bayesovske metode koje su ključne za razumijevanje evolucijskih procesa i izgradnju filogenetskih stabala.

Substitucijski modeli (npr. Jukes-Cantor, Kimura)

Substitucijski modeli opisuju stope zamjena nukleotida u DNA sekvencama. Ovi modeli pružaju matematički okvir za procjenu vjerojatnosti promjena između različitih baza (A, T, C, G) tijekom evolucije.

1. Jukes-Cantor model (JC69) Jukes-Cantor model je najjednostavniji substitucijski model. Pretpostavlja jednaku vjerojatnost zamjene između bilo koje dvije baze i konstantnu stopu zamjene (α).

Matematički, vjerojatnost da baza i ostane nepromijenjena tijekom vremena t je:

$$P(i \to i) = \frac{1}{4} + \frac{3}{4}e^{-4\alpha t}$$

Vjerojatnost zamjene baze i s bazom j $(i \neq j)$ je:

$$P(i \to j) = \frac{1}{4} - \frac{1}{4}e^{-4\alpha t}$$

2. Kimura model (K80) Kimura model uvodi različite stope za prelaze (κ) i transverzije (λ): - **Prelazi**: Zamjene između purina $(A \leftrightarrow G)$ ili pirimidina $(C \leftrightarrow T)$. - **Transverzije**: Zamjene između purina i pirimidina.

Matematički, prijelazna matrica za Kimura model je:

$$P = \begin{bmatrix} P(A \to A) & P(A \to G) & P(A \to C) & P(A \to T) \\ P(G \to A) & P(G \to G) & P(G \to C) & P(G \to T) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Primjena substitucijskih modela u R-u Koristimo paket `ape` za primjenu ovih modela:

```
library(ape)

# Generiranje sekvence
```

```
seq1 <- "ACGTACGTACGT"
seq2 <- "ACCTACGTTCGT"

# Izračun Jukes-Cantor udaljenosti
jc_dist <- dist.dna(DNAbin = as.DNAbin(c(seq1, seq2)), model = "JC69")
print(jc_dist)

# Izračun Kimura udaljenosti
k80_dist <- dist.dna(DNAbin = as.DNAbin(c(seq1, seq2)), model = "K80")
print(k80_dist)
```

Bayesovske metode za rekonstrukciju filogenetskih stabala

Bayesovske metode koriste a posteriori vjerojatnosti za rekonstrukciju filogenetskih stabala, kombinirajući vjerojatnosti podataka (substitucijskog modela) s prior distribucijama.

1. Bayesova formula A posteriori vjerojatnost stabla T za dane podatke D je:

$$P(T|D) = \frac{P(D|T)P(T)}{P(D)}$$

gdje su: - $P(D|T)$: vjerojatnost podataka za zadano stablo, - $P(T)$: prior distribucija stabla, - $P(D)$: marginalna vjerojatnost podataka.

2. Markov Chain Monte Carlo (MCMC) Bayesovske metode često koriste MCMC algoritam za uzorkovanje filogenetskih stabala iz a posteriori distribucije. Ključni koraci uključuju: - Generiranje kandidata za novo stablo. - Izračunavanje omjera vjerojatnosti između novog i trenutnog stabla. - Prihvaćanje ili odbacivanje novog stabla na temelju omjera.

3. Primjena u R-u Koristimo paket `phangorn` za rekonstrukciju filogenetskog stabla pomoću Bayesovskog pristupa:

```
library(phangorn)

# Učitavanje primjera DNA podataka
data(woodmouse)
dna <- woodmouse

# Izgradnja stabla pomoću Jukes-Cantor modela
dist_matrix <- dist.dna(dna, model = "JC69")
tree <- nj(dist_matrix)  # Neighbor-joining metoda

# Procjena stabla Bayesovskom metodom
fit <- pml(tree, dna)
fit <- optim.pml(fit, model = "JC")
print(fit$tree)
```

Zaključak

Substitucijski modeli poput Jukes-Cantor i Kimura modela pružaju temelj za analizu evolucijskih promjena, dok Bayesovske metode omogućuju preciznu rekonstrukciju filogenetskih stabala koristeći vjerojatnosni pristup. Kombinacija ovih metoda ključna je za razumijevanje evolucijskih odnosa među vrstama.

Konstruktivni algoritmi filogenetske analize

Filogenetska analiza koristi konstruktivne algoritme za izgradnju filogenetskih stabala temeljenih na podacima sekvenci. Dvije najvažnije metode su maksimalna parsimony i maksimalna vjerojatnost. Ove metode omogućuju preciznu analizu evolucijskih odnosa među vrstama.

Maksimalna parsimony

Maksimalna parsimony metoda temelji se na principu štedljivosti i pretpostavlja da je najvjerojatnije stablo ono koje zahtijeva najmanji broj evolucijskih promjena (substitucija). Ova metoda često se koristi za analizu DNA, RNA ili proteinskih sekvenci.

1. Matematički model Za zadani skup sekvenci $S = \{S_1, S_2, \ldots, S_n\}$, cilj je pronaći filogenetsko stablo T koje minimizira ukupan broj promjena:

$$L(T) = \sum_{i=1}^{m} C(T_i)$$

gdje je: - m: broj pozicija u sekvencama, - $C(T_i)$: broj promjena na i-toj poziciji u stablu T.

Stablo se evaluira izračunavanjem troška za svaku poziciju, a ukupan trošak $L(T)$ minimizira se pretraživanjem kroz moguća stabla.

2. Heuristički algoritmi Zbog velikog broja mogućih stabala ($(2n-5)!!$ za n sekvenci), koriste se heuristički algoritmi kao što su: - Wagnerova metoda za inicijalno stablo. - Pretraživanje stabala razmjenom grana (branch swapping).

3. Primjena maksimalne parsimony metode u R-u Paket `phangorn` omogućuje primjenu ove metode:

```
library(phangorn)

# Učitavanje primjera DNA podataka
data(woodmouse)
dna <- woodmouse

# Izgradnja početnog stabla koristeći metodu neighbor-joining
dist_matrix <- dist.dna(dna, model = "JC69")
tree <- nj(dist_matrix)

# Parsimony analiza
parsimony_score <- parsimony(tree, dna)
optimized_tree <- optim.parsimony(tree, dna)

# Rezultati
cat("Početni parsimony score:", parsimony_score, "\n")
cat("Optimizirano stablo:\n")
print(optimized_tree)
```

Maksimalna vjerojatnost

Maksimalna vjerojatnost metoda temelji se na vjerojatnosnim modelima evolucije, kao što su substitucijski modeli (npr. Jukes-Cantor). Cilj je pronaći stablo koje maksimizira vjerojatnost opaženih podataka.

1. Matematički model Za stablo T s parametrima Θ, vjerojatnost podataka D (sekvence) je:

$$P(D|T, \Theta) = \prod_{i=1}^{m} P(D_i|T, \Theta)$$

gdje je: - m: broj pozicija u sekvencama, - D_i: opaženi podatak na i-toj poziciji.

Cilj je maksimizirati log-vjerojatnost:

$$\mathcal{L}(T, \Theta) = \log P(D|T, \Theta)$$

2. Algoritmi za optimizaciju Maksimizacija vjerojatnosti postiže se iterativnim algoritmima kao što su: - Expectation-Maximization (EM) algoritam. - Metode numeričke optimizacije (npr. Newton-Raphson).

3. Primjena maksimalne vjerojatnosti u R-u Paket `phangorn` omogućuje procjenu stabala koristeći ovu metodu:

```
library(phangorn)

# Učitavanje DNA podataka
data(woodmouse)
dna <- woodmouse

# Izgradnja početnog stabla koristeći metodu neighbor-joining
dist_matrix <- dist.dna(dna, model = "JC69")
tree <- nj(dist_matrix)

# Maksimalna vjerojatnost
fit <- pml(tree, dna, model = "JC")
optimized_fit <- optim.pml(fit, model = "JC")

# Rezultati
cat("Log-vjerojatnost početnog stabla:", fit$logLik, "\n")
cat("Optimizirano stablo s maksimalnom vjerojatnošću:\n")
print(optimized_fit$tree)
```

Usporedba metoda

Maksimalna parsimony je prikladna za male skupove sekvenci i jednostavne analize. Pretpostavlja minimalni broj promjena, što može biti neprecizno za složene evolucijske procese.

Maksimalna vjerojatnost koristi sofisticirane modele i pruža preciznije rezultate, ali je računalno intenzivna i zahtijeva više podataka.

Zaključak

Maksimalna parsimony i maksimalna vjerojatnost ključne su metode za konstrukciju filogenetskih stabala. Dok prva nudi jednostavniji pristup, druga pruža veći stupanj preciznosti koristeći vjerojatnosne modele. Kombinacija ovih metoda često se koristi za provjeru robusnosti filogenetskih analiza.

Identifikacija ponavljajućih motiva

Identifikacija ponavljajućih motiva u DNA i proteinskim sekvencama ključna je za razumijevanje regulatornih elemenata, strukturalnih motiva i funkcionalnih regija. Motivi su kratke sekvence koje se često pojavljuju u genomima i često imaju biološku ulogu, kao što su vezna mjesta za proteine ili enzime. Ovo poglavlje obuhvaća algoritme i statističke modele za otkrivanje i analizu ovih uzoraka.

Algoritmi za otkrivanje uzoraka (npr. MEME)

Algoritmi za otkrivanje ponavljajućih motiva koriste sekvencijalne podatke kako bi identificirali uzorke ili motive koji se pojavljuju s visokim stupnjem konzervacije. Jedan od najpoznatijih alata za ovu svrhu je **MEME (Multiple Expectation Maximization for Motif Elicitation)**.

1. Matematička osnova MEME algoritma

MEME koristi pristup temeljen na maksimalizaciji očekivanja (EM) kako bi identificirao najvjerojatnije motive. Osnovni model je pozicijski specifična matrica (PSWM - Position Specific Weight Matrix), koja kvantificira vjerojatnosti baza A, T, C, G na svakoj poziciji u motivu. Svaka pozicija ima distribuciju:

$$P(X_i = b|\Theta) = \theta_{i,b}, \quad b \in \{A, T, C, G\}$$

gdje je $\Theta = \{\theta_{i,b}\}$ skup parametara.

MEME iterativno maksimizira log-vjerojatnost podataka:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} \log P(X_i|\Theta)$$

Proces uključuje: - **E-korak**: Izračunavanje očekivanih vjerojatnosti uzorka. - **M-korak**: Maksimizaciju parametara Θ.

15

2. Primjena MEME-a u R-u

Za korištenje MEME-a u R-u, potrebno je instalirati vanjski alat MEME Suite. Nakon instalacije, MEME se može pokrenuti direktno iz R-a koristeći sistemske naredbe:

```
# Sistemskom naredbom pokrećemo MEME za otkrivanje motiva
system("meme input.fasta -oc output_dir -nmotifs 3 -minw 6 -maxw 15")
```

MEME generira izlazne datoteke koje uključuju identificirane motive i njihovu statističku značajnost.

Statistički modeli za identifikaciju regulatornih elemenata

Statistički modeli omogućuju identificiranje regulatornih elemenata, poput promotora, enhancera i veznih mjesta za transkripcijske faktore. Jedan od ključnih pristupa temelji se na Bayesovim modelima i pozicijski specifičnim matricama.

1. Pozicijski specifične matrice težine (PSWM)

PSWM je matrica veličine $L \times 4$, gdje L označava duljinu motiva, a 4 stupca odgovaraju bazama A, T, C, G. Elementi matrice predstavljaju vjerojatnosti baza na svakoj poziciji:

$$PSWM[i, b] = \frac{\text{broj pojavljivanja baze } b \text{ na poziciji } i}{\text{ukupan broj sekvenci}}$$

PSWM se može koristiti za izračunavanje vjerojatnosti sekvence S koja odgovara motivu:

$$P(S|PSWM) = \prod_{i=1}^{L} PSWM[i, S_i]$$

2. Hidden Markov Models (HMM) za regulatorne elemente

HMM se često koristi za identificiranje regulatornih elemenata jer omogućuje modeliranje sekvenci s heterogenim regijama (npr. promotora i kodirajućih regija). HMM je definiran s: - Skupom stanja S (npr. "promotor", "kodirajuća regija"), - Prijelaznim vjerojatnostima $A_{ij} = P(S_t = j|S_{t-1} = i)$, - Emisijskim vjerojatnostima $B = P(X_t|S_t)$.

Viterbijev algoritam koristi se za identificiranje najvjerojatnijeg niza stanja za zadanu sekvencu.

Primjer izračuna vjerojatnosti pomoću PSWM-a u R-u

```
# Funkcija za izračunavanje vjerojatnosti sekvence na temelju PSWM-a
calculate_pswm_probability <- function(sequence, pswm) {
seq_split <- unlist(strsplit(sequence, ""))
probability <- 1
for (i in seq_along(seq_split)) {
base <- seq_split[i]
probability <- probability * pswm[i, base]
}
return(probability)
}

# Primjer PSWM-a
pswm <- matrix(c(0.8, 0.1, 0.05, 0.05,
0.1, 0.7, 0.1, 0.1,
0.2, 0.2, 0.5, 0.1,
0.05, 0.1, 0.1, 0.75),
nrow = 4, byrow = TRUE,
dimnames = list(NULL, c("A", "T", "C", "G")))

# Sekvenca za evaluaciju
sequence <- "ATCG"
prob <- calculate_pswm_probability(sequence, pswm)
print(prob)
```

3. Identifikacija regulatornih elemenata pomoću HMM-a u R-u

Za korištenje HMM-a u R-u, može se koristiti paket `HMM`:

```
library(HMM)

# Definicija HMM-a
states <- c("Promotor", "Kodirajuća")
symbols <- c("A", "T", "C", "G")
trans_probs <- matrix(c(0.9, 0.1, 0.2, 0.8), nrow = 2, byrow = TRUE)
emission_probs <- matrix(c(0.3, 0.3, 0.2, 0.2, 0.2, 0.2, 0.3, 0.3),
nrow = 2, byrow = TRUE)

hmm <- initHMM(states, symbols, transProbs = trans_probs, emissionProbs = emission_probs)

# Viterbijev algoritam za određivanje stanja
sequence <- c("A", "T", "C", "G", "A", "C", "T")
viterbi_path <- viterbi(hmm, sequence)
print(viterbi_path)
```

Zaključak

Algoritmi poput MEME-a i statistički modeli poput PSWM-a i HMM-a ključni su alati za identificiranje ponavljajućih motiva i regulatornih elemenata u genomima. Ovi alati pružaju detaljan uvid u biološke funkcije i strukturu genetskog materijala.

Pretraživanje motiva i sekvencijalnih uzoraka

Pretraživanje motiva i sekvencijalnih uzoraka ključno je za analizu genetskog materijala jer omogućuje identificiranje značajnih uzoraka koji se ponavljaju unutar DNA, RNA ili proteinskih sekvenci. Primjena teorije grafova i algoritama za pretraživanje nizova osigurava učinkovitost i točnost pri analizi velikih količina podataka.

Primjena teorije grafova u pretraživanju motiva

Teorija grafova pruža snažan matematički okvir za modeliranje i analizu sekvenci. Sekvence se mogu predstaviti kao grafovi, gdje čvorovi predstavljaju nukleotide ili aminokiseline, a bridovi odnose između njih.

1. Prikaz sekvence kao grafa Sekvenca $S = \{s_1, s_2, \ldots, s_n\}$ može se prikazati kao usmjereni graf $G = (V, E)$, gdje: - V je skup čvorova ($V = \{s_1, s_2, \ldots, s_n\}$), - E je skup bridova koji povezuju susjedne nukleotide ($E = \{(s_i, s_{i+1})\}$).

Za identifikaciju motiva koriste se algoritmi za pretraživanje putova unutar grafa, poput algoritma pretraživanja u dubinu (DFS) ili širinu (BFS).

2. Algoritmi za traženje putova Jedan od pristupa identifikaciji motiva je traženje Eulerovog puta ili Hamiltonovog puta: - **Eulerov put**: Put koji prolazi svakim bridom grafa točno jednom. - **Hamiltonov put**: Put koji prolazi svakim čvorom grafa točno jednom.

Primjer primjene teorije grafova u R-u Koristimo paket `igraph` za modeliranje i analizu grafa:

```
library(igraph)

# Kreiranje grafa za sekvencu
sequence <- c("A", "T", "C", "G", "A", "T")
edges <- c()
for (i in 1:(length(sequence) - 1)) {
edges <- c(edges, sequence[i], sequence[i + 1])
}

graph <- graph(edges = edges, directed = TRUE)

# Vizualizacija grafa
plot(graph)
```

```
# Pretraživanje motiva kao Eulerov put
eulerian <- eulerian_path(graph)
print(eulerian)
```

Algoritmi za pretraživanje nizova

Algoritmi za pretraživanje nizova koriste se za pronalazak uzoraka unutar sekvence. Najčešće korišteni algoritmi uključuju: 1. **Naivni algoritam za pretraživanje**: Provjera uzorka na svakoj poziciji sekvence. 2. **Knuth-Morris-Pratt (KMP) algoritam**: Koristi predprocesiranje uzorka za brže pretraživanje. 3. **Boyer-Moore algoritam**: Optimizira pretraživanje koristeći pravila pomaka.

1. Knuth-Morris-Pratt algoritam KMP algoritam koristi funkciju preklapanja π kako bi odredio koliko se uzorak može pomicati prilikom nepodudaranja. Funkcija π definira duljinu najvećeg prefiksa koji je istovremeno i sufiks uzorka.

Matematička definicija funkcije preklapanja π:

$$\pi[i] = \max\{k : 1 \leq k < i \text{ i } P[1 \dots k] = P[(i-k+1) \dots i]\}$$

Primjena KMP algoritma u R-u Sljedeći kod implementira KMP algoritam za pretraživanje uzorka:

```
# Funkcija za izračunavanje funkcije pi
compute_pi <- function(pattern) {
m <- nchar(pattern)
pi <- numeric(m)
k <- 0
for (q in 2:m) {
while (k > 0 && substr(pattern, k + 1, k + 1) != substr(pattern, q, q)) {
k <- pi[k]
}
if (substr(pattern, k + 1, k + 1) == substr(pattern, q, q)) {
k <- k + 1
}
pi[q] <- k
}
return(pi)
}

# KMP algoritam
kmp_search <- function(text, pattern) {
n <- nchar(text)
m <- nchar(pattern)
pi <- compute_pi(pattern)
q <- 0
matches <- c()

for (i in 1:n) {
while (q > 0 && substr(pattern, q + 1, q + 1) != substr(text, i, i)) {
q <- pi[q]
}
if (substr(pattern, q + 1, q + 1) == substr(text, i, i)) {
q <- q + 1
}
if (q == m) {
matches <- c(matches, i - m + 1)
q <- pi[q]
}
}
return(matches)
}
```

```
# Primjer
text <- "ATCGATCGATCG"
pattern <- "ATCG"
matches <- kmp_search(text, pattern)
print(matches)
```

Zaključak

Pretraživanje motiva i uzoraka u sekvencama koristi kombinaciju teorije grafova i algoritama za pretraživanje nizova. Teorija grafova omogućuje modeliranje sekvenci kao grafova i identificiranje važnih putova, dok algoritmi poput KMP-a osiguravaju učinkovito pretraživanje velikih sekvenci za specifične uzorke.

19