Zadanie 1: Dopasowanie par sekwencji - algorytm kropkowy
Anna Lasota, 236727

1.1 Macierz kropkowa, analiza złożoności czasowej:

```
1. def make_dotplot(self, fasta1, fasta2):
2.     seq1 = fasta1.get_sequence()
3.     seq2 = fasta2.get_sequence()
4.     n = len(seq1)
5.     m = len(seq2)
6.     matrix = [[0 for x in range(m)] for y in range(n)]
7.     for i in range(n):
8.         for j in range(m):
9.             if seq1[i] == seq2[j]:
10.                    matrix[i][j] = 1
11.                else:
12.                    matrix[i][j] = 0
13.         return matrix
```

Operacje podstawowe:

- 4 podstawienia (linie 2-5)
- n*m podstawienia (linia 6)
- n inkrementacji licznika pętli (linia 7)
- n+1 sprawdzeń warunku pętli (linia 7)
- 1 podstawienie licznika pętli (linia 7)
- n powtórzeń pętli wewnętrznej (linie 8-12)
    - 1 podstawienie licznika pętli (linia 8)
    - m+1 sprawdzeń warunku pętli (linia 8)
    - m inkrementacji licznika pętli (linia 8)
    - m powtórzeń ciała pętli (10-12)
        -1 sprawdzenia zgodności liter (linia 9)
        -1 podstawienia wartości macierzy kropkowej (linia 10 lub 12)

Łączna liczba operacji wynosi:
$T(n,m) = 4+nm+n+n+1+1+n(1+m+1+m+m+m(1+1)) = 9nm+4n+2 =< c*nm$

Złożoność czasowa algorytmu tworzenia macierzy kropkowej jest więc rzędu co najwyżej nm, co można zapisać jako O(nm)

1.2 Macierz kropkowa, analiza złożoności obliczeniowej przestrzennej:

- n+m – rozmiar sekwencji wejściowych (linia 2-3)
- 2 zmienne przechowujące rozmiary sekwencji wejściowych (linie 4-5)
- nm – rozmiar macierzy kropkowej (linia 6)
- 2 zmienne przechowujące liczniki pętli (linie 7 i 8)

$S(n,m) = n+m+4+nm =< c*nm$

Zatem złożoność obliczeniowa przestrzenna może być zapisana jako: O(nm)

1.3 Filtracja macierzy kropkowej, analiza złożoności czasowej:

```
1. def filter_dotplot(self, matrix, treshold, window):
2.
3.     arr = numpy.array(matrix)
4.     diags = [arr.diagonal(i) for i in range(-arr.shape[0] + 1,
   arr.shape[1])]
5.     filteredArray = numpy.zeros((arr.shape[0], arr.shape[1]), dtype=int)
6.     diagCounter = 1
7.     for n in diags:
8.         if len(n) >= window:
9.             for i in range((len(n) + 1) - window):
10.                    counter = 0
11.                    for j in range(window):
12.                        if n[j + i] == 1:
13.                            counter = counter + 1
14.                    if (counter + treshold) >= window:
15.                        for k in range(window):
16.                            if (arr.shape[0] >= diagCounter):
17.                                filteredArray[arr.shape[0] - diagCounter
   + k + i][k + i] = n[k + i]
18.                            else:
19.                                filteredArray[k + i][(diagCounter -
   arr.shape[0] + k + i)] = n[k + i]
20.         diagCounter = diagCounter + 1
21.         return filteredArray
```

Operacje podstawowe:

- n+m-1 podstawień (linia 4)
- nm podstawień (linia 5)
- 1 podstawienie (linia 6)
- n+m-1 inkrementacji licznika pętli (7)
- 1 podstawienie licznika pętli (7)
- n+m sprawdzenia warunku pętli (7)
- n+m-1 powtórzeń pętli wewnętrznej(linia 8-20)
  - 1 porównanie (linia 8)
  - 1 podstawienia licznika pętli (linia 9)
  - (mn/(m+n-1))-w+1 sprawdzenia warunku pętli (linia 9)
  - (mn/(m+n-1))-w inkrementacji licznika pętli (linia 9)
  - (mn/(m+n-1))-w powtórzeń pętli wewnętrznej(linia 10-19)
    - 1 podstawienie (linia 10)
    - w inkrementacji licznika pętli wewnętrznej(linia 11)
    - 1 podstawienie licznika pętli (linia 11)
    - w+1 sprawdzeń warunku pętli (linia 11)
    - w powtórzeń pętli wewnętrznej (linia 12-13)
      - 1 porównanie (linia 12)
      - 1 inkrementacja (linia 13)
      - 1 podstawienie (linia 13)
    - 1 porównanie (linia 14)
    - 1 podstawienie licznika pętli (linia 15)
    - w+1 sprawdzeń warunku pętli (linia 15)
    - w inkrementacji licznika pętli (linia 15)
    - w powtórzeń pętli wewnętrznej (linie 16-19)
      - 1 porównanie (linia 16)
      - 1 przypisanie (linia 17 lub 19)

- 1 inkrementacja (linia 20)
- 1 podstawienie (linia 20)

T(n,m,w) = n+m-1+nm+1+1+n+m+(n+m-1)*(1+1+1+ mn/(m+n-1) - w+1 + mn/(m+n-1)-w +1+ (mn/(m+n-1))-w)*(1+1+w+1+w(1+1)+1+1+w+1+w(1+1)=2n+2m+1+nm+(m+n-1)((mn)/(m+n-1)+ (mn)/(m+n+1)-2w+4)*(6+6w)< c*w*nm

W notacji dużego O: O(w*nm)

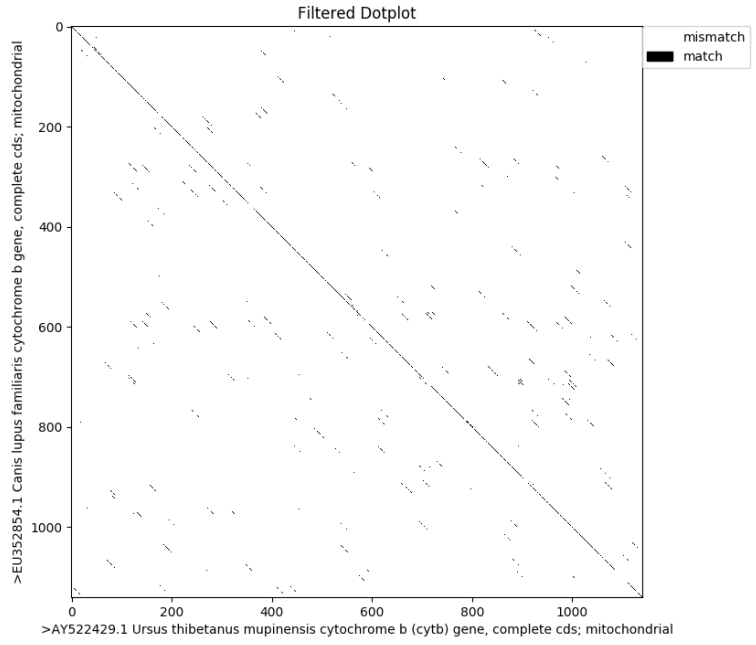
1.4 Filtracja macierzy kropkowej, analiza złożoności obliczeniowej przestrzennej:

- nm – macierzy wejściowej (linia 3)
- 1 zmienna do przechowywania wielkości okna (linia 1)
- 1 zmienna do przechowywania wartości błędu (linia 1)
- zmienna przechowująca wszystkie przekątne macierzy, ze względu na ich różne długości do analizy bierzemy ich średnią długość →n * (mn/(n+m-1)) (linia 4)
- 1 zmienna przechowująca licznik przekątnych (linia 5)
- 4 zmienne przechowujące liczniki pętli (linie 7,9,11,15)
- 1 zmienna przechowująca licznik służący do naliczania zgodności fragmentu sekwencji (linia 10)
- nm – rozmiar przefiltrowanej macierzy kropkowej (linia 21)

S(n,m) = nm+1+1+ n * (mn/(n+m-1))+1+4+1+nm=2nm+8+n(mn/(n+m-1))< c*(n*nm)/(n+m)

W notacji dużego O: O(w*nm(n*nm)/(n+m))

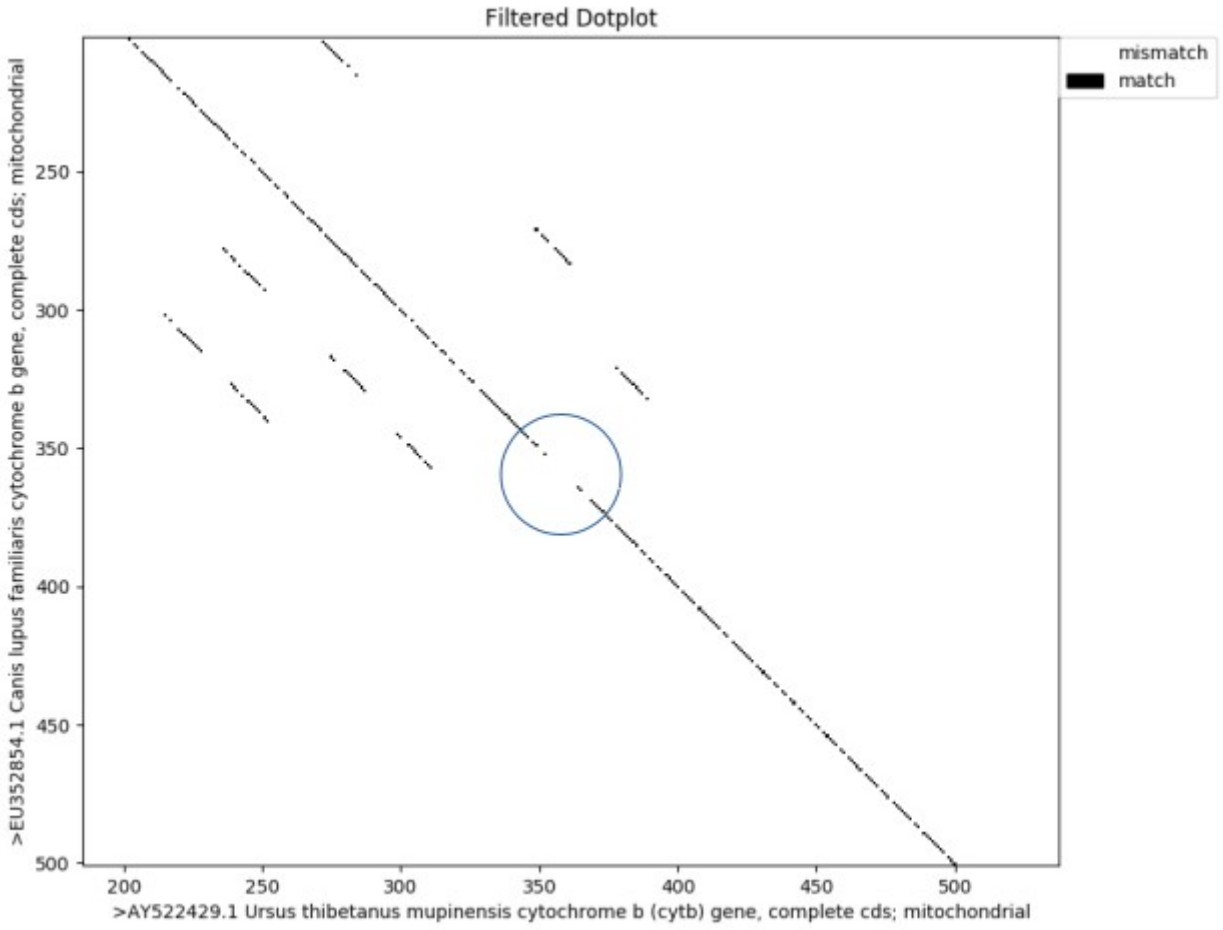
2. Porównanie sekwencji ewolucyjnie powiązanych:
DNA cytochromu b Niedźwiedzia tybetańskiego i wilka



Rysunek 1: Dotplot, window 13, error 3

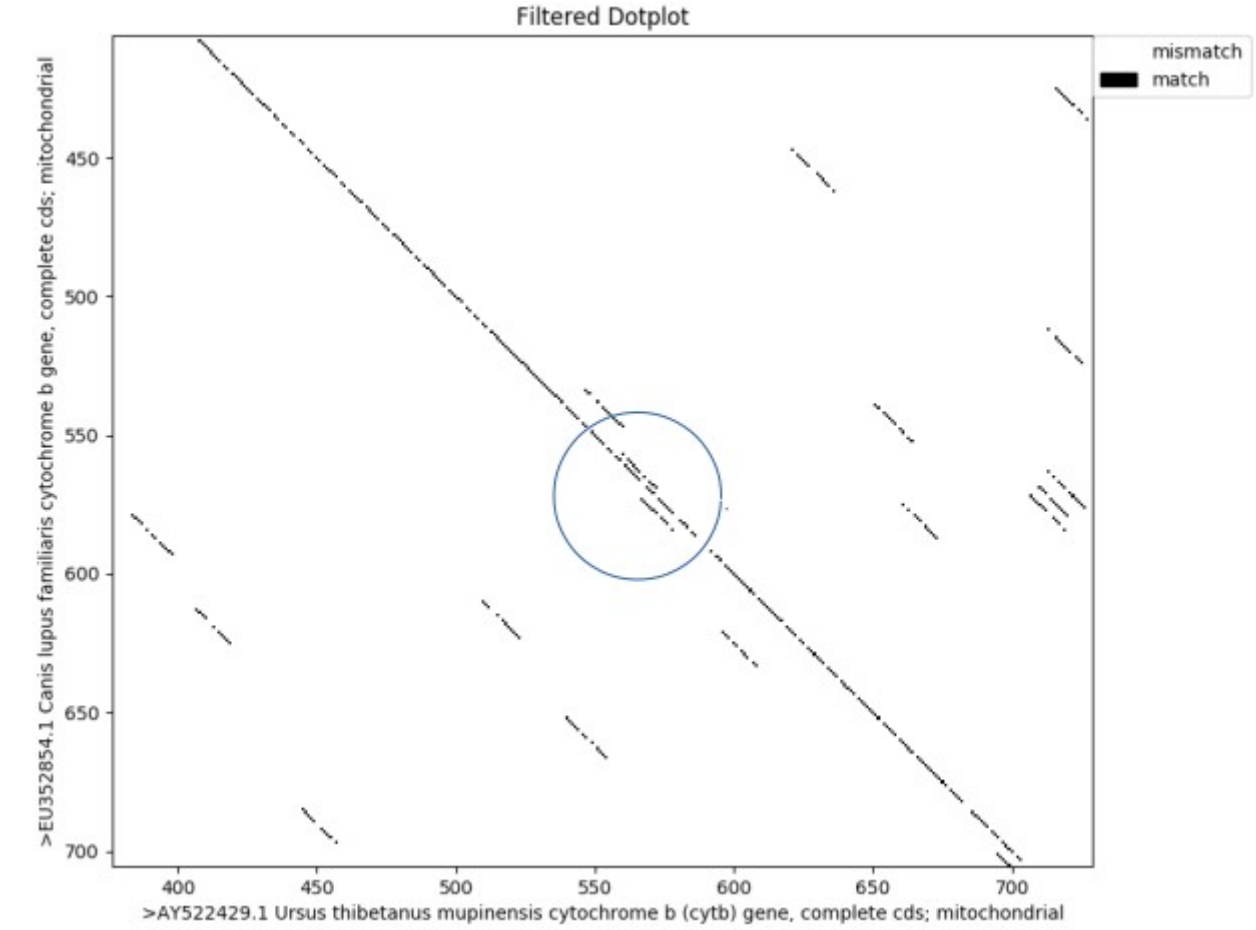
Na podstawie Rysunku 1, możemy stwierdzić iż sekwencje kodujące cytochrom b dla psa i niedźwiedzia tybetańskiego są bardzo podobne. Co wskazuje na bardzo bliskie pokrewieństwo gatunków. Sekwencje niedźwiedzia tybetańskiego i wilka charakteryzują się analogią fragmentów nukleotydowych w większej części długości łańcucha. Obydwa gatunki pochodzą z rodziny psowatych. Na podstawie powyższego porównania, możemy pokazać również, niektóre z mutacji genetycznych:

- Substytucja



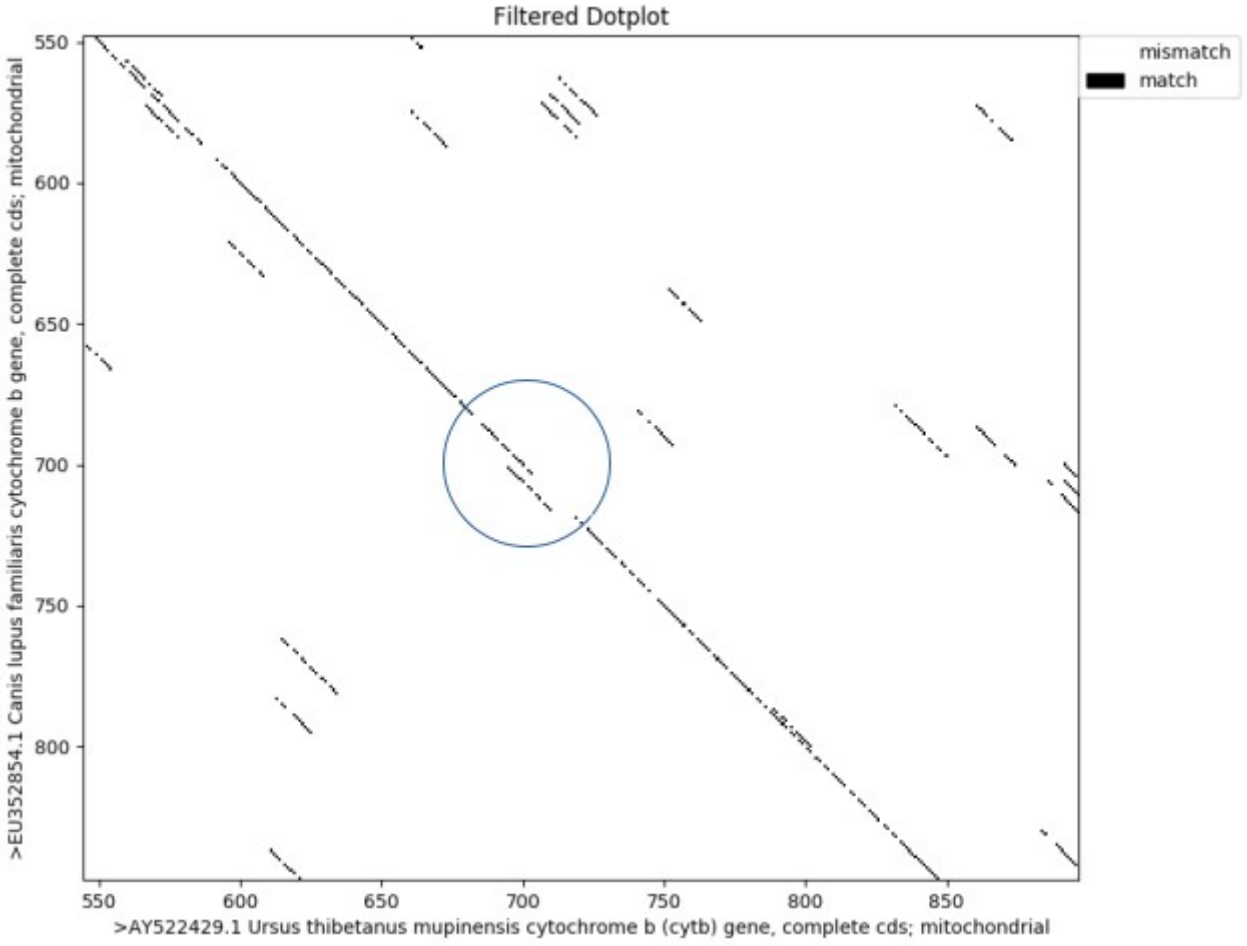
Rysunek 2: Dotplot, window 13, error 3, substytucja

- Duplikacja



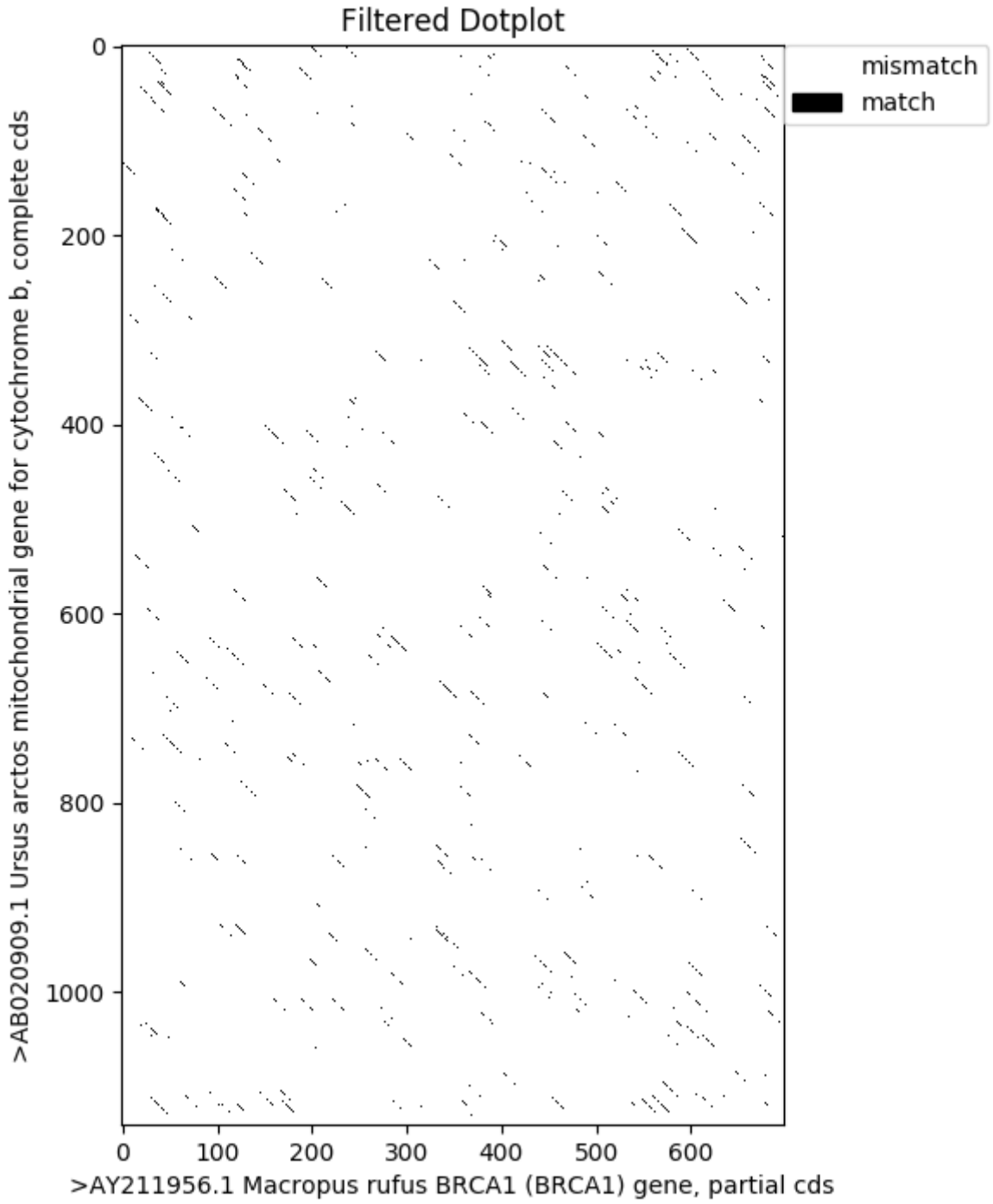
Rysunek 3: Dotplot, window 13, error 3, duplikacja

- Insercja lub delecja



Rysunek 4: Dotplot, window 13, error 3, insefcja lub delecja

3. Porównanie sekwencji ewolucyjnie niepowiązanych:
DNA cytochromu b niedźwiedzia arktycznego i genu BRCA1 kangura rudego:



Rysunek 5: Dotplot, window 13, error 3, sekwencje niepowiązane genetycznie

Na podstawie Rysunku 5, możemy stwierdzić iż sekwencje kodujące cytochrom b niedźwiedzia arktycznego i genu BRCA1 kangura rudego nie są ze sobą ewolucyjnie powiązane. Na rysunku nie ma przekątnej, a podobieństwa krótkich podsekwencji możemy uznać za przypadkowe.