Dot plot

Dot plot is a `D=MxN` matrix that denotes similarity of two sequences `A` and `B`. `D[i,j] = 1` if and only if `A[i]==B[j]`. Otherwise, `D[i,j] = 0`. We call the visual representation of the matrix the 'dot plot', because usually the 1s are replaced with dots and 0s are replaced with blanks.

Dot plot is used for serveral tasks:

- finding common substrings
- finding reversed substrings
- finding the longest common substring
- finding displacements
- finding repeats
- visualizing similarity of two sequences

In [1]:
```
import sys

!{sys.executable} -m pip install pypng

import png
import numpy as np

# Stringency equal to 0 denotes scaled colors

def get_color(dotplot, y, x, window, stringency=0):
    dots=0
    for i in range(min(window,len(dotplot)-y,len(dotplot[0])-x)):
        dots+=dotplot[y+i,x+i]

    degree=int(255*(1-dots/window)) if stringency == 0 else (0 if dots >= stringency else 255)
    return degree

def show_dotplot(dotplot, window=1, stringency=0, shift=0,filename='output.png', verbose = True):
    h = len(dotplot)
    w = len(dotplot[0])

    canvas_width=w # change to compress (or enlarge) the image
    canvas_height=h # change to compress (or enlarge) the image
    canvas = np.zeros((canvas_width, canvas_height), dtype=int)

    square_height=canvas_height//h
    square_width=canvas_width//w

    image = np.zeros(dotplot.shape, dtype=int)

    if verbose:
        print('Computing {}x{} dotplot.'.format(w,h))
        print('Window size: {}'.format(window))
        print('Stringency: {}'.format(stringency))
        print('Linear shift in color values: {}'.format(shift))

    for y in range(h):
        for x in range(w):
            image[y,x]=get_color(dotplot, y, x, window, stringency) # get color a of a dot in the dotplot

    # ratios of canvas and dotplot size
    wratio = int(w/canvas_width) + (1 if w%canvas_width != 0 else 0)
    hratio = int(h/canvas_height) + (1 if h%canvas_height != 0 else 0)

    if(wratio == 1 and hratio == 1): # both are the same - use the data we have
        canvas = image
    elif wratio <= 1 and hratio <= 1:
        if verbose:
            print('Enlarging image.')

        for y in range(h):
            for x in range(w):
                col=image[y,x]

                col+=int(shift*(col-255/2)) # linear shift towards bounds
                col=max(col,0)
                col=min(col,255)

                for i in range(square_width): # fill a square with our data
                    for j in range(square_height):
                        canvas[square_width*x+i,square_height*y+j] = col
    else:
        if verbose:
            print('Compressing image.')
        k = 1 # size of 'pixels'
        wratio *= k
        hratio *= k
        for y in range(0,h,hratio):
            for x in range(0,w,wratio):
                i=y//hratio
                j=x//wratio
                # computed mean of values in a square that will be contained in a pixel
                col=int(round(np.sum(image[y:min(h,y+hratio),x:min(w,x+wratio)])/(hratio*wratio)))

                col+=int(shift*(col-255/2)) # linear shift towards bounds
                col=max(col,0)
                col=min(col,255)

                for l in range(k):
                    for m in range(k):
                        canvas[k*j+l,k*i+m] = col

    if verbose:
        print('Saving image as {}.'.format(filename))
    png.from_array(canvas, 'L;8').save(filename) # save the image into a file
    if verbose:
        print('Image saved.')

def create_dotplot(A, B):
    size=(len(A), len(B))
    D=np.zeros(size, dtype=int) # create a matrix filled with zeros
    for i in range(size[0]):
```

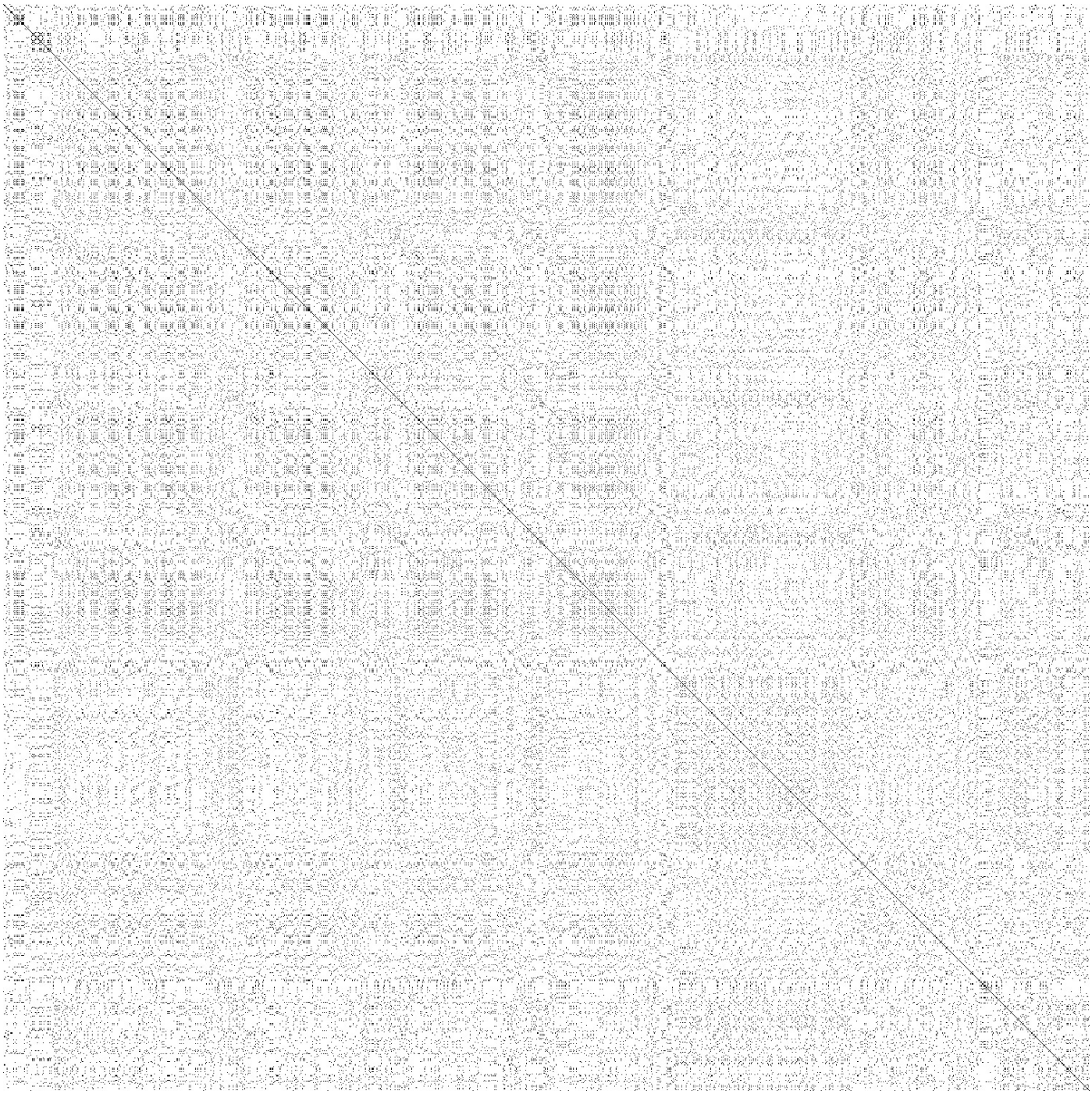
```
Requirement already satisfied: pypng in c:\users\jakub\anaconda3\lib\site-packages (0.0.18)
You are using pip version 18.0, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

In [2]:
```
seq='\
MAAPSRTTLMPPPFRLQLRLLILPILLLLRHDAVHAEPYSGGFGSSAVSSGGLGSVGIHIPGGGVGVITE\
ARCPRVCSCTGLNVDCSHRGLTSVPRKISADVERLELQGNNLTVIYETDFQRLTKLRMLQLTDNQIHTIE\
RNSFQDLVSLERLRLNNNRLKAIPENFVTSSASLLRLDISNNVITTVGRRVFKGAQSLRSLQLDNNQITC\
LDEHAFKGLVELEILTLNNNNLTSLPHNIFGGLGRLRALRLSDNPFACDCHLSWLSRFLRSATRLAPYTR\
CQSPSQLKGQNVADLHDQEFKCSGLTEHAPMECGAENSCPHPCRCADGIVDCREKSLTSVPVTLPDDTTE\
LRLEQNFITELPPKSFSSFRRLRRIDLSNNNISRIAHDALSGLKQLTTLVLYGNKIKDLPSGVFKGLGSL\
QLLLLNANEISCIRKDAFRDLHSLSLLSLYDNNIQSLANGTFDAMKSIKTVHLAKNPFICDCNLRWLADY\
LHKNPIETSGARCESPKRMHRRRIESLREEKFKCSWDELRMKLSGECRMDSDCPAMCHCEGTTVDCTGRG\
LKEIPRDIPLHTTELLLNDNELGRISSDGLFGRLPHLVKLELKRNQLTGIEPNAFEGASHIQELQLGENK\
IKEISNKMFLGLHQLKTLNLYDNQISCVMPGSFEHLNSLTSLNLASNPFNCNCHLAWFAEWLRKKSLNGG\
AARCGAPSKVRDVQIKDLPHSEFKCSSENSEGCLGDGYCPPSCTCTGTVVRCSRNQLKEIPRGIPAETSE\
LYLESNEIEQIHYERIRHLRSLTRLDLSNNQITILSNYTFANLTKLSTLIISYNKLQCLQRHALSGLNNL\
RVLSLHGNRISMLPEGSFEDLKSLTHIALGSNPLYCDCGLKWFSDWIKLDYVEPGIARCAEPEQMKDKLI\
LSTPSSSFVCRGRVRNDILAKCNACFEQPCQNQAQCVALPQREYQCLCQPGYHGKHCEFMIDACYGNPCR\
NNATCTVLEEGRFSCQCAPGYTGARCETNIDDCLGEIKCQNNATCIDGVESYKCECQPGFSGEFCDTKIQ\
FCSPEFNPCANGAKCMDHFTHYSCDCQAGFHGTNCTDNIDDCQNHMCQNGGTCVDGINDYQCRCPDDYTG\
KYCEGHNMISMMYPQTSPCQNHECKHGVCFQPNAQGSDYLCRCHPGYTGKWCEYLTSISFVHNNSFVELE\
PLRTRPEANVTIVFSSAEQNGILMYDGQDAHLAVELFNGRIRVSYDVGNHPVSTMYSFEMVADGKYHAVE\
LLAIKKNFTLRVDRGLARSIINEGSNDYLKLTTPMFLGGLPVDPAQQAYKNWQIRNLTSFKGCMKEVWIN\
HKLVDFGNAQRQQKITPGCALLEGEQQEEEDDEQDFMDETPHIKEEPVDPCLENKCRRGSRCVPNSNARD\
GYQCKCKHGQRGRYCDQGEGSTEPPTVTAASTCRKEQVREYYTENDCRSRQPLKYAKCVGGCGNQCCAAK\
IVRRRKVRMVCSNNRKYIKNLDIVRKCGCTKKCY'

dotplot=create_dotplot(seq, seq)
show_dotplot(dotplot,window=1,stringency=0,shift=0,filename='dots_1_0_0.png')
show_dotplot(dotplot,window=7,stringency=0,shift=0,filename='dots_7_0_0.png')
show_dotplot(dotplot,window=7,stringency=0,shift=0.8,filename='dots_7_0_0,8.png')
show_dotplot(dotplot,window=7,stringency=4,shift=0,filename='dots_7_4_0.png')
show_dotplot(dotplot,window=12,stringency=0,shift=0,filename='dots_12_0_0.png')
show_dotplot(dotplot,window=12,stringency=0,shift=0.8,filename='dots_12_0_0,8.png')
show_dotplot(dotplot,window=12,stringency=7,shift=0,filename='dots_12_7_0.png')
show_dotplot(dotplot,window=20,stringency=0,shift=0,filename='dots_20_0_0.png')
show_dotplot(dotplot,window=20,stringency=0,shift=0.8,filename='dots_20_0_0,8.png')
show_dotplot(dotplot,window=20,stringency=0,shift=0.8,filename='dots_20_0_2.png')
show_dotplot(dotplot,window=20,stringency=0,shift=0.8,filename='dots_20_0_5.png')
show_dotplot(dotplot,window=20,stringency=5,shift=0,filename='dots_20_5_0.png')
show_dotplot(dotplot,window=20,stringency=10,shift=0,filename='dots_20_10_0.png')
show_dotplot(dotplot,window=20,stringency=15,shift=0,filename='dots_20_15_0.png')
```

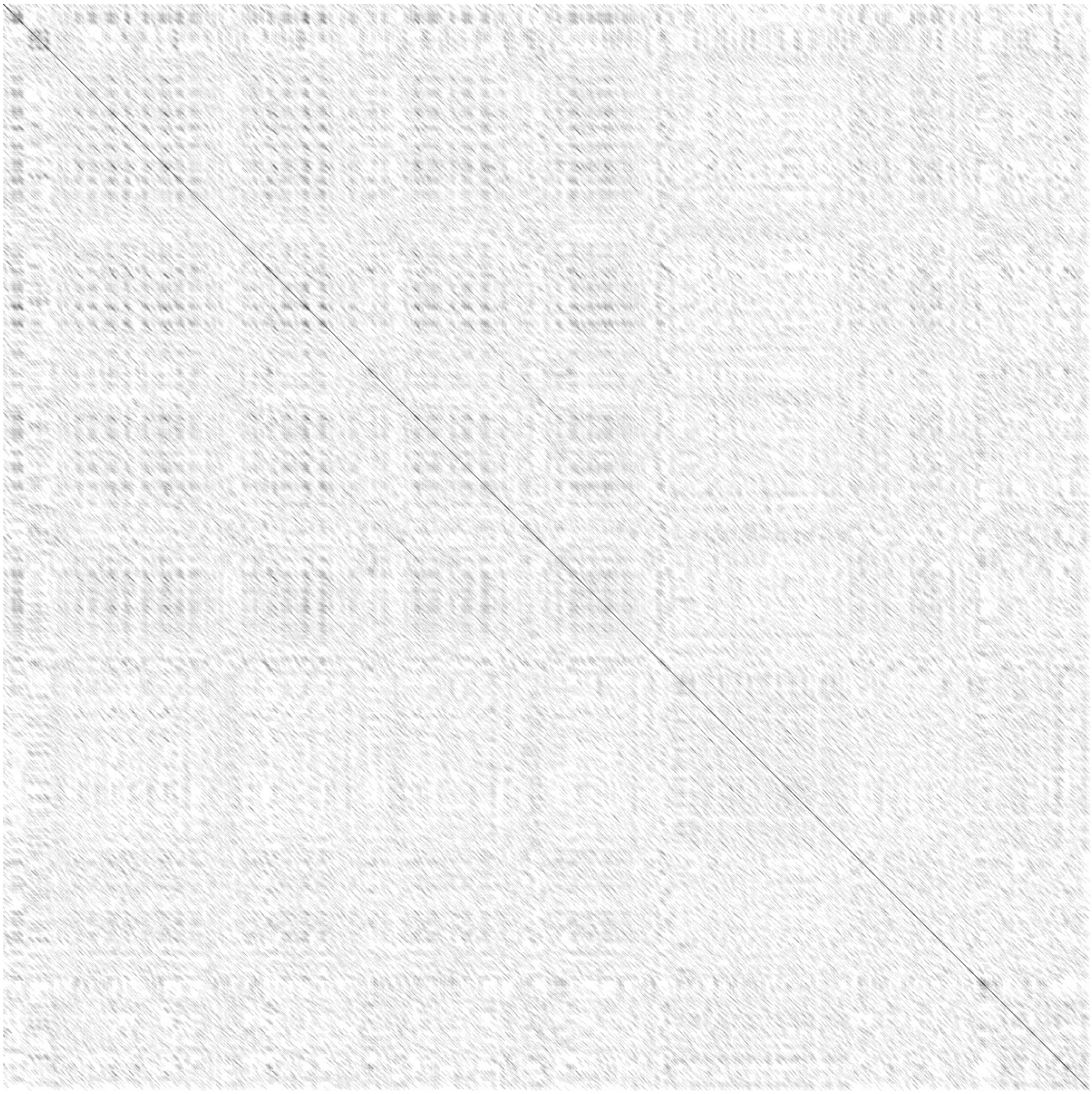
```
Computing 1504x1504 dotplot.
Window size: 1
Stringency: 0
Linear shift in color values: 0
Saving image as dots_1_0_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 7
Stringency: 0
Linear shift in color values: 0
Saving image as dots_7_0_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 7
Stringency: 0
Linear shift in color values: 0.8
Saving image as dots_7_0_0,8.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 7
Stringency: 4
Linear shift in color values: 0
Saving image as dots_7_4_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 12
Stringency: 0
Linear shift in color values: 0
Saving image as dots_12_0_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 12
Stringency: 0
Linear shift in color values: 0.8
Saving image as dots_12_0_0,8.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 12
Stringency: 7
Linear shift in color values: 0
Saving image as dots_12_7_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 0
Linear shift in color values: 0
Saving image as dots_20_0_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 0
Linear shift in color values: 0.8
Saving image as dots_20_0_0,8.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 0
Linear shift in color values: 0.8
Saving image as dots_20_0_2.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 0
Linear shift in color values: 0.8
Saving image as dots_20_0_5.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 5
Linear shift in color values: 0
Saving image as dots_20_5_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 10
Linear shift in color values: 0
Saving image as dots_20_10_0.png.
Image saved.
Computing 1504x1504 dotplot.
Window size: 20
Stringency: 15
Linear shift in color values: 0
Saving image as dots_20_15_0.png.
Image saved.
```

Some results:

dots_1_0_0



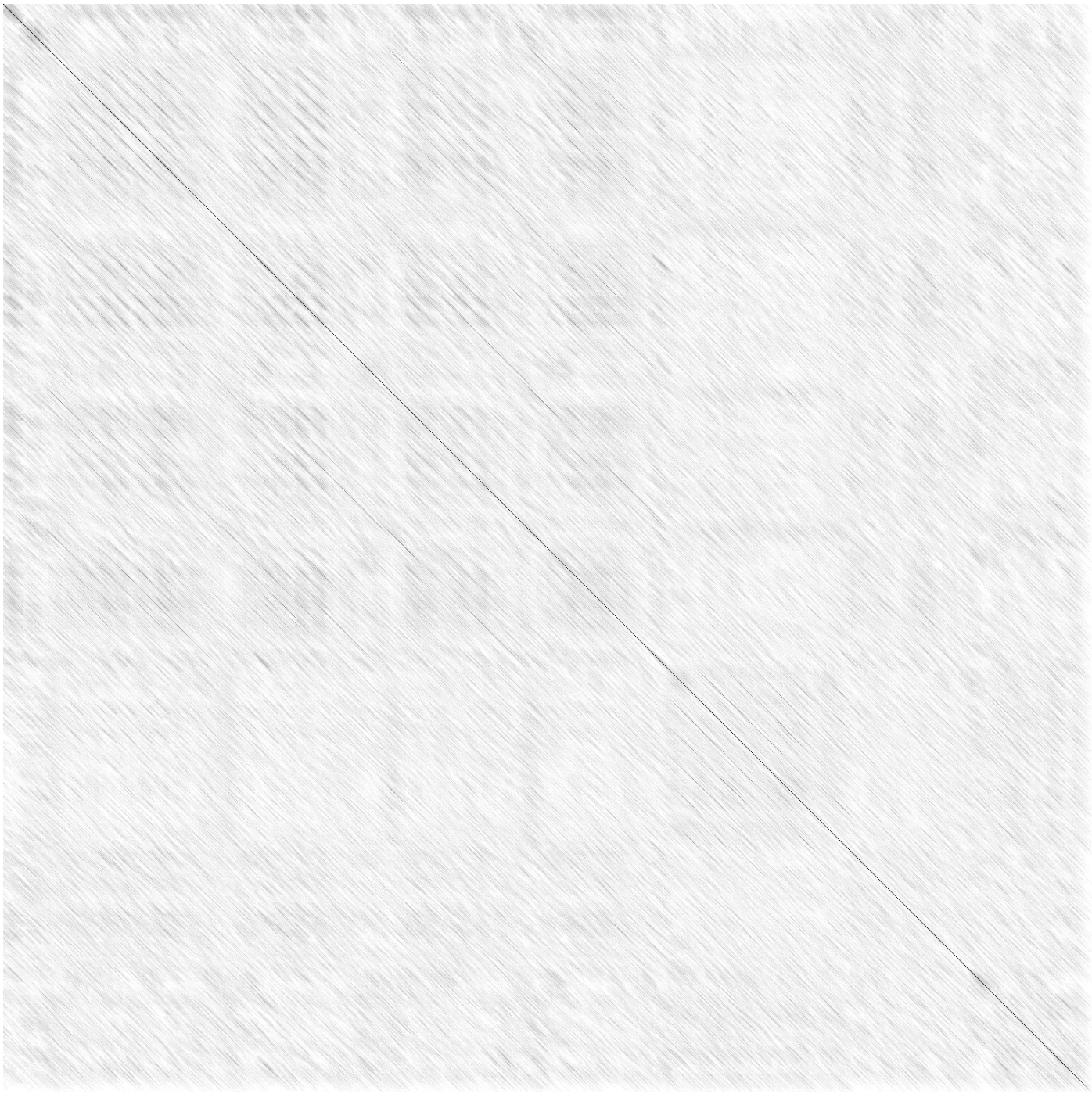
dots_7_0_0



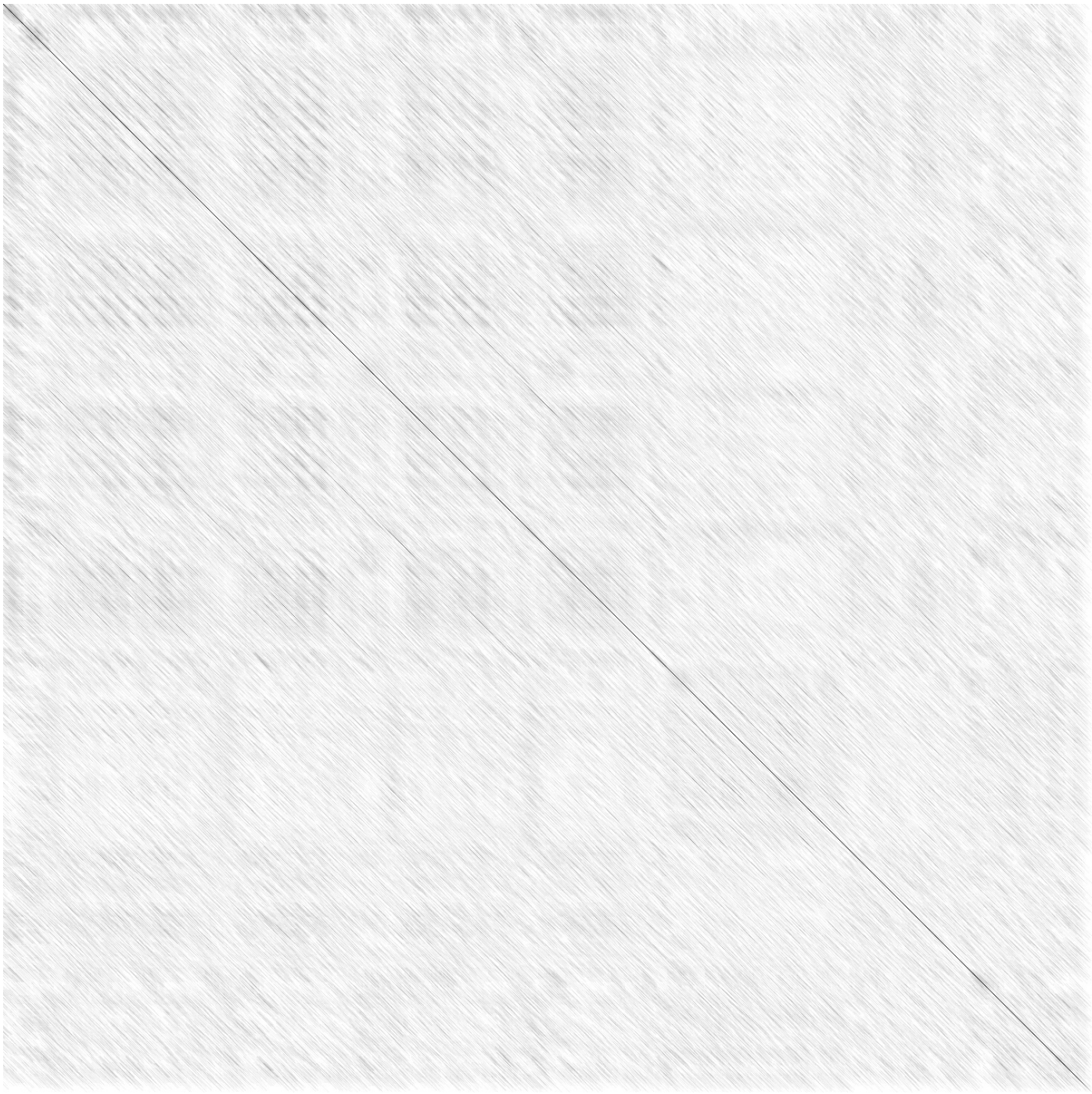
dots_7_4_0

dots_12_7_0

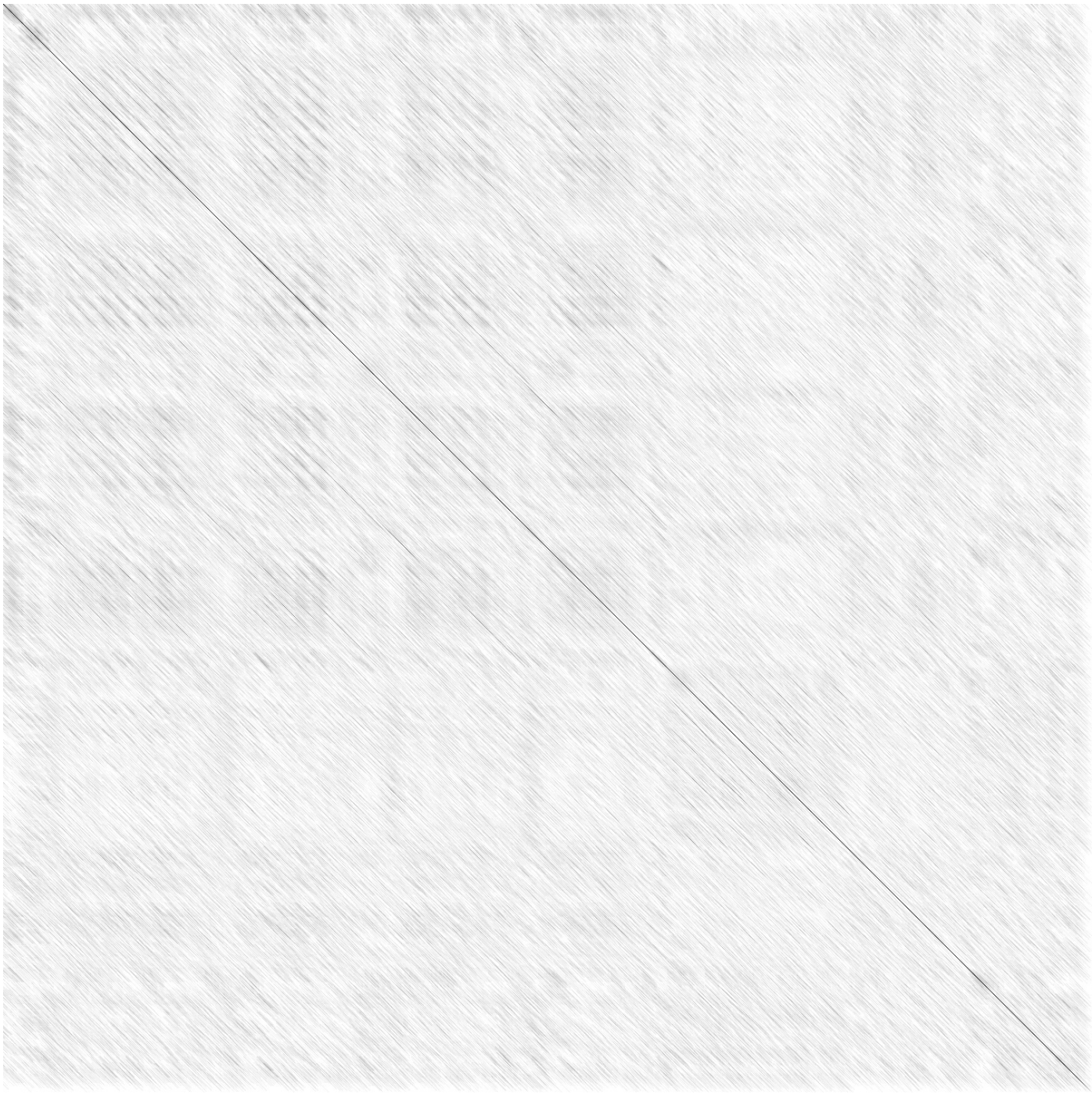
dots_20_0_0



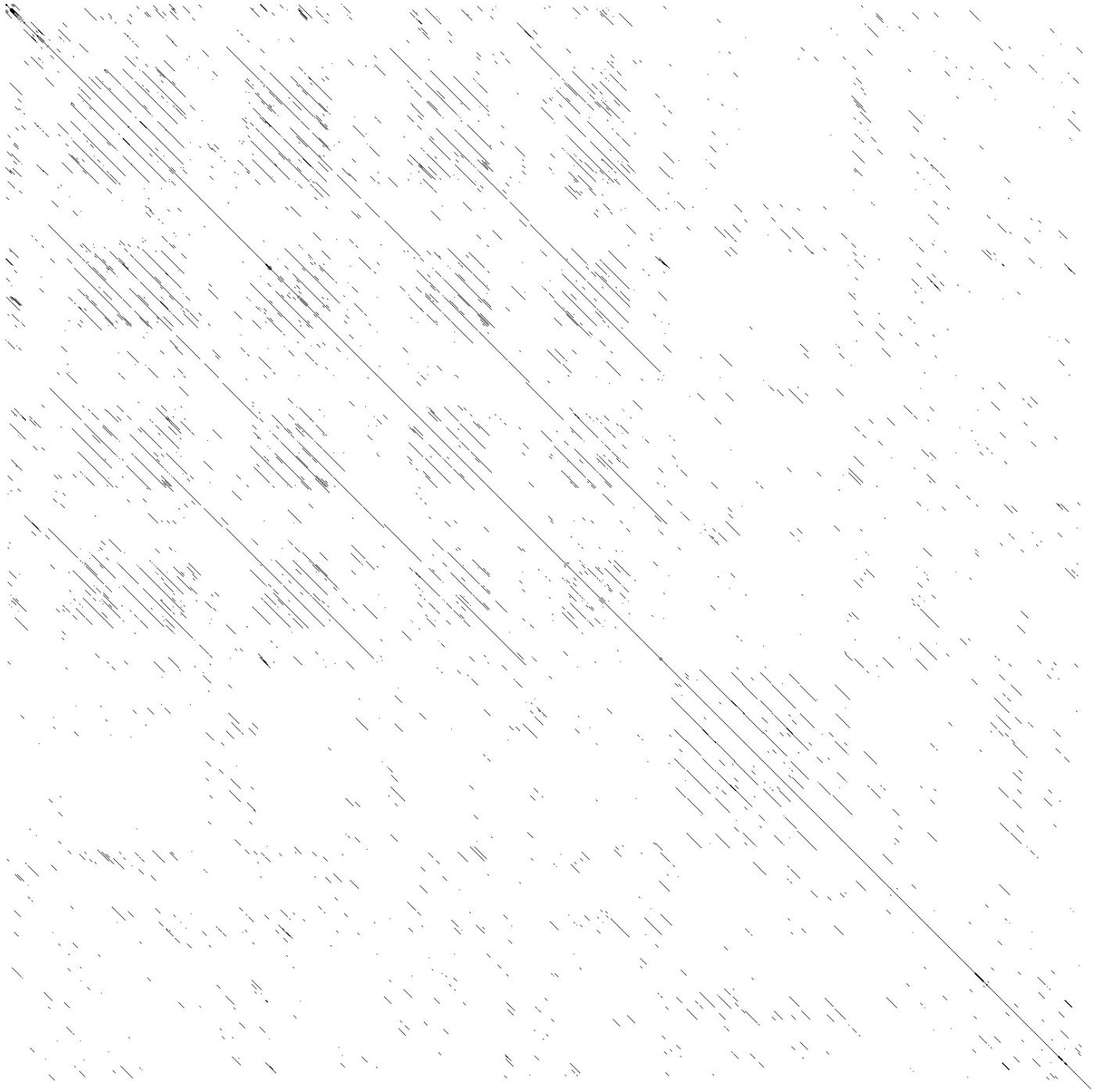
dots_20_0_0,8



dots_20_0_5

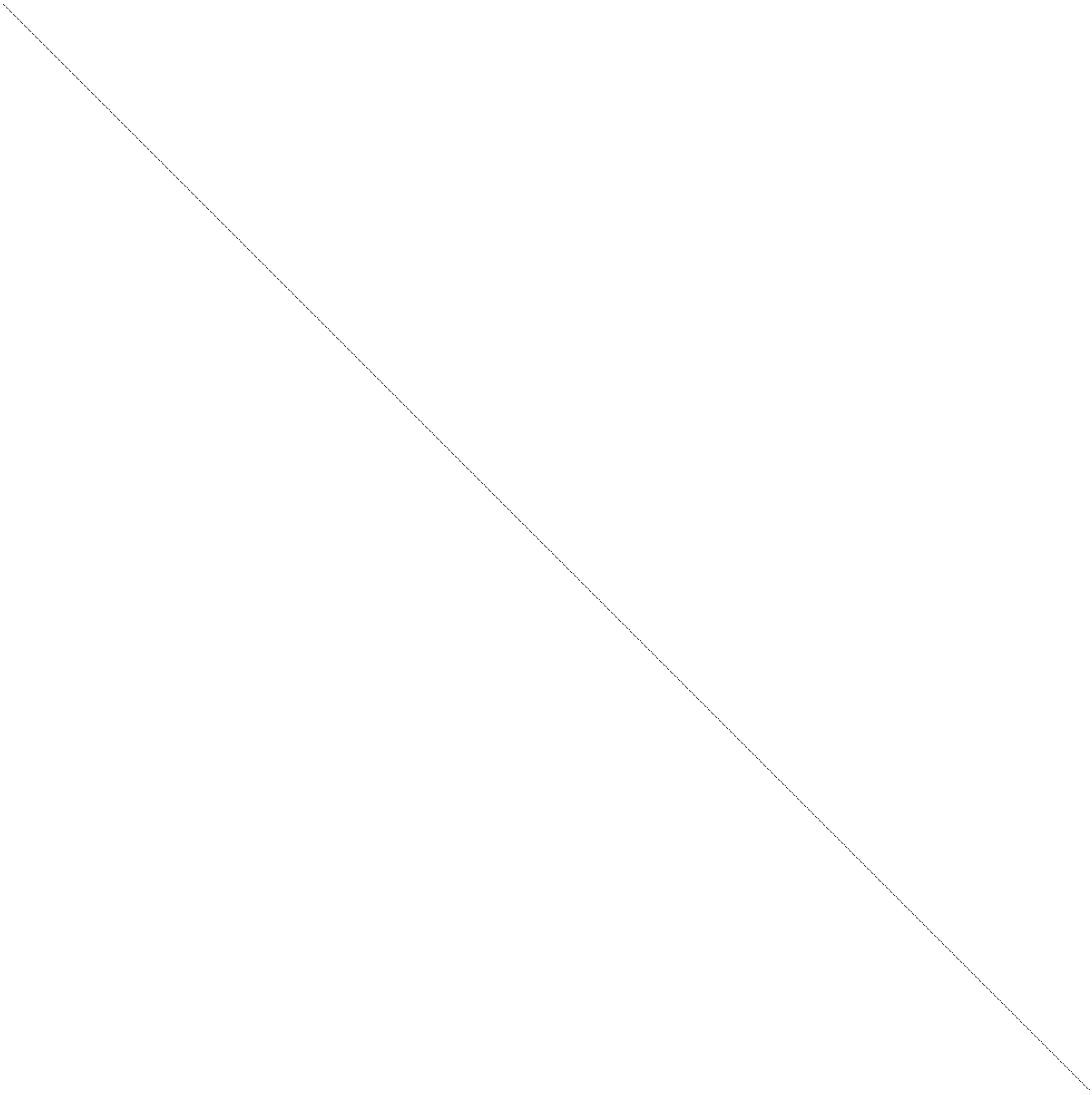


dots_20_5_0



dots_20_10_0

dots_20_15_0



From the images above, we can clearly see, that we compared a sequence against itself - this is visible from the main diagonal which is filled whole.

Beside that, we can see, that there are no long similar sequences. When we lower the window size or lower the stringency, two squares around the main diagonal start to appear. Big one at the start of the sequence and second, smaller one around 3/4 of the sequence.

> Parallel lines indicate tandem repeats of a larger motif in both sequences, e.g. $(AGCTCTGAC)_{20}$, so called minisatellite patterns. The distance between the diagonals equals the distance of the motif.
>
> Cited from text of Jan Schulz Introduction to dot-plots.

This indicates that there are minisatellite patters which include larger number mistmatches - around 10-15 edit operations is needed for sequences of length 20 for minisatellites to be clearly visible.

Based on the information from the site https://www.ncbi.nlm.nih.gov/protein/P24014#sequence_P24014.2 (https://www.ncbi.nlm.nih.gov/protein/P24014#sequence_P24014.2) I would conclude that the smaller minisatellite consists of EGF-like domains which repeat a lot in positions 935-1173. The bigger minisatellite probably consists of LRR-like repeats that dominate the sequence in positions 64-920.

NM_000044

This sequence belongs to Homo sapiens and contains genom of androgen receptor.

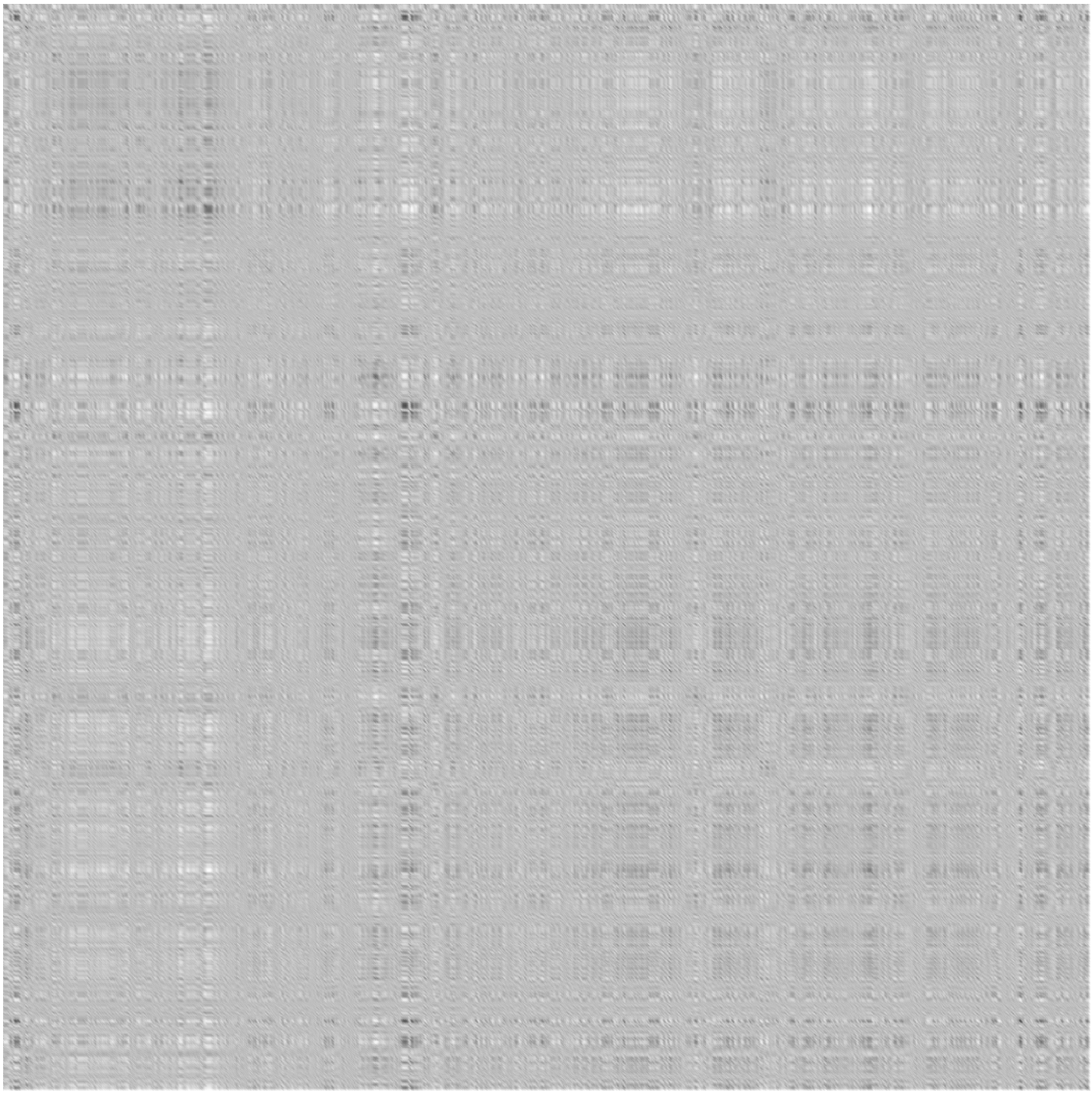
In [3]:
```
import gzip

def loadfasta(filename,verbose=0):
    """ Parses a classically formatted and possibly
        compressed FASTA file into a dictionary where the key
        for a sequence is the first part of its header without
        any white space; if verbose is nonzero then the identifiers
        together with lengths of the read sequences are printed"""
    if (filename.endswith(".gz")):
        fp = gzip.open(filename, 'rt')
    else:
        fp = open(filename, 'r')
    # split at headers
    # data = fp.read().split('>')
    data = fp.read()
    data = data.split('>')
    fp.close()
    # ignore whatever appears before the 1st header
    data.pop(0)
    # prepare the dictionary
    D = {}
    for sequence in data:
        lines = sequence.split('\n')
        header = lines.pop(0).split()
        key = header[0]
        D[key] = ''.join(lines)
        if verbose:
            print("Sequence %s of length %d read" % (key,len(D[key])))
    return D

seq = loadfasta('Arabidopsis.fasta')
seq.keys()
```

Out[3]:
```
dict_keys(['NM_000044.4'])
```

In [4]:
```
dotplot=create_dotplot(seq['NM_000044.4'], seq['NM_000044.4'])
show_dotplot(dotplot,window=25,stringency=0,shift=100,filename='nm_000044_25_0_100.png')
```

```
Computing 10070x10070 dotplot.
Window size: 25
Stringency: 0
Linear shift in color values: 100
Saving image as nm_000044_25_0_100.png.
Image saved.
```

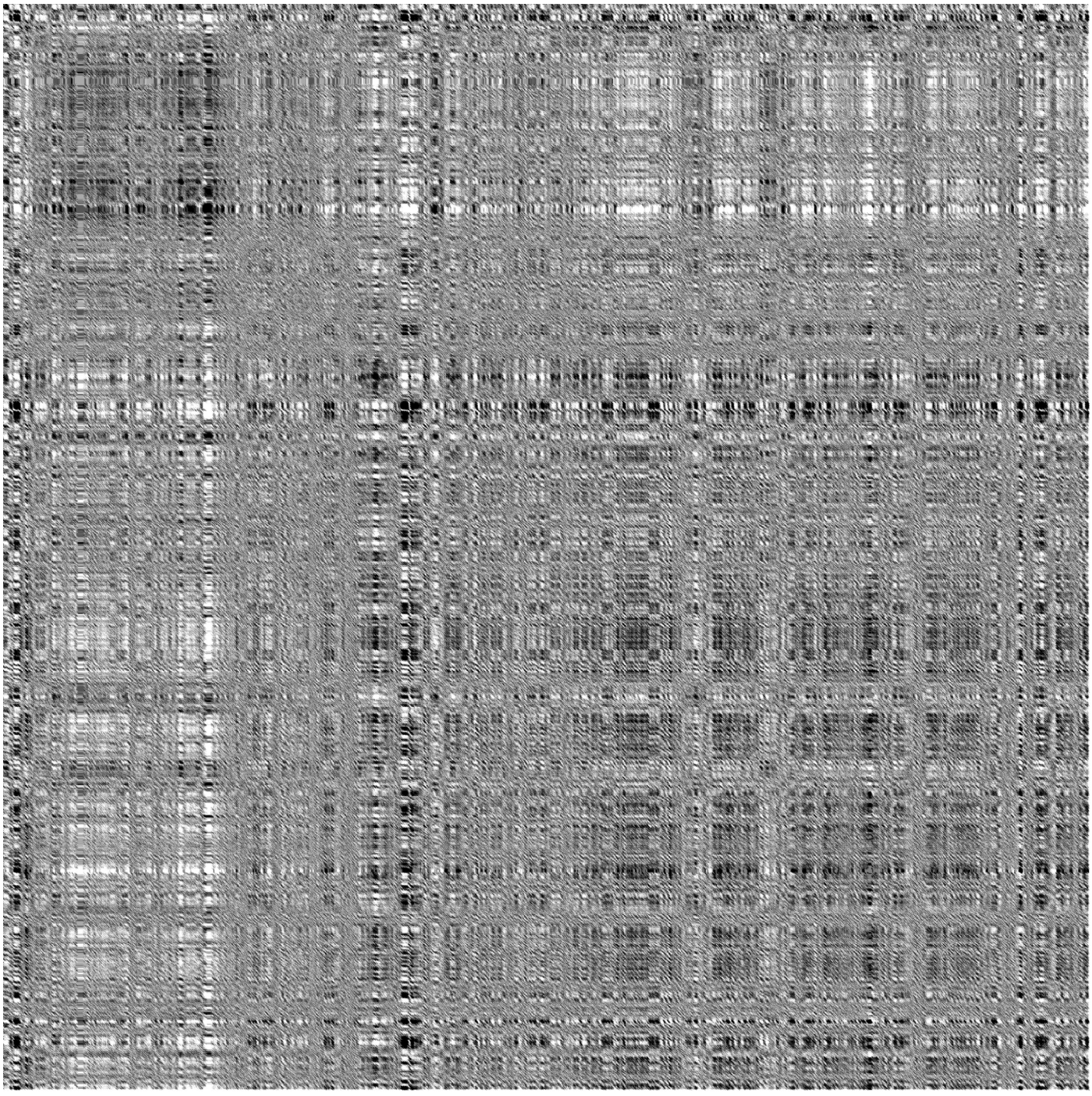
nm_000044_25_0_100



In [5]:
```
show_dotplot(dotplot,window=25,stringency=7,shift=0,filename='nm_000044_25_7_0.png')
```

```
Computing 10070x10070 dotplot.
Window size: 25
Stringency: 7
Linear shift in color values: 0
Saving image as nm_000044_25_7_0.png.
Image saved.
```

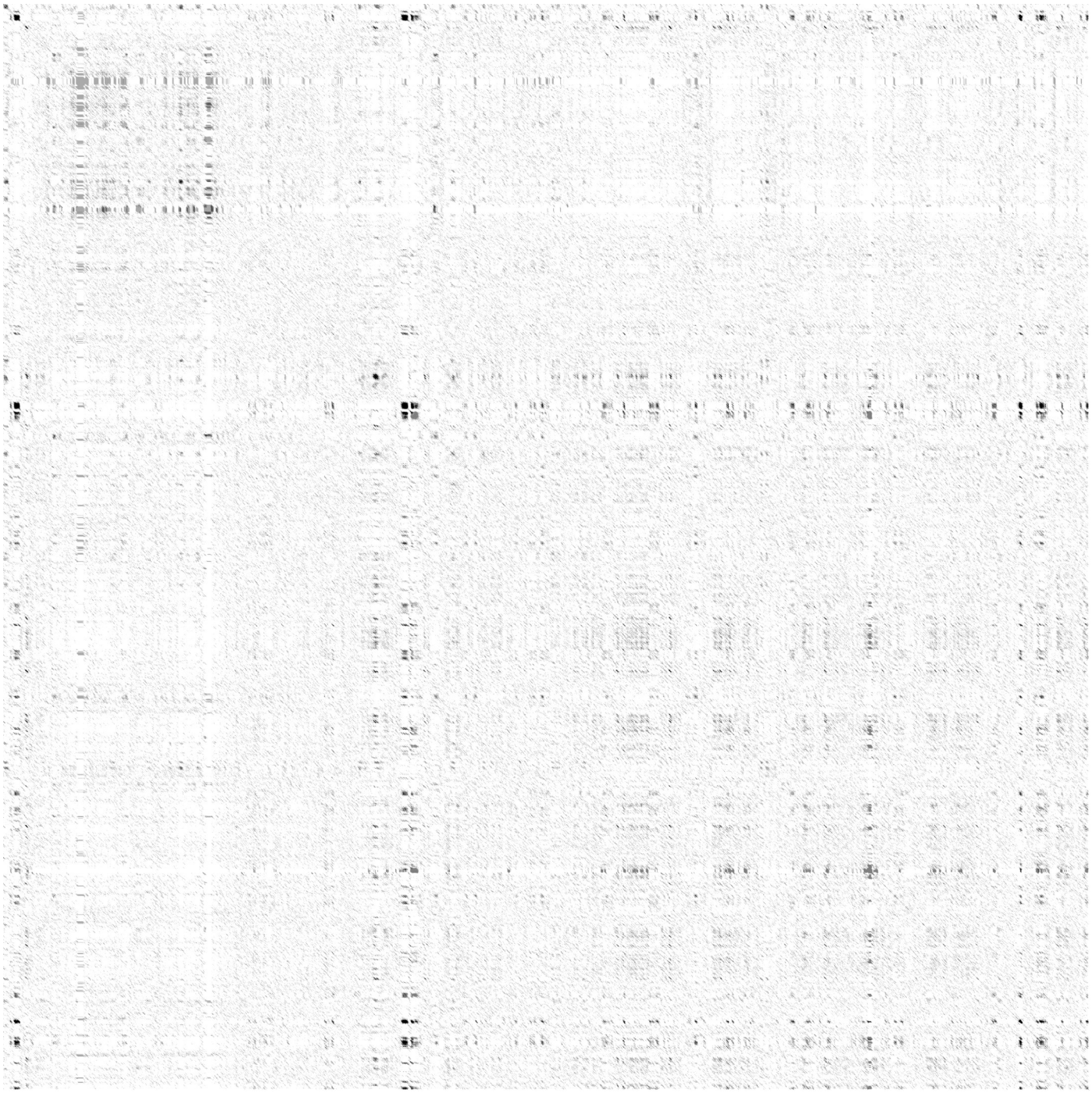
nm_000044_25_7_0



In [6]:
```
show_dotplot(dotplot,window=25,stringency=11,shift=0,filename='nm_000044_25_11_0.png')
```

```
Computing 10070x10070 dotplot.
Window size: 25
Stringency: 11
Linear shift in color values: 0
Saving image as nm_000044_25_11_0.png.
Image saved.
```

nm_000044_25_11_0

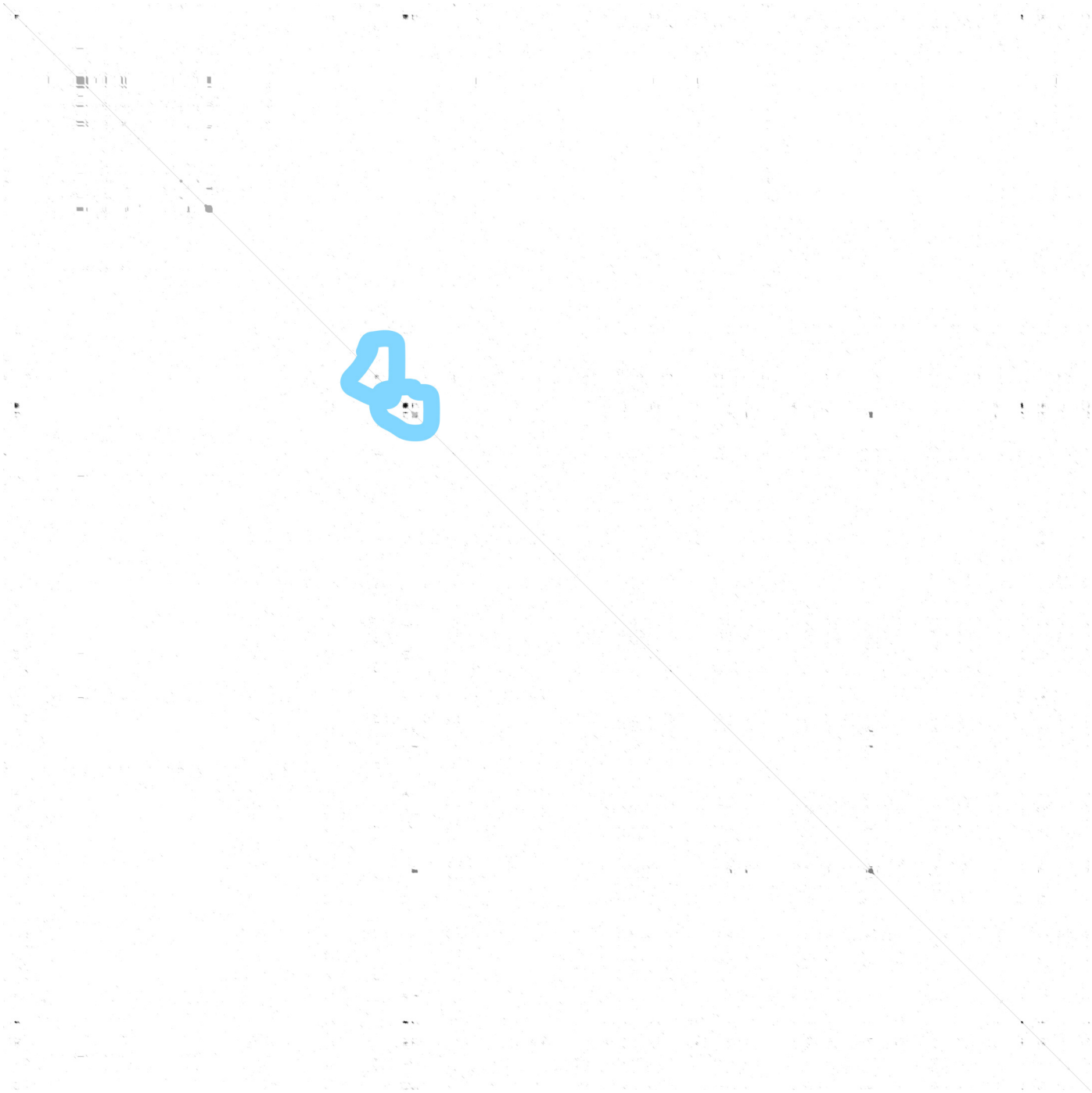


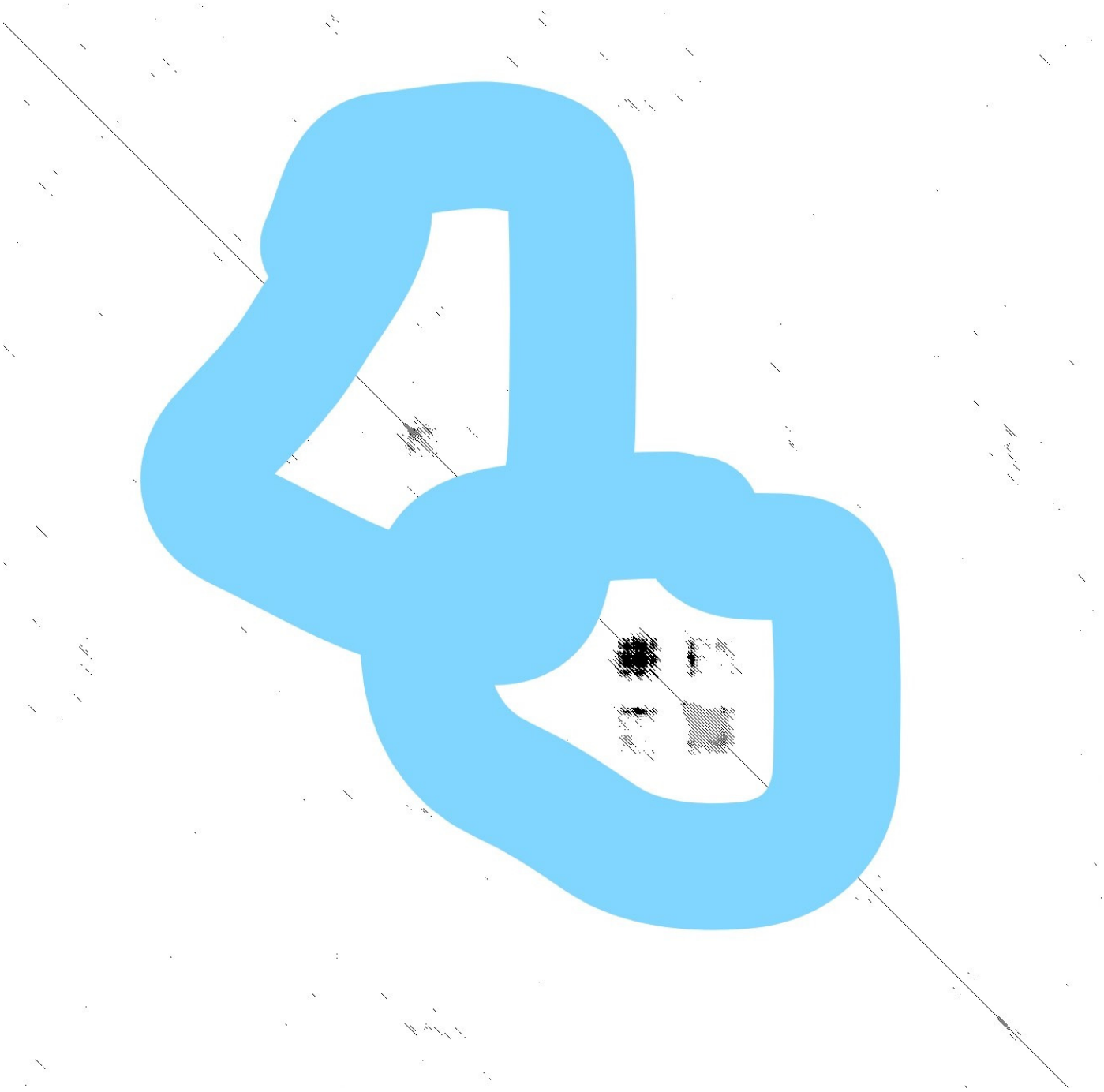
In [7]:
```
show_dotplot(dotplot,window=25,stringency=15,shift=0,filename='nm_000044_25_15_0.png')
```

```
Computing 10070x10070 dotplot.
Window size: 25
Stringency: 15
Linear shift in color values: 0
Saving image as nm_000044_25_15_0.png.
Image saved.
```

nm_000044_25_15_0

There are no highly similar parts right in the middle as can be observed from images above. However, there are some interesting patterns at about 2/5 of the sequence:





Especially the lower one consists of three patterns around the diagonal from which two lie right on the diagonal. Both form minisatellites.

If we open the FASTA file containing the genom at about the position of those patterns, two places immediatelly pop out:

- `TTCCTGAATTCTATTTGCTGGGCTTTTTTTTTCTCTTTCTCTCCTTTCTTTTTCTTCTTCCCTCCCTATCTAACCCTCCCATGGCACCTTCAGACTTTGCTTCCCATTGTGGCTCCTATCTGTGTTTTGAATG` - at positions 3470:3620
- `AAATCAAAACAAAAACAAGCAAACAAAAAAAAAAAGCAAAAACAAAACAAAAAATAAGCCAAAAAACCTTGCTAGTGTTTTTTCCTCAAAAATAAATAAATAAATAAATAAATACGTACATACATACACACAT` - at positions 3751:3942

The first sequence is thymine rich. There is also long sequence of thymines occasionaly interrupted by cytosine.

The second sequence is not that continous but adenine-rich and contains a subsequence of repeated `TAC`.

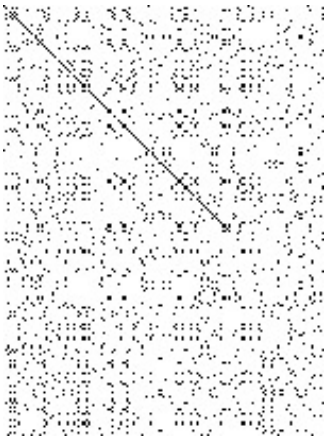
In the file we may also notice repetitive sequence `GGTGGTGGTGGGGGTGGTGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGGCGAGGCGGGAGCTGTAGC` at positions 1900:1987 which in our opinion corresponds to the pattern at the bottom right corner of a 'square' formed at the beginning of the sequence (see the zoomed out image above).

Globins

In [8]:
```
armadillo='\
MESPEPELIRQSWRVVSRSPLEHGTILFARLFDLEPDLLSLFQYNCRQFSSVEACLSSPEFLDHIRKVMV\
VIDTAVTNVEDLSSLEEYLAGLGRKHRAVGVKLSSFSEIQERQWDLLQVIRRKQPEKSRRVCRVKGGSSG\
RALQPDPRQHLDLGQVLLHQRREPRGAPSPPQYLGRTLSPGAPAVPPEQPSPLGHPLLPCAPR'
human='\
MERPEPELIRQSWRAVSRSPLEHGTVLFARLFALEPDLLPLFQYNCRQFSSPEDCLSSPEFLDHIRKVML\
VIDAAVTNVEDLSSLEEYLASLGRKHRAVGVKLSSFSTVGESLLYMLEKCLGPAFTPATRAAWSQLYGAV\
VQAMSRGWDGE'

dotplot=create_dotplot(armadillo,human)
for k in range(1,5):
    show_dotplot(dotplot,window=k,stringency=0,shift=0,filename='armahuman_{}_0_0.png'.format(k))
```

```
Computing 151x203 dotplot.
Window size: 1
Stringency: 0
Linear shift in color values: 0
Saving image as armahuman_1_0_0.png.
Image saved.
Computing 151x203 dotplot.
Window size: 2
Stringency: 0
Linear shift in color values: 0
Saving image as armahuman_2_0_0.png.
Image saved.
Computing 151x203 dotplot.
Window size: 3
Stringency: 0
Linear shift in color values: 0
Saving image as armahuman_3_0_0.png.
Image saved.
Computing 151x203 dotplot.
Window size: 4
Stringency: 0
Linear shift in color values: 0
Saving image as armahuman_4_0_0.png.
Image saved.
```

Results:




From the pictures above we can see that the longest similar subsequence starts at the beginning of the sequence and goes up to about 2/3 of human sequence.

From sequence analysis we can conclude that the similar parts are for human and armadillo respectively:

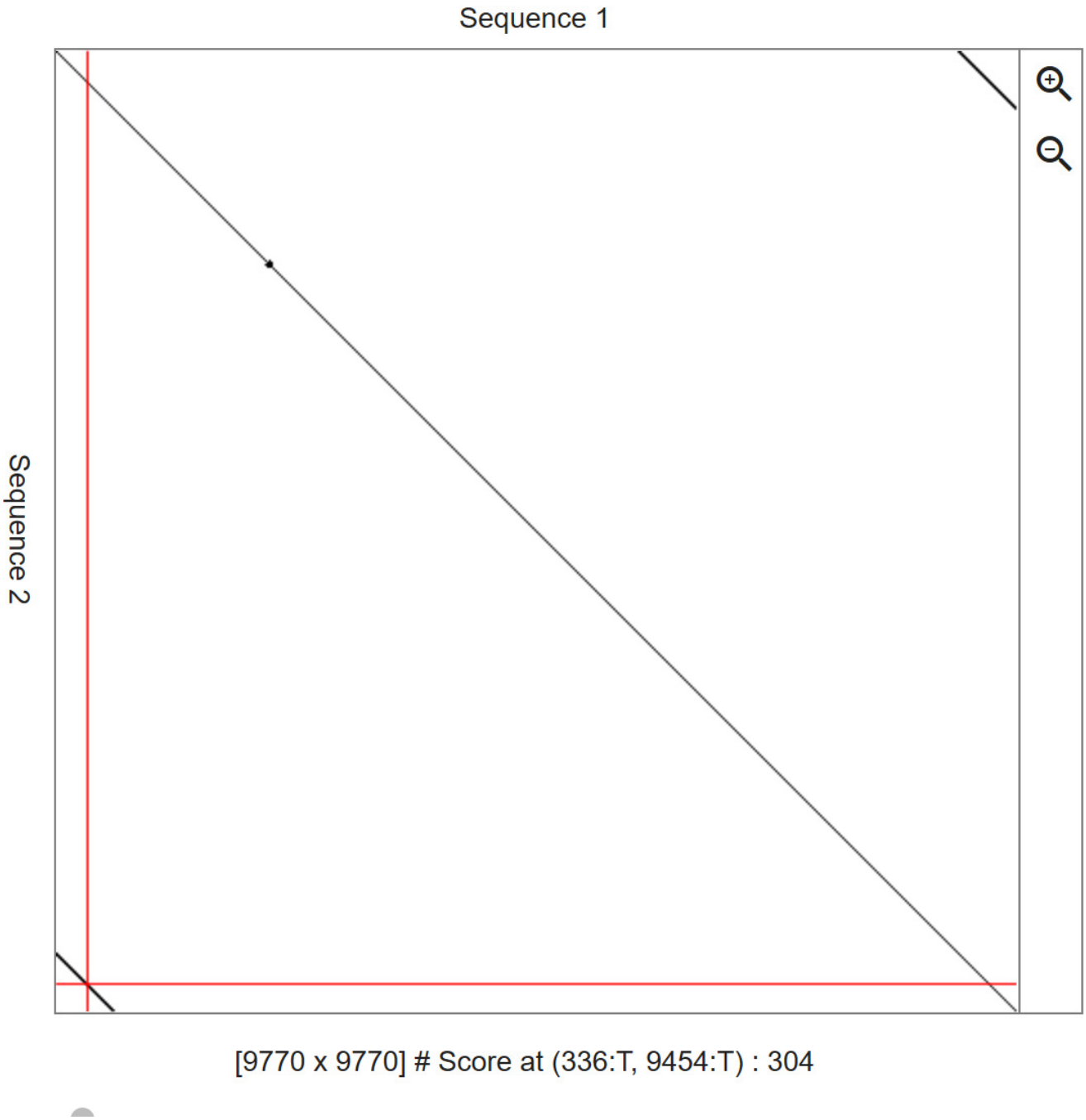
- MERPEPELIRQSWRAVSRSPLEHGTVLFARLFALEPDLLPLFQYNCRQFSSPEDCLSSPEFLDHIRKVMLVIDAAVTNVEDLSSLEEYLASLGRKHRAVGVKLSSFS
- MESPEPELIRQSWRVVSRSPLEHGTILFARLFDLEPDLLSLFQYNCRQFSSVEACLSSPEFLDHIRKVMVVIDTAVTNVEDLSSLEEYLAGLGRKHRAVGVKLSSFS

Positions of these sequences are 0:106.

In [9]:
```
armadillo='MESPEPELIRQSWRVVSRSPLEHGTILFARLFDLEPDLLSLFQYNCRQFSSVEACLSSPEFLDHIRKVMVVIDTAVTNVEDLSSLEEYLAGLGRKHRAVGVKLSSFS'
human='MERPEPELIRQSWRAVSRSPLEHGTVLFARLFALEPDLLPLFQYNCRQFSSPEDCLSSPEFLDHIRKVMLVIDAAVTNVEDLSSLEEYLASLGRKHRAVGVKLSSFS'

matches=len([1 for x in range(len(human)) if human[x] == armadillo[x]])
print('There were {} matches in sequences of length {}.'.format(matches,len(human)))
```

```
There were 97 matches in sequences of length 107.
```

HIV virus



From the dotplot above we can see that there are same segments of the sequence at the beginning and at the end of the squence. This means that there is a prefix of the sequence that is the same as the suffix. There is also a repetitive segment at about 1/4 of the sequence.

We tried multiple settings of the dotplot but found no other interesting patterns. However, there still may be some due to the fact that the image must be higly compressed.

I used https://dotlet.vital-it.ch/ (https://dotlet.vital-it.ch/) for generating and studying the picture because the tool provided in the task was not accessable (403 - Forbidden).