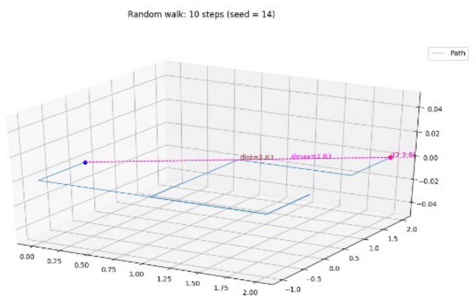
Random Walks

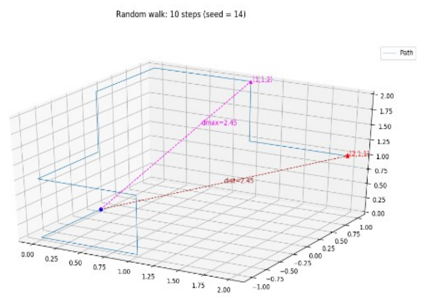
Imagine a *daemon* that strides out a certain number of paces, of equal unit length, in a constrained space matrix and with each pace being in a random direction. Can we estimate how far from its starting point, on average, our daemon will be after *N* steps? And how many steps should it be expected take to reach a certain distance?

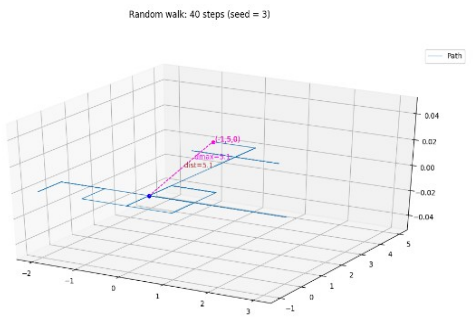
Consider three kinds of constrained movements: namely left-right travel in 1-dimensional space; east-west-north-south movement on a 2-D plane; and then adding an up-down axis of motion for the 3-D scenario. In all these travels we allow for orthogonal movement only – that is, the direction of each successive step shall be 0° (same), 180° (backtrack) or 90° (at right angles) from the previous step's direction.

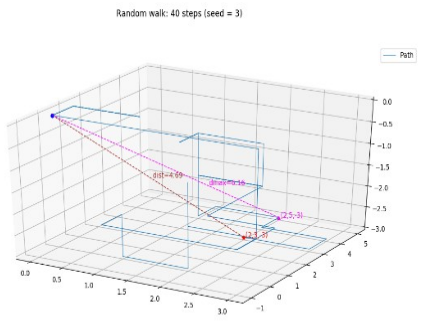
*   *   *

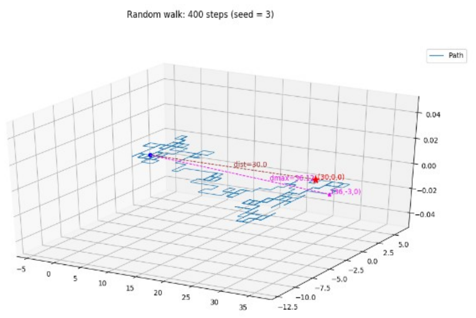
Simulating some random walks as described above, using computer generated "random" sequences, allows us to generate some sample values for various values of *N* (number of paces). Also, we can trace the paths followed in 2 or 3 dimensions graphically. Below are diagrams for 2D and 3D walks of 10, 40 and 400 steps. In each example, the blue line traces the walk; and the brown (and pink) line indicates the final (and maximum) distance achieved.   [see `randwalk4*.py`]

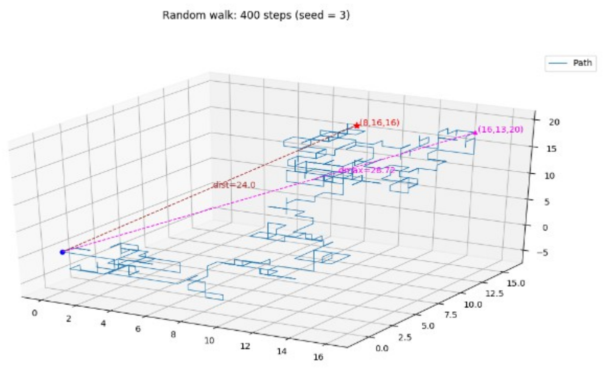












The results of running simulations for various values of *N*, using differing seed values, are shown in the tables below. For example, for a 5 step perambulation with varying seed values gives distances *d* as follows... `[see randwalk2*.py]`

Table 1

***N* (steps)**	**Seed**	***d* (1-dim)**	***d* (2-dim)**	***d* (3-dim)**
5	0	1	2.2	2.2
	1	1	2.2	1.0
	2	1	1.0	2.2
	3	1	1.0	1.0
	7	1	1.0	3.0
	9	3	3.0	1.7
	10	1	1.0	1.0
	99	3	2.2	2.2
	--	1	1.0	2.2
	Avg	**1.4**	**1.6**	**1.9**

Performing this for selected *N*-steps yields averages...

Table 2

***N* (steps)**	**$\bar{d}$ (1-dim)**	**$\bar{d}$ (2-dim)**	**$\bar{d}$ (3-dim)**
5	1.4	1.6	1.9
6	1.8	2.1	2.5
9	3.0	2.8	3.0
10	2.5	3.9	2.8
100	8.7	10.1	9.0
400	20.3	16.6	16.1
1000	29.1	26.1	24.3
10000	94.9	93.1	85.3
40000	88.3	123.0	163.5
100000	187.7	216.5	301.5
500000	453.3	510.3	512.9

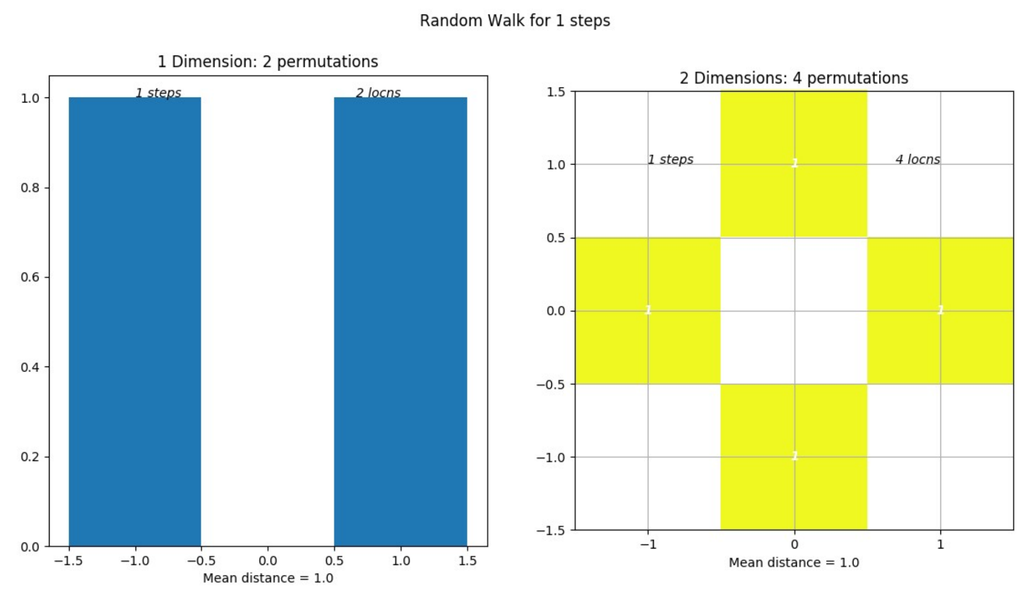
* * *

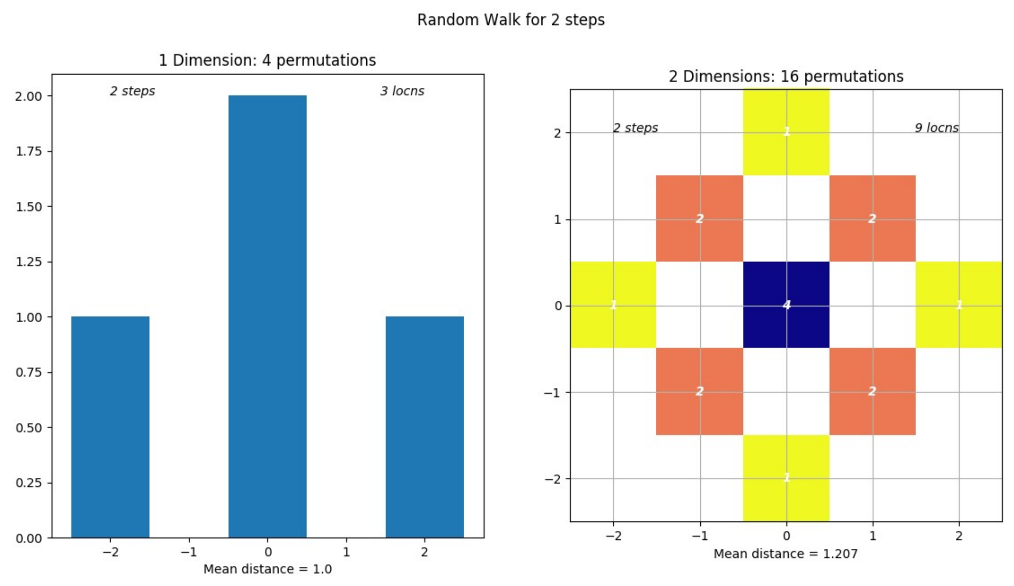
Of course, we can computationally determine an expected value $\bar{d}$ for any finite value of *N*, simply by calculating the position (and thus distance) for every permutation of *N* steps in *m* dimensions, and taking the mean average of these distances. As there are two directions of movement for each dimension, the number of permutations equals $(2m)^N$. `[see randwalk5*.py]`

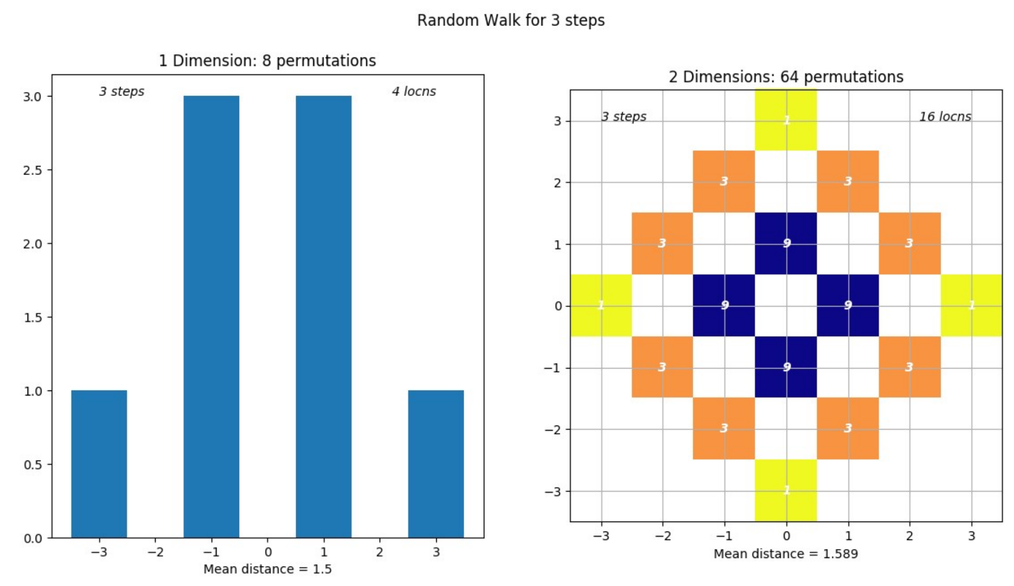
It will be appreciated that as *N* increases this number rapidly becomes very large indeed, and so in practice beyond feasible computation. Nevertheless, it will be instructive to perform the exercise for smaller *N* values; the histograms overleaf show some results, which are tabulated as follows...

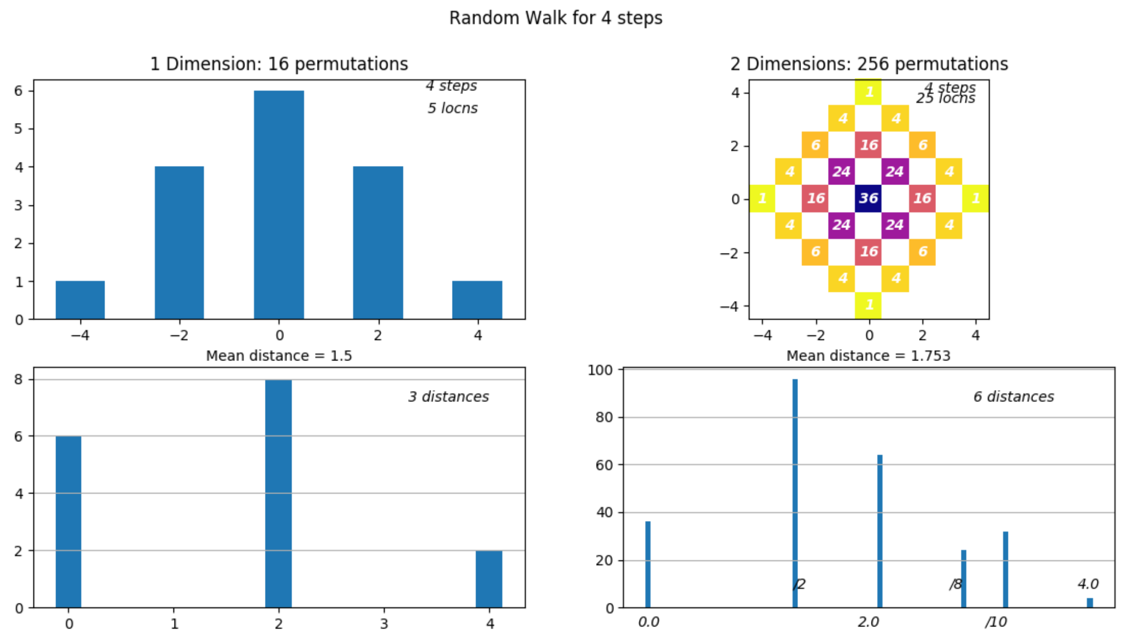
Table 3

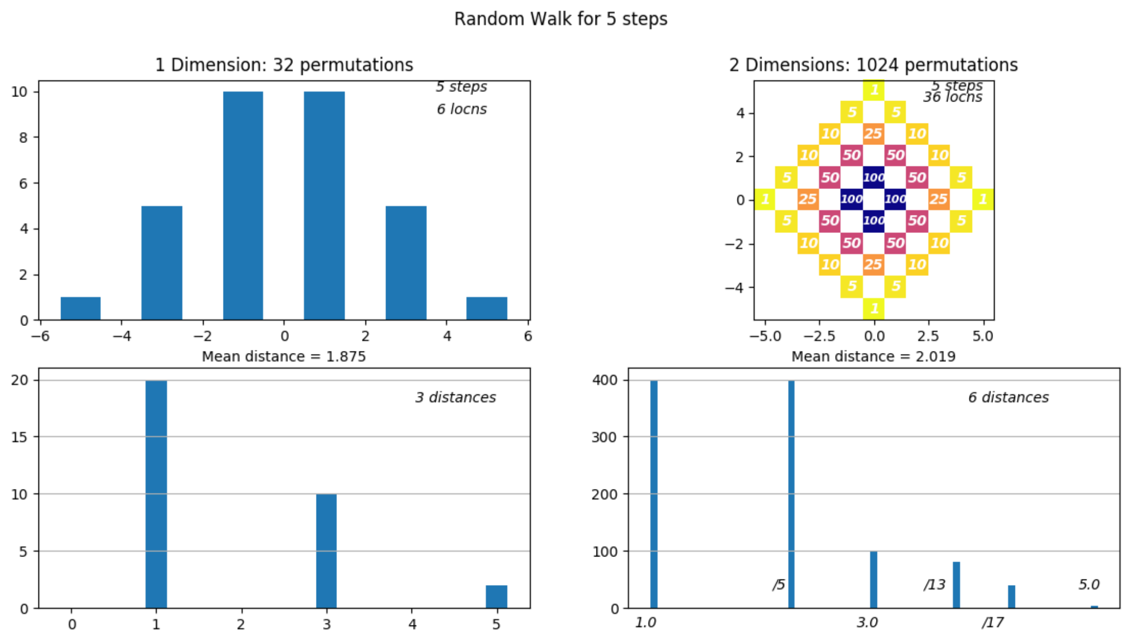
***N* (steps)**	**$\bar{d}$ (1-dim)**	**$\bar{d}$ (2-dim)**	**$\bar{d}$ (3-dim)**
1	1.00	1.00	1.00
2	1.00	1.21	1.28
3	1.50	1.59	1.63
4	1.50	1.75	1.84
5	1.88	2.02	2.08
6	1.88	2.16	2.26
7	2.19	2.37	2.46
8	2.19	2.50	2.61
9	2.46	2.68	2.78

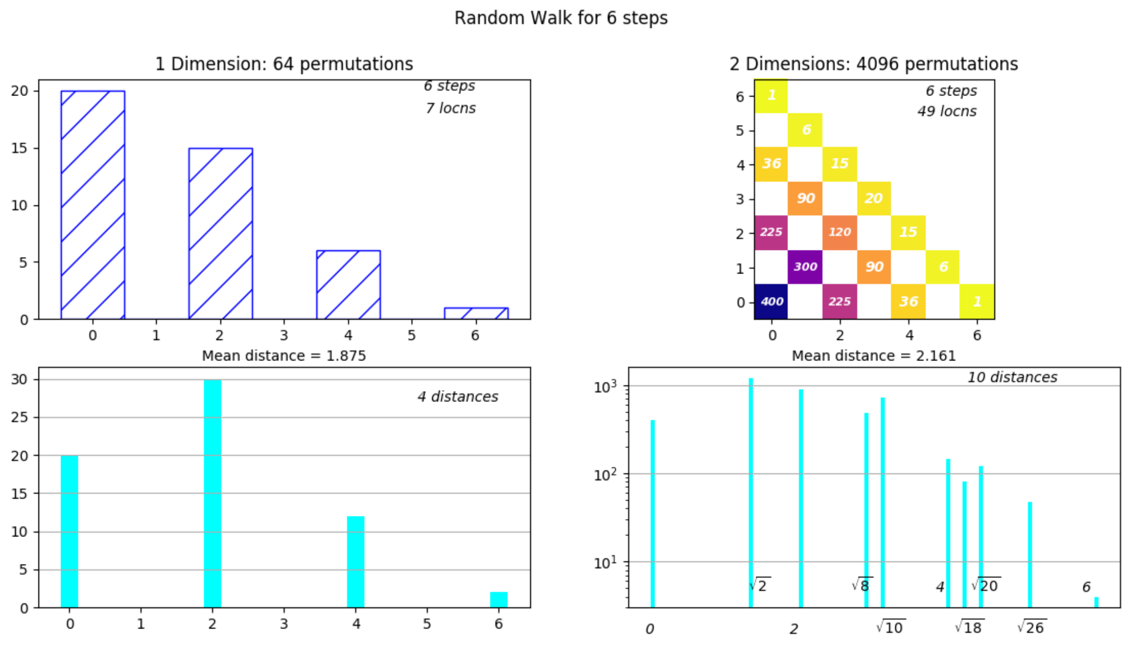

Random Walk for 1 steps
1 Dimension: 2 permutations
1 steps
2 locns
Mean distance = 1.0
2 Dimensions: 4 permutations
1 steps
4 locns
Mean distance = 1.0


Random Walk for 2 steps
1 Dimension: 4 permutations
2 steps
3 locns
Mean distance = 1.0
2 Dimensions: 16 permutations
2 steps
9 locns
Mean distance = 1.207

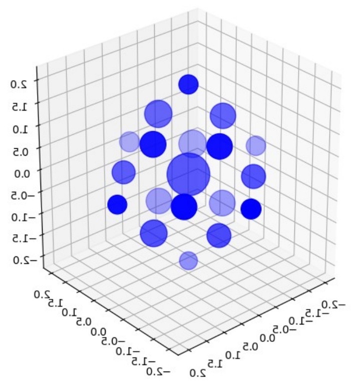

Random Walk for 3 steps
1 Dimension: 8 permutations
3 steps
4 locns
Mean distance = 1.5
2 Dimensions: 64 permutations
3 steps
16 locns
Mean distance = 1.589

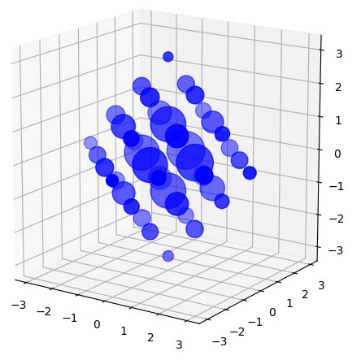

Random Walk for 4 steps
1 Dimension: 16 permutations
4 steps
5 locns
2 Dimensions: 256 permutations
4 steps
25 locns
Mean distance = 1.5
3 distances
Mean distance = 1.753
6 distances
0.0
√2
2.0
√8
√10
4.0

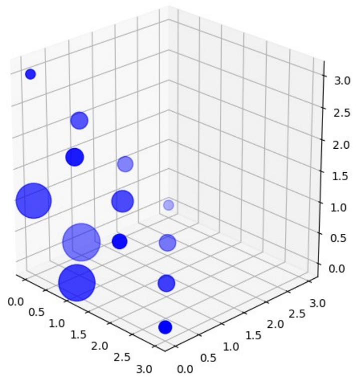

Random Walk for 5 steps
1 Dimension: 32 permutations
5 steps
6 locns
2 Dimensions: 1024 permutations
5 steps
36 locns
Mean distance = 1.875
3 distances
Mean distance = 2.019
6 distances
1.0
√5
3.0
√13
√17
5.0

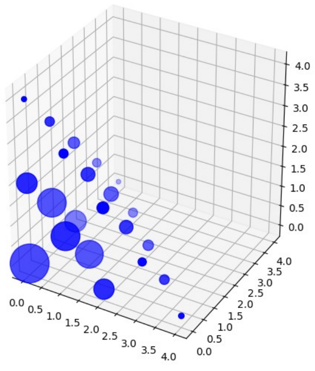


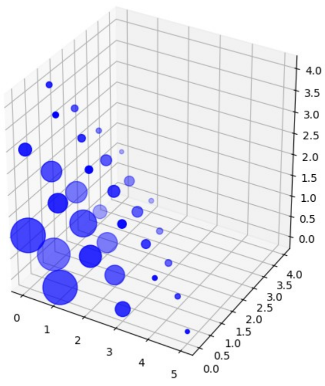
And in 3 dimensions…

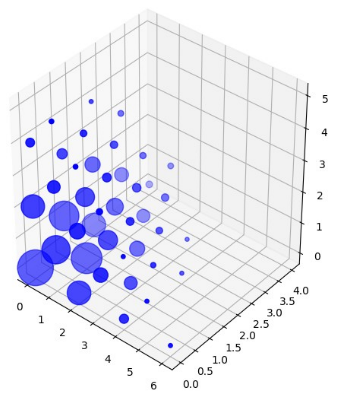

2 steps


3 steps


3 steps (1st octant)


4 steps (1st octant)


5 steps (1st octant)


6 steps (1st octant)

* * *

To extend the results in table 3 above, let us modify our algorithm to perform *N*-step walks for *N* above single figures many thousands of times, deriving empirical values of *đ* using Monte Carlo methods. `[see randwalk6*.py]`

Table 4

N (steps)	**$\bar{d}$ (1-dim)**	**$\bar{d}$ (2-dim)**	**$\bar{d}$ (3-dim)**
1	1.00	1.00	1.00
2	0.99	1.22	1.29
3	1.50	1.60	1.64
4	1.50	1.76	1.86
5	1.87	2.03	2.08
9	2.45	2.69	2.79
10	2.44	2.80	2.93
11	2.69	2.96	3.09
12	2.68	3.07	3.22
13	2.93	3.22	3.35
14	2.94	3.32	3.45
15	3.15	3.46	3.57
16	3.16	3.56	3.69
17	3.35	3.68	3.81
18	3.35	3.76	3.91
19	3.53	3.88	4.02
20	3.53	3.97	4.12
25	4.04	4.44	4.61
30	4.35	4.87	5.05
35	4.77	5.27	5.45
40	5.02	5.62	5.83
45	5.38	5.95	6.20
50	5.62	6.25	6.52
75	6.96	7.70	8.01
100	7.98	8.89	9.22
225	12.00	13.32	13.81
400	15.98	17.75	18.36
900	23.89	26.51	27.54

We see that this sampling gives results close to the exhaustive values for $N = 1 – 9$ determined previously. As we continue, let us look for some relationship between N and $\bar{d}$ being evidenced.

Perhaps, the figures suggest, $\bar{d}$ varies somehow with the square root of N, say –

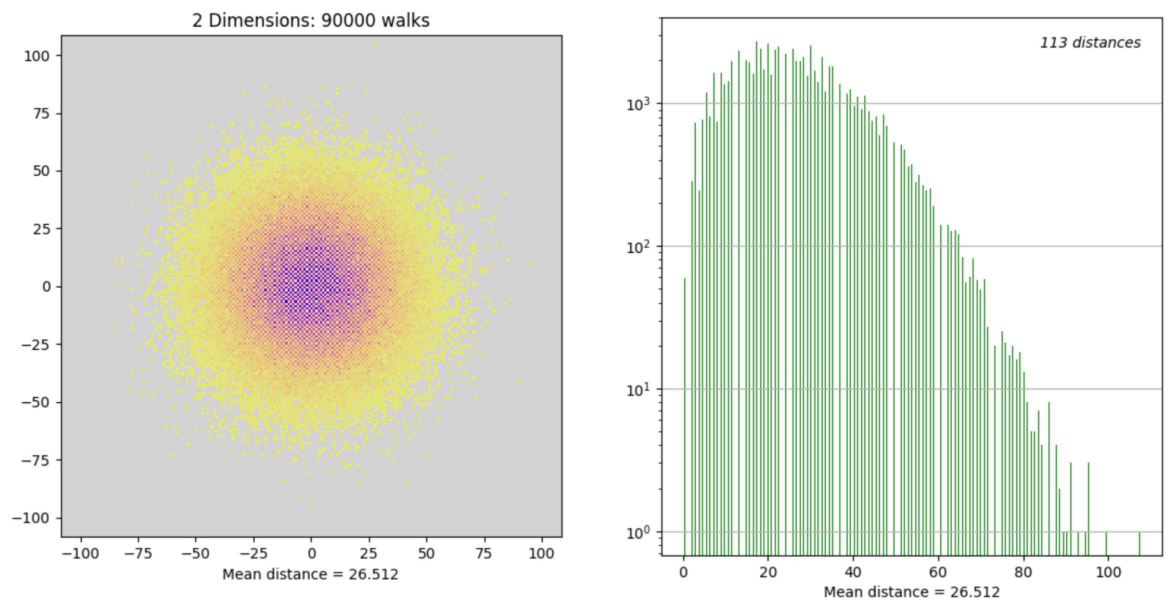
$d \rightarrow k\sqrt{N}$ as N increases, with different values of k for 1, 2 or 3 dimensional walks.

Examining this further, we see that for $D = 1$ (1 dimension), $k \approx 0.8$; for $D = 2$, $k \approx 0.889$, and $D = 3$, $k \approx 0.923$. Consequently we might surmise…

Conjecture: $d \Rightarrow \frac{4D}{4D+1} . \sqrt{N}$ as $N \Rightarrow \infty$.

* * *

90000 Random Walks for 900 steps in 2 Dimensions



We said at the outset that our daemon shall travel along an orthoganal trajectory, which is to say that it should only be permitted to continue along in a straight line, or to reverse 180°, or to turn 90°; thus moving always parallel to the axes of our grid. Let us now remove this constraint and allow for our aimless perambulater to stride forth in two dimensions, for *N* paces (of unit length), but in any direction which may at an angle from 0° to 360°.

`[see randwalk103b.py]`