Star Formation

Q & A Session 09.06.2020

Stellar Clustering

The Virial Theorem

The escape velocity from a virialised star cluster

Star clusters are groups of stars which typically contain a few hundred thousand members. Consider a star cluster of average radius $\langle R \rangle$, which consists of N identical stars of mass m.

Calculate the escape velocity of a star from this cluster.

The escape velocity of a star from a cluster is the velocity at which the star's kinetic energy exactly balances its gravitational potential energy, i.e. it is the minimal energy required for the star to become unbound from the cluster.

The total force on a star in the cluster is the sum of the two-body gravitational interactions with all the other stars in the cluster; therefore, the gravitational potential energy of a typical star is given by the sum of the two-particle potential energies for all the stars in the cluster.

Assuming that the cluster consists of n identical stars of mass m, which have a mean separation comparable to the cluster radius $\langle R \rangle$, the total energy of a star in the cluster is given by:

$$T + V = \frac{1}{2} m v^2 - (N - 1) \frac{G m^2}{\langle R \rangle} \tag{1}$$

The minimal velocity needed to escape the cluster's gravitational pull is obtained by setting the total energy to zero i.e.

$$T + V = \frac{1}{2} m v_{esc}^2 - (N - 1) \frac{G m^2}{\langle R \rangle} = 0 \tag{2}$$

The escape velocity of a star in the cluster is therefore given by

$$v_{esc} = 2 (N - 1) \frac{G m}{\langle R \rangle} \tag{3}$$

Escape velocity

Assuming that the cluster is in Virial equilibrium, relate the escape velocity to the characteristic velocity of stars inside the cluster which is given by $\overline{v} = \sqrt{\langle v^2 \rangle_t}$, where $\langle . \rangle_t$ denotes a time

average.

When a star cluster is in Virial equilibrium, the inward gravitational force is balanced by the motions of the stars, which implies a relation between the time averaged gravitational potential energy $\overline{V}$ and the time averaged kinetic energy $\overline{T}$ of the system. This relation is the Virial Theorem, which is given by

$$2\,\overline{T} + \overline{V} = 0 \tag{4}$$

We assume that the star cluster consists of N identical stars of mass m. The mean separation between two stars will be of the order of the mean cluster radius ⟨R⟩. The total gravitational potential energy of the cluster is the sum of all the two body gravitational potential energies. Noting that there are $N(N-1)/2$ possible pair interactions, gives the total gravitational potential energy of the cluster as

$$\overline{V} = \frac{N\,(N-1)}{2}\,\frac{G\,m^2}{\langle R\rangle} \tag{5}$$

The total kinetic energy of the star cluster is the sum of the single star kinetic energies; denoting the time averaged stellar velocity squared as $\langle v^2\rangle_t$ gives a total kinetic energy of

$$\overline{T} = \frac{1}{2}\,N\,m\,\langle v^2\rangle_t \tag{6}$$

Inserting these two expressions into Eq. 4 allows us to write the time averaged squared stellar velocity as

$$\langle v^2\rangle_t = \frac{(N-1)}{2}\,\frac{G\,m}{\langle R\rangle} \tag{7}$$

Using Eq. 7 we can write the escape velocity of a star, derived in Eq. 3, as

$$v_{esc} = 2\,\sqrt{\langle v^2\rangle_t} \tag{8}$$

Due to two-body encounters stars will regularly be accelerated to speeds comparable to the escape velocity. This means that star clusters gradually evaporate on timescales of $t_{evap} \simeq 10^{10}$ years.

The Coma cluster

A galaxy cluster is a structure which consists of hundreds to thousands of galaxies bound together by gravity.In 1933, Fritz Zwicky studied clusters of galaxies and especially the Coma cluster. In his paper (Die Rotverschiebung von extragalaktischen Nebeln, Zwicky, F., Helvetica PhysicaActa, Vol. 6, p. 110-127) he calculated the mass of the Coma cluster in two different ways and compared the two estimates.The aim of this exercise is to repeat Zwicky's calculations using the values given in his original paper.

1. Method

The first method to estimate the mass of the Coma cluster is to set it equal to the total visible matter. Assuming that the Coma cluster contains $N_{Gal} \simeq 800$ galaxies of average stellar mass of $M_{star} \simeq 10^9\,M_\odot$, make an estimate for the mass of the cluster.

We can get a rough estimate for the mass of the Coma cluster by adding up all the matter we see. At the time Zwicky published his paper, it was believed that the cluster consisted of $N_{Gal} \simeq 800$ galaxies

of average mass of $M_{\text{Gal}} \simeq 10^9\, M_\odot$. This allows us to estimate the total visible mass of the Coma cluster as

$$M_{\text{vis}} = N_{\text{Gal}}\, M_{\text{Gal}} \simeq 8 \times 10^{11}\, M_\odot \tag{9}$$

2. Method

A second method to estimate the mass of the Coma cluster is to infer its mass through the gravitational force it exerts. Assuming that the Coma cluster is in Virial equilibrium, has an average radius of $\langle R \rangle \simeq 0.3$ Mpc and a radial velocity dispersion $\sqrt{\langle \sigma_r^2 \rangle} \simeq 1000\ \text{km}\, s^{-1}$ estimate the mass of the cluster.

Hint: Comment on why it is sufficient to know the velocity dispersion of Coma to estimate its dynamical mass. You can then determine the total velocity dispersion from the radial velocity dispersion by assuming that the galaxy motions are isotropic

Instead of adding up all the mass we see, we can also estimate the cluster mass by considering its gravitational effect. Assuming that the Coma cluster is in Virial equilibrium, we can use Eq. 7 to obtain a rough estimate of the mass which gives rise to the observed galaxy velocities, called the dynamical mass M_{dyn}

$$M_{\text{dyn}} \simeq (N-1)\, m = \frac{2 \langle R \rangle \langle v^2 \rangle_t}{G} \tag{10}$$

We can set the time averaged squared galaxy velocity $\langle v^2 \rangle_t$ equal to the velocity dispersion $\langle \sigma^2 \rangle_t$ because the mean galaxy velocity in the cluster is the velocity of the cluster relative to the observer and it does not support the system from gravitational collapse. We further assume that the instantaneous velocity dispersion is similar to its time averaged value.

Experimentally it is only possible to obtain measurements of the radial velocity dispersion, because measuring tangential velocities is too time-consuming. Assuming that the motion of galaxies in Coma is isotropic means that there is nothing special about the radial direction and so the velocity dispersion should be equal for the other two directions, i.e.

$$\langle \sigma^2 \rangle_t = \langle \sigma_r^2 \rangle_t + \langle \sigma_\theta^2 \rangle_t + \langle \sigma_\phi^2 \rangle_t = 3 \langle \sigma_r^2 \rangle_t \tag{11}$$

Inserting the numerical values given in the problem sheet gives

$$M_{\text{dyn}} \simeq 4.4 \times 10^{14}\, M_\odot \tag{12}$$

What do you conclude by comparing these two mass estimates?

We see that the two results differ by almost 3 orders of magnitude. It is highly improbable that either of our mass estimates is wrong by such a large amount. We are therefore led to the assumption that most of the mass which makes up the Coma cluster is not visible but only makes itself felt through gravitational interaction.

Problem: The two-point correlation function

Introduction

It is well known that the distribution of galaxies on the sky is far from uniform. Qualitatively, the non-uniformity has been described in terms such as clusters, clusters of clusters, filaments, voids,

etc. Large-scale surveys, where the positional information (coordinates on the sky) have been complemented with depth information from redshifts, have mapped the 3D structure of this distribution in exquisite detail. In this assignment we shall however only deal with the projected 2D (or surface) distribution of galaxies on the sky.

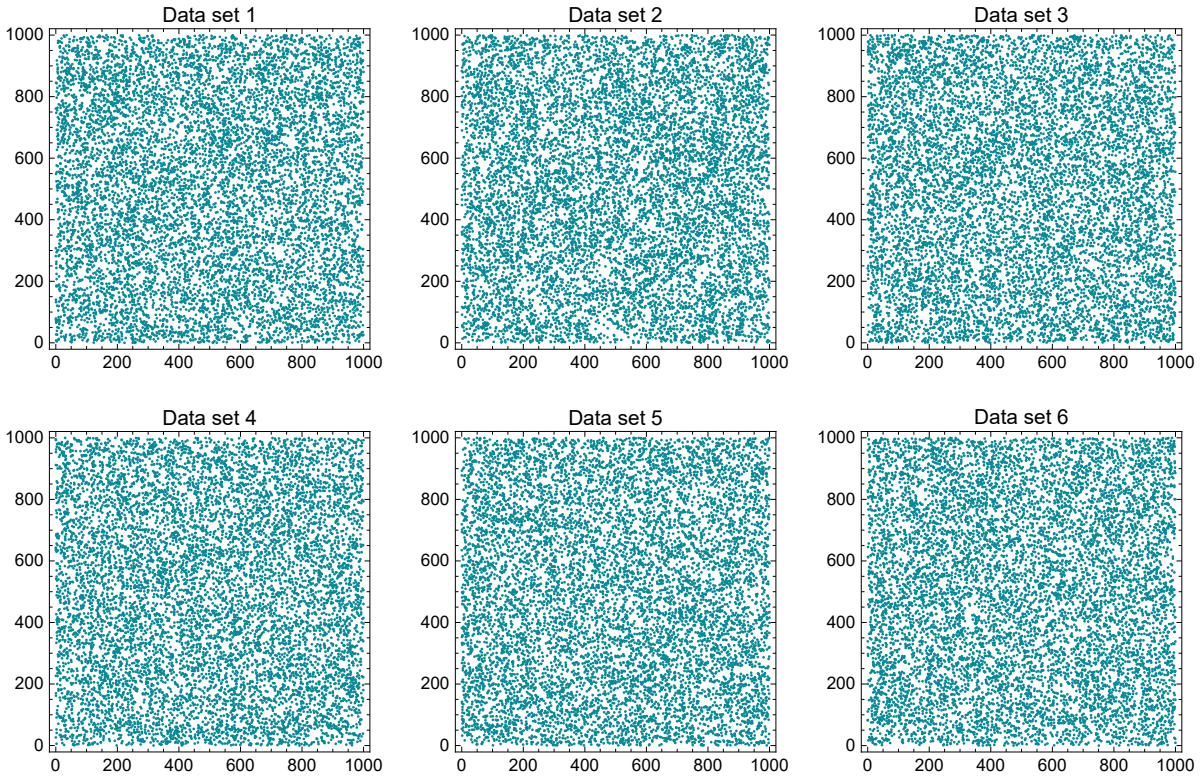
Whether we want to investigate the 3D or 2D distribution of galaxies (or of any other kind of object), a fundamental problem is how to quantify the non-uniformity. Whereas a completely uniform (random) distribution is characterized by a single number (the mean density of objects, i.e., per unit surface or unit volume), there are an infinite number of possible ways in which a non-uniform distribution could be quantified (modelled, or parameterized). The two-point angular correlation function $w(\theta)$ is one, very useful, way to do this. It

Problem

In this assignment, six different datasets will be used. Each dataset is a list of 9404 positions (x and y coordinates) of points in a square field. The coordinates have been scaled such that 0 < x < 1000 and 0 < y < 1000. The datasets can be retrieved from the web page as text files `P1data01.txt`, `P1data02.txt`, etc.

One of the datasets contains the measured positions of galaxies in the Hubble Ultra Deep Field (HUDF) [1], as observed in the i (F775W) band, and therefore exhibits a certain degree of clustering on some scales. The other five datasets were randomly generated with a uniform distribution, and therefore by definition has no clustering.

```
GraphicsGrid[Partition[Table[tmp = Import[
      "C:\\Users\\roell\\OneDrive\\Dokumente_Uni\\Projekte\\Teaching\\Vorlesung\\Star
         Formation Ffm\\SS 20\\Q & A
         Sessions\\09.06.2020\\P1data0" <> i <> ".txt", "Table"];
    ListPlot[tmp, Frame → True, FrameLabel → None, AspectRatio → 1,
     PlotLabel → "Data set " <> i], {i, ToString /@ Range[6]}], 3], ImageSize → Full]
```



The task is to estimate the two-point correlation function w(θ) for each dataset and hence decide which is the galaxy dataset. The angle θ (expressed in the same unit as the x and y coordinates) ranges from 0 to ≃1400 units.

It is recommended to estimate w(θ) only for $\theta \leq \theta_{max} \simeq 1000$, since for larger separations there are too few pairs to get reliable statistics.

To different estimators should be used and compared: the `natural' estimator(w_1) and the Landy & Szalay estimator (w_3). Once the galaxy field has been identified, describe how it deviates from a uniform distribution on different scales.

Theory

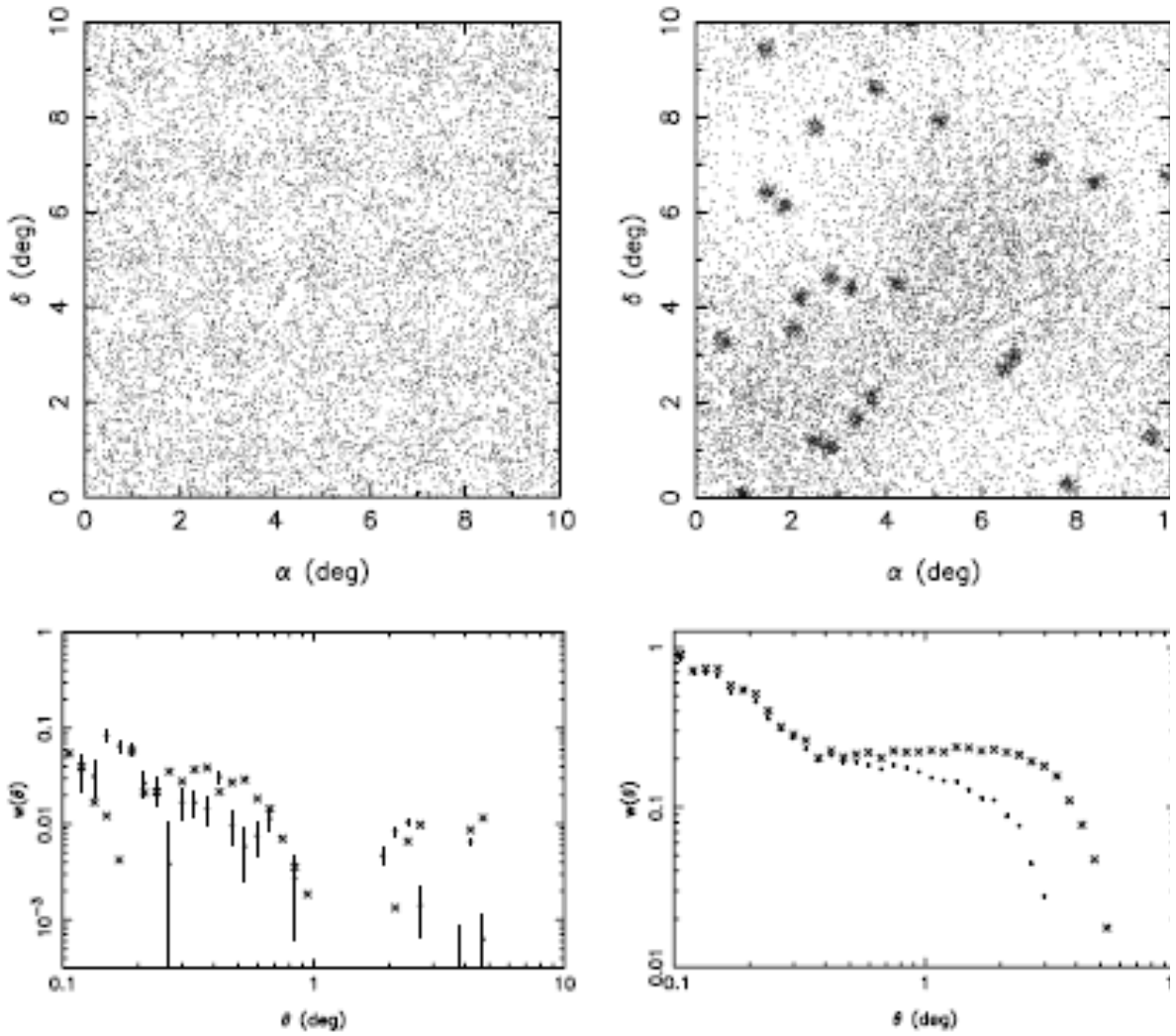
Details (including the derivation of w_3) can be found in the paper by Landy & Szalay [2]. For convenience, the formulae for w_1 and w_3 are given hereafter.

Clustering increases the number of close pairs; w(θ) quantifies this increase as a function of galaxy separation θ.

It is the fractional increase relative to a random distribution in the probability δP of finding objects in each of two solid angle elements $\delta\Omega_1$ and $\delta\Omega_2$ separated by angle θ:

$\delta P = \zeta^2 [1 + w(\theta)]\, \delta\Omega_1\, \delta\Omega_2$

where ζ is the object surface density.



The Figure shows two generated sky distributions, one (upper left) simulating low-contrast galaxy clusters in a regular grid on a random background, the other (upper right) simulating galaxy clusters on a background with large-scale structure in the form of a quadrupole.

The first 'sky' consists of a uniform random background of 8500 points, with a further 1500 points in 25 equal clusters of Gaussian width 0.4° placed on a uniform 2° by 2° grid across the area. The second has a background of 10 000 points generated from a power-spectrum representation of the sky with signal in one term only: $l = 2$. Another 2000 points were added in 25 equal 'clusters' of Gaussian width 0.1° at random positions.

Although the eye struggles to discern any features in the first sky, the two-point correlation function shows a strong signal at small θ describing the clusters themselves, and a resurgent signal at larger separations due to the 2° by 2° grid on which the clusters were placed.

Both the quadrupole and the clusters are very evident in the second sky, and the correspondingly stronger $w(\theta)$ again shows two components; the signal at the smallest separations describes the

galaxy clusters while the signal on degree scales is due to the dipole.

The dataset under investigation (i.e., one of P1data*.txt) is denoted D and contains $n = 9404$ points. The two-point correlation function is obtained by comparing the distribution of angular separations in D with the corresponding distribution of separations in a random set, called R, and which contains r points. In general r and n may be different, but it is often convenient to choose $r = n$.

The angular separation of two points i, j is $\theta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. The total number of pairs in D and R is $n(n-1)/2$ and $r(r-1)/2$, respectively. To characterize the clustering at the angular scale θ, count the number of pairs in D and R that have separations in the interval $\theta \pm \delta\theta/2$. Let DD and RR be the number of such pairs in D and R, respectively. The `natural' estimator of $w(\theta)$ is

$$w_1 = \frac{\frac{\mathrm{DD}}{n(n-1)/2} - \frac{\mathrm{RR}}{r(r-1)/2}}{\frac{\mathrm{RR}}{r(r-1)/2}} = \frac{r(r-1)}{n(n-1)}\frac{\mathrm{DD}}{\mathrm{RR}} - 1$$

To reduce the effect of statistical fluctuations in RR it is advisable to generate several (say, 10) random fields, and use the average number $\langle \mathrm{RR} \rangle$ in the above formula.

As shown by Landy & Szalay [2], w_1 is biased (i.e., it deviates systematically from 0 even when there is no clustering). A much better estimate is obtained by using also the cross-correlation statistic DR, that is the number of pairs in the separation interval $\theta \pm \delta\theta/2$, with one point taken from D and the other from R. There are $n \times r$ such pairs. The Landy-Szalay estimator is then computed as

$$w_3 = \frac{\frac{\mathrm{DD}}{n(n-1)/2} - \frac{2\,\mathrm{DR}}{n\,r} + \frac{\mathrm{RR}}{r(r-1)/2}}{\frac{\mathrm{RR}}{r(r-1)/2}} = \frac{r(r-1)}{n(n-1)}\frac{\mathrm{DD}}{\mathrm{RR}} - \frac{(r-1)}{n}\frac{\mathrm{DR}}{\mathrm{RR}} + 1$$

It is advisable to replace RR and DR by the mean counts from several realizations of R.

In the actual calculations, DD, RR, and DR should be arrays of length m ($\simeq 100$), counting the number of pairs in m bins of equal width. For example, DD(1) is the number of pairs with $0 < \theta \le \delta\theta$, DD(2) the number of pairs with $\delta\theta < \theta \le 2\,\delta\theta$, and so on, up to DD($n$), which is the number of pairs with $(m-1)\,\delta\theta < \theta \le m\,\delta\theta = \theta_{\max}$. Here, $\delta\theta = \theta_{\max}/m$ is the bin width.

As this problem is computationally rather intensive, you should be careful so that you do not repeat calculations unnecessarily. This is often a question of doing nested loops in the right order. For example, rather than looping through the bins and checking which pairs fall in that bin, it is better to loop through all the pairs only once, calculate the index of the bin to which the pair belongs, and add 1 to the count in that bin.

References

- Beckwith S.V.W., Stiavelli M., Koekemoer A.M., et al., 2006: The Hubble Ultra Deep Field, AJ 132, 1729
- Landy S.D., Szalay A.S., 1993: Bias and variance of angular correlation functions, ApJ 412, 64
- Wall J.V., Jenkins C.R., 2012: Practical Statistics for Astronomers (2nd ed.), Cambridge University Press

Functions

```
In[•]:= Length /@ {p1, p2, p3, p4, p5, p6}
Out[•]= {0, 0, 0, 0, 0, 0}

In[•]:= len = Length[p1];

In[•]:=
Clear[randomField];
randomField[len_ : len, ranges_ : {{0, 1000}, {0, 1000}}] := Module[{n, x, y},
  x = RandomReal[ranges[[1]], len];
  y = RandomReal[ranges[[2]], len];
  Transpose[{RandomSample[x], RandomSample[y]}]]

In[•]:= findDivisions[{x1_, x2_}, n_] := FindDivisions[Evaluate[-Log[10, #] & /@ {x1, x2}], n]

In[•]:= findDivisions[{0.001, 1000}, 20]
Out[•]= {-3, -5/2, -2, -3/2, -1, -1/2, 0, 1/2, 1, 3/2, 2, 5/2, 3}

In[•]:= findDivisions[{10^-3, 1000}, 30]
Out[•]= {-3, -14/5, -13/5, -12/5, -11/5, -2, -9/5, -8/5, -7/5, -6/5, -1, -4/5,
  -3/5, -2/5, -1/5, 0, 1/5, 2/5, 3/5, 4/5, 1, 6/5, 7/5, 8/5, 9/5, 2, 11/5, 12/5, 13/5, 14/5, 3}

In[•]:= getBins[bins_] := Partition[FindDivisions[{0, 1000}, bins], 2, 1]

In[•]:= Mean /@ Partition[FindDivisions[{0, 1000}, 100], 2, 1]
Out[•]= {5, 15, 25, 35, 45, 55, 65, 75, 85, 95, 105, 115, 125, 135, 145, 155, 165, 175, 185,
 195, 205, 215, 225, 235, 245, 255, 265, 275, 285, 295, 305, 315, 325, 335, 345, 355,
 365, 375, 385, 395, 405, 415, 425, 435, 445, 455, 465, 475, 485, 495, 505, 515,
 525, 535, 545, 555, 565, 575, 585, 595, 605, 615, 625, 635, 645, 655, 665, 675,
 685, 695, 705, 715, 725, 735, 745, 755, 765, 775, 785, 795, 805, 815, 825, 835,
 845, 855, 865, 875, 885, 895, 905, 915, 925, 935, 945, 955, 965, 975, 985, 995}
```

In[•]:=
```
Clear[computeDistanceBins];
Options[computeDistanceBins] =
  {"BinNumber" → 30, "MinDistance" → 0, "MaxDistance" → 1000, "Scaling" → "Linear"};
computeDistanceBins[data_List, opts : OptionsPattern[]] :=
 computeDistanceBins[data, opts] = Module[{dmD, dmR, upperR, upperD,
    binsD, binsR, n, r, maxDistance, minDistance, binnum, binArgs},
   binnum = OptionValue["BinNumber"];
   maxDistance = OptionValue["MaxDistance"];
   minDistance = OptionValue["MinDistance"];
   If[OptionValue["Scaling"] == "Linear",
    binArgs = {minDistance, maxDistance, (maxDistance - minDistance)/binnum},
    binArgs = {findDivisions[{Max[10^-3, minDistance], maxDistance}, binnum]}];
   n = Length[data];
   dmD = DistanceMatrix[data, data];
   upperD = Table[dmD[[i, i + 1 ;; n]], {i, 1, n - 1}] // Flatten;
   binsD = BinCounts[upperD, binArgs]]
computeDistanceBins[data1_List, data2_List, opts : OptionsPattern[]] :=
 computeDistanceBins[data1, data2, opts] =
  Module[{dmDR, upperDR, binsDR, n, r, maxDistance, minDistance, binnum, binArgs},
   binnum = OptionValue["BinNumber"];
   maxDistance = OptionValue["MaxDistance"];
   minDistance = OptionValue["MinDistance"];
   If[OptionValue["Scaling"] == "Linear",
    binArgs = {minDistance, maxDistance, (maxDistance - minDistance)/binnum},
    binArgs = {findDivisions[{Max[10^-3, minDistance], maxDistance}, binnum]}];
   n = Length[data1];
   r = Length[data2];
   dmDR = DistanceMatrix[data1, data2];
   upperDR = dmDR // Flatten;
   binsDR = BinCounts[upperDR, binArgs]]
```

In[•]:=
```
FindDivisions[{0, 1000, (1000 - 0)/30}, 30]
```

Out[•]=
$$\left\{0, \frac{100}{3}, \frac{200}{3}, 100, \frac{400}{3}, \frac{500}{3}, 200, \frac{700}{3}, \frac{800}{3}, 300, \frac{1000}{3}, \frac{1100}{3}, 400, \frac{1300}{3}, \frac{1400}{3}, 500, \frac{1600}{3}, \frac{1700}{3}, 600, \frac{1900}{3}, \frac{2000}{3}, 700, \frac{2200}{3}, \frac{2300}{3}, 800, \frac{2500}{3}, \frac{2600}{3}, 900, \frac{2800}{3}, \frac{2900}{3}, 1000\right\}$$

In[•]:=
```
Clear[w1]
Options[w1] = {"ShowBins" → False, "RandomFieldNumber" → 10,
    "BinNumber" → 30, "XRange" → {0, 1000}, "YRange" → {0, 1000}, "MinDistance" → 0,
    "MaxDistance" → 1000, "ComparisonField" → {}, "Scaling" → "Linear"};
w1[data_, opts : OptionsPattern[]] := Module[{randomSets, binsD, binnum,
    binsR, n = Length[data], r, maxDistance = 1000, minDistance = 0, binArgs},
  binnum = OptionValue["BinNumber"];
  If[OptionValue["Scaling"] == "Linear",
   binArgs = Mean /@ Partition[FindDivisions[
        {minDistance, maxDistance, (maxDistance - minDistance)/binnum}, binnum], 2, 1],
   binArgs = Mean /@ Partition[findDivisions[{Max[minDistance, 10^-3], maxDistance},
       binnum], 2, 1]];
  binsD = computeDistanceBins[data, Sequence @@ FilterRules[
      {opts}, Options[computeDistanceBins]]];
  If[OptionValue["ComparisonField"] == {},
   r = n;
   randomSets = Table[randomField[n, {OptionValue["XRange"], OptionValue["YRange"]}],
     {OptionValue["RandomFieldNumber"]}];
   binsR = Mean /@ Transpose[(computeDistanceBins[#, Sequence @@
          FilterRules[{opts}, Options[computeDistanceBins]]] & /@ randomSets)],
   r = Length[OptionValue["ComparisonField"]];
   binsR = computeDistanceBins[OptionValue["ComparisonField"],
     Sequence @@ FilterRules[{opts}, Options[computeDistanceBins]]]];
  If[OptionValue["ShowBins"], {(r (r - 1) binsD)/(n (n - 1) binsR) - 1., {binsD, binsR}},
   Transpose[{binArgs, (r (r - 1) binsD)/(n (n - 1) binsR) - 1.}]]]
```

In[•]:=
```
Clear[w3]
Options[w3] = {"ShowBins" → False, "RandomFieldNumber" → 10, "BinNumber" → 30,
   "XRange" → {0, 1000}, "YRange" → {0, 1000}, "ComparisonField" → {},
   "MinDistance" → 0, "MaxDistance" → 1000, "Scaling" → "Linear"};
w3[data_, opts : OptionsPattern[]] := Module[{randomSets, binsD, binsR, binnum,
   binsDR, n = Length[data], r, maxDistance = 1000, minDistance = 0, binArgs},
  binnum = OptionValue["BinNumber"];
  If[OptionValue["Scaling"] == "Linear",
   binArgs = Mean /@ Partition[FindDivisions[
       {minDistance, maxDistance, (maxDistance - minDistance)/binnum}, binnum], 2, 1],
   binArgs = Mean /@ Partition[findDivisions[{Max[minDistance, 10^-3], maxDistance},
       binnum], 2, 1]];
  binsD = computeDistanceBins[data, Sequence @@ FilterRules[
      {opts}, Options[computeDistanceBins]]];
  If[OptionValue["ComparisonField"] == {},
   r = n;
   randomSets = Table[randomField[n, {OptionValue["XRange"], OptionValue["YRange"]}],
     {OptionValue["RandomFieldNumber"]}];
   binsR = Mean /@ Transpose[(computeDistanceBins[#, Sequence @@
         FilterRules[{opts}, Options[computeDistanceBins]]] & /@ randomSets)],
   r = Length[OptionValue["ComparisonField"]];
   binsR = computeDistanceBins[OptionValue["ComparisonField"],
     Sequence @@ FilterRules[{opts}, Options[computeDistanceBins]]]];
  binsDR = Mean /@ Transpose[(computeDistanceBins[data, #, Sequence @@
        FilterRules[{opts}, Options[computeDistanceBins]]] & /@ randomSets)];

  If[OptionValue["ShowBins"],
   {(r (r - 1))/(n (n - 1)) binsD/binsR - (r - 1)/n binsDR/binsR + 1., {binsD, binsR, binsDR}},
   Transpose[{binArgs, (r (r - 1))/(n (n - 1)) binsD/binsR - (r - 1)/n binsDR/binsR + 1.}]]
 ]
```

In[•]:=
```
Clear[wLS]
Options[wLS] = {"ShowBins" → False, "RandomFieldNumber" → 10, "BinNumber" → 30,
   "XRange" → {0, 1000}, "YRange" → {0, 1000}, "ComparisonField" → {},
   "MinDistance" → 0, "MaxDistance" → 1000, "Scaling" → "Linear"};
wLS[data_, opts : OptionsPattern[]] := Module[{randomSets, binsD, binsR, binnum,
   binsDR, n = Length[data], r, maxDistance = 1000, minDistance = 0, binArgs},
  binnum = OptionValue["BinNumber"];
  If[OptionValue["Scaling"] == "Linear",
   binArgs = Mean /@ Partition[FindDivisions[
       {minDistance, maxDistance, (maxDistance - minDistance)/binnum}, binnum], 2, 1],
   binArgs = Mean /@ Partition[findDivisions[{Max[minDistance, 10^-3], maxDistance},
       binnum], 2, 1]];
  binsD = computeDistanceBins[data, Sequence @@ FilterRules[{opts},
       Options[computeDistanceBins]]] / (n (n - 1) / 2);
  If[OptionValue["ComparisonField"] == {},
   r = n;
   randomSets = Table[randomField[n, {OptionValue["XRange"], OptionValue["YRange"]}],
     {OptionValue["RandomFieldNumber"]}];
   binsR = (Mean /@ Transpose[(computeDistanceBins[#, Sequence @@ FilterRules[{opts},
              Options[computeDistanceBins]]] & /@ randomSets)]) / (r (r - 1) / 2),
   r = Length[OptionValue["ComparisonField"]];
   binsR = computeDistanceBins[OptionValue["ComparisonField"],
      Sequence @@ FilterRules[{opts}, Options[computeDistanceBins]]] / (r (r - 1) / 2)];
  binsDR = (Mean /@ Transpose[(computeDistanceBins[data, #, Sequence @@ FilterRules[
            {opts}, Options[computeDistanceBins]]] & /@ randomSets)]) / (r * n);

  If[OptionValue["ShowBins"], {(binsD - 2 binsDR + binsR)/binsR, {binsD, binsR, binsDR}},
   Transpose[{binArgs, (binsD - 2 binsDR + binsR)/binsR}]]
 ]
```

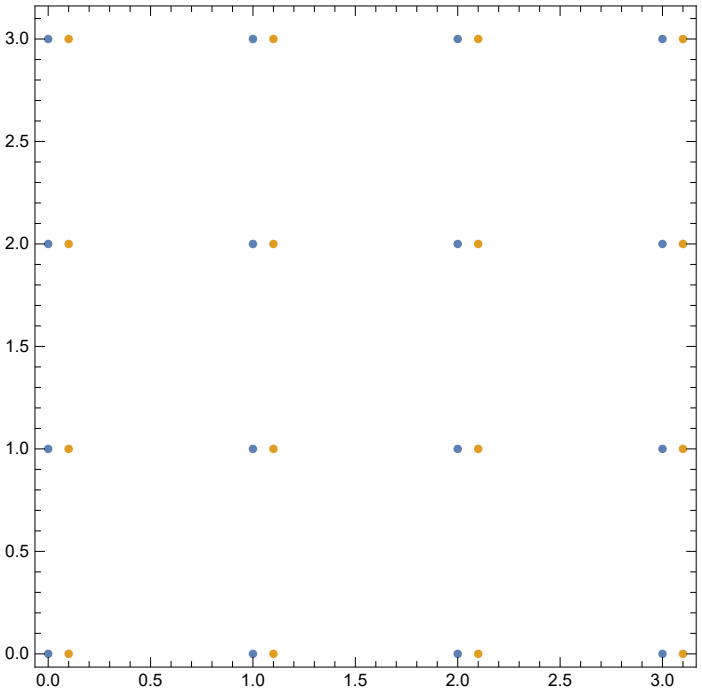
Example

In[•]:=
```
set1 = Flatten[Outer[List, {0, 1, 2, 3}, {0, 1, 2, 3}], 1]
set2 = Flatten[Outer[List, {0, 1, 2, 3} + .1, {0, 1, 2, 3}], 1]
ListPlot[{set1, set2}, AspectRatio → 1, Frame → True]
```

Out[•]=
```
{{0, 0}, {0, 1}, {0, 2}, {0, 3}, {1, 0}, {1, 1}, {1, 2},
 {1, 3}, {2, 0}, {2, 1}, {2, 2}, {2, 3}, {3, 0}, {3, 1}, {3, 2}, {3, 3}}
```

Out[•]=
```
{{0.1, 0}, {0.1, 1}, {0.1, 2}, {0.1, 3}, {1.1, 0}, {1.1, 1}, {1.1, 2}, {1.1, 3},
 {2.1, 0}, {2.1, 1}, {2.1, 2}, {2.1, 3}, {3.1, 0}, {3.1, 1}, {3.1, 2}, {3.1, 3}}
```

Out[•]=



In[•]:=
```
DistanceMatrix[set1] // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 1 & \sqrt{2} & \sqrt{5} & \sqrt{10} & 2 & \sqrt{5} & 2\sqrt{2} & \sqrt{13} & 3 & \sqrt{10} & \sqrt{1} \\
1 & 0 & 1 & 2 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & \sqrt{5} & 2 & \sqrt{5} & 2\sqrt{2} & \sqrt{10} & 3 & \sqrt{1} \\
2 & 1 & 0 & 1 & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 2\sqrt{2} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{13} & \sqrt{10} & 3 \\
3 & 2 & 1 & 0 & \sqrt{10} & \sqrt{5} & \sqrt{2} & 1 & \sqrt{13} & 2\sqrt{2} & \sqrt{5} & 2 & 3\sqrt{2} & \sqrt{13} & \sqrt{1} \\
1 & \sqrt{2} & \sqrt{5} & \sqrt{10} & 0 & 1 & 2 & 3 & 1 & \sqrt{2} & \sqrt{5} & \sqrt{10} & 2 & \sqrt{5} & 2\sqrt{} \\
\sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & 1 & 0 & 1 & 2 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & \sqrt{5} & 2 & \sqrt{5} \\
\sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 2 & 1 & 0 & 1 & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 2\sqrt{2} & \sqrt{5} & 2 \\
\sqrt{10} & \sqrt{5} & \sqrt{2} & 1 & 3 & 2 & 1 & 0 & \sqrt{10} & \sqrt{5} & \sqrt{2} & 1 & \sqrt{13} & 2\sqrt{2} & \sqrt{5} \\
2 & \sqrt{5} & 2\sqrt{2} & \sqrt{13} & 1 & \sqrt{2} & \sqrt{5} & \sqrt{10} & 0 & 1 & 2 & 3 & 1 & \sqrt{2} & \sqrt{5} \\
\sqrt{5} & 2 & \sqrt{5} & 2\sqrt{2} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & 1 & 0 & 1 & 2 & \sqrt{2} & 1 & \sqrt{2} \\
2\sqrt{2} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 2 & 1 & 0 & 1 & \sqrt{5} & \sqrt{2} & 1 \\
\sqrt{13} & 2\sqrt{2} & \sqrt{5} & 2 & \sqrt{10} & \sqrt{5} & \sqrt{2} & 1 & 3 & 2 & 1 & 0 & \sqrt{10} & \sqrt{5} & \sqrt{2} \\
3 & \sqrt{10} & \sqrt{13} & 3\sqrt{2} & 2 & \sqrt{5} & 2\sqrt{2} & \sqrt{13} & 1 & \sqrt{2} & \sqrt{5} & \sqrt{10} & 0 & 1 & 2 \\
\sqrt{10} & 3 & \sqrt{10} & \sqrt{13} & \sqrt{5} & 2 & \sqrt{5} & 2\sqrt{2} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & 1 & 0 & 1 \\
\sqrt{13} & \sqrt{10} & 3 & \sqrt{10} & 2\sqrt{2} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 2 & 1 & 0 \\
3\sqrt{2} & \sqrt{13} & \sqrt{10} & 3 & \sqrt{13} & 2\sqrt{2} & \sqrt{5} & 2 & \sqrt{10} & \sqrt{5} & \sqrt{2} & 1 & 3 & 2 & 1
\end{pmatrix}$$

```
In[•]:= DistanceMatrix[set1] // N // Flatten // Tally
Out[•]= {{0., 16}, {1., 48}, {2., 32}, {3., 16}, {1.41421, 36},
  {2.23607, 48}, {3.16228, 24}, {2.82843, 16}, {3.60555, 16}, {4.24264, 4}}

In[•]:= computeDistanceBins[set1, "BinNumber" → 5, "MinDistance" → 0, "MaxDistance" → 5]
        computeDistanceBins[set2, "BinNumber" → 5, "MinDistance" → 0, "MaxDistance" → 5]
Out[•]= {0, 42, 48, 28, 2}

Out[•]= {0, 42, 48, 28, 2}
```

$$w_1 = \frac{\frac{\mathrm{DD}}{n(n-1)/2} - \frac{\mathrm{RR}}{r(r-1)/2}}{\frac{\mathrm{RR}}{r(r-1)/2}} = \frac{r(r-1)}{n(n-1)}\frac{\mathrm{DD}}{\mathrm{RR}} - 1$$

```
In[•]:= computeDistanceBins[set1, "BinNumber" → 30,
          "MinDistance" → 10^-3, "MaxDistance" → 5, "Scaling" -> "Log"]
Out[•]= {0, 0, 0, 0, 0, 0, 0, 0, 0, 24, 0, 18, 0, 0, 16, 24, 0, 0, 8}

In[•]:= findDivisions[{10^-3, 5}, 30]
```

$$\left\{-\frac{4}{5}, -\frac{3}{5}, -\frac{2}{5}, -\frac{1}{5}, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, \frac{6}{5}, \frac{7}{5}, \frac{8}{5}, \frac{9}{5}, 2, \frac{11}{5}, \frac{12}{5}, \frac{13}{5}, \frac{14}{5}, 3\right\}$$

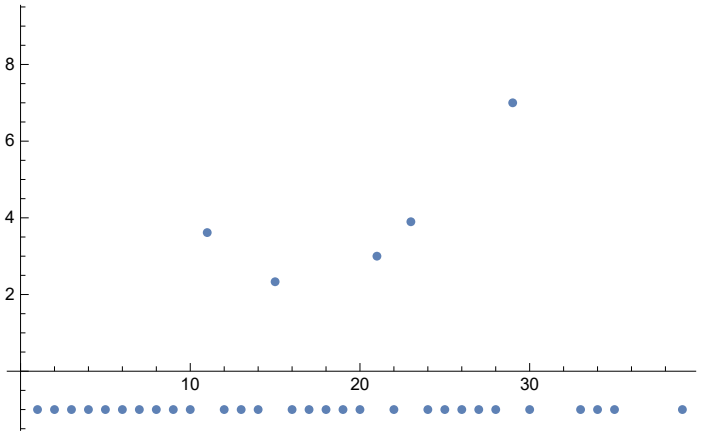
```
In[•]:= w1[set1, "XRange" → {0, 3}, "YRange" → {0, 3}, "BinWidth" → 1,
          "MinDistance" → 0, "MaxDistance" → 5, "ComparisonField" → set2] // Quiet
{25, 75, 125, 175, 225, 275, 325, 375, 425, 475, 525, 575, 625, 675, 725, 775, 825, 875, 925, 975}

Out[•]= Transpose[{{25, 75, 125, 175, 225, 275, 325,
     375, 425, 475, 525, 575, 625, 675, 725, 775, 825, 875, 925, 975},
    {Indeterminate, Indeterminate, Indeterminate, Indeterminate, Indeterminate,
     Indeterminate, 0., Indeterminate, 0., Indeterminate, Indeterminate,
     Indeterminate, 0., 0., Indeterminate, Indeterminate, 0., Indeterminate, 0.,
     Indeterminate, Indeterminate, 0., Indeterminate, Indeterminate, Indeterminate,
     0., Indeterminate, Indeterminate, Indeterminate, Indeterminate}}]
```

In[•]:=
```
SeedRandom[1]
w1[set1, "XRange" → {0, 3}, "YRange" → {0, 3},
  "BinWidth" → .1, "MinDistance" → 0, "MaxDistance" → 5] // Quiet
ListPlot[
 %]
```

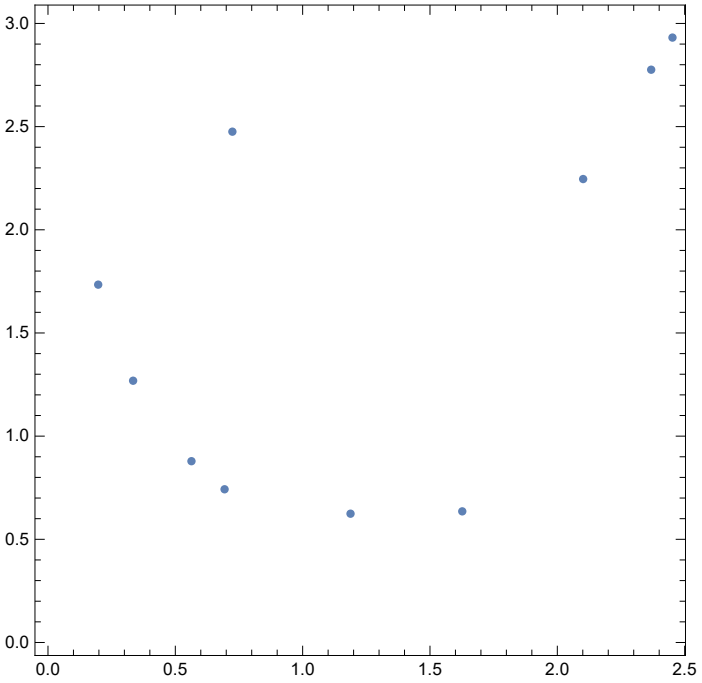
Out[•]=
```
{-1., -1., -1., -1., -1., -1., -1., -1., -1., -1., 3.61538, -1., -1., -1., 2.33333,
 -1., -1., -1., -1., -1., 3., -1., 3.89796, -1., -1., -1., -1., -1., 7., -1.,
 12.3333, 29., -1., -1., -1., Indeterminate, 79., Indeterminate, -1., Indeterminate,
 Indeterminate, Indeterminate, ComplexInfinity, Indeterminate, Indeterminate,
 Indeterminate, Indeterminate, Indeterminate, Indeterminate, Indeterminate}
```

Out[•]=

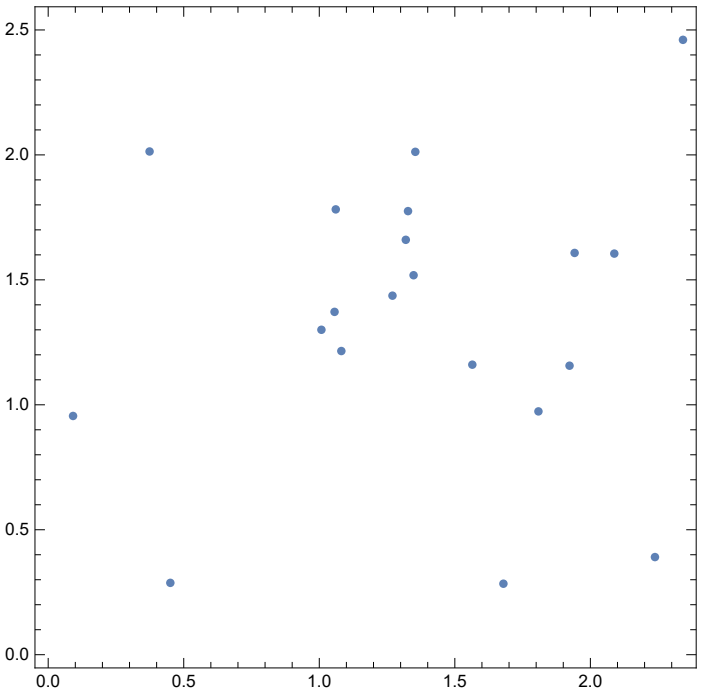


In[•]:=
```
SeedRandom[1]
rand1 = randomField[10, {{0, 3}, {0, 3}}];
rand2 = Join[randomField[10, {{0, 3}, {0, 3}}], randomField[10, {{1, 2}, {1, 2}}]];
ListPlot[rand1, AspectRatio → 1, Frame → True]
ListPlot[rand2, AspectRatio → 1, Frame → True]
```

Out[•]=



Out[•]=



In[•]:=
```
computeDistanceBins[rand1, "BinNumber" → 15, "MinDistance" → 0, "MaxDistance" → 5]
computeDistanceBins[rand2, "BinNumber" → 15, "MinDistance" → 0, "MaxDistance" → 5]
```

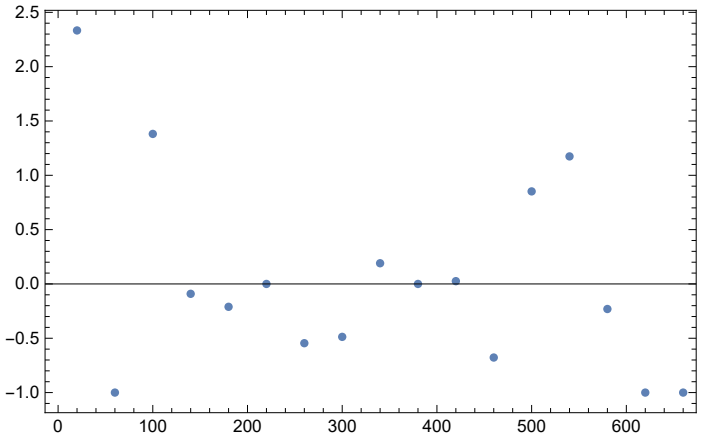
Out[•]= {2, 6, 5, 4, 4, 8, 5, 8, 3, 0, 0, 0, 0, 0, 0}

Out[•]= {18, 34, 46, 32, 31, 19, 7, 1, 2, 0, 0, 0, 0, 0, 0}

In[•]:=
```
SeedRandom[1]
w1[rand1, "XRange" → {0, 3}, "YRange" → {0, 3},
  "BinNumber" → 25, "MinDistance" → 0, "MaxDistance" → 5] // Quiet
ListPlot[%, Frame → True]
```

Out[•]=
```
{{20, 2.33333}, {60, -1.}, {100, 1.38095}, {140, -0.0909091}, {180, -0.210526},
 {220, 0.}, {260, -0.545455}, {300, -0.487179}, {340, 0.190476}, {380, 0.},
 {420, 0.025641}, {460, -0.677419}, {500, 0.851852}, {540, 1.17391}, {580, -0.230769},
 {620, -1.}, {660, -1.}, {700, Indeterminate}, {740, Indeterminate},
 {780, Indeterminate}, {820, Indeterminate}, {860, Indeterminate},
 {900, Indeterminate}, {940, Indeterminate}, {980, Indeterminate}}
```

Out[•]=

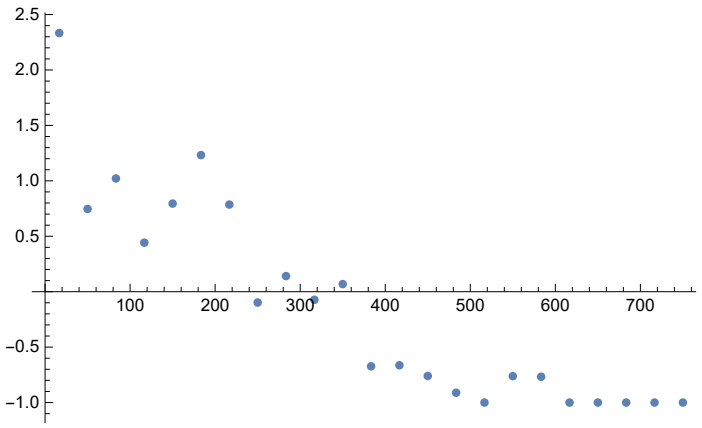


In[•]:=
```
SeedRandom[1]
w1[rand2, "XRange" → {0, 3}, "YRange" → {0, 3},
  "BinNumber" → 30, "MinDistance" → 0, "MaxDistance" → 5] // Quiet
ListPlot[%]
w3[rand2, "XRange" → {0, 3}, "YRange" → {0, 3},
   "BinNumber" → 30, "MinDistance" → 0, "MaxDistance" → 5] // Quiet;
ListPlot[
 %]
```

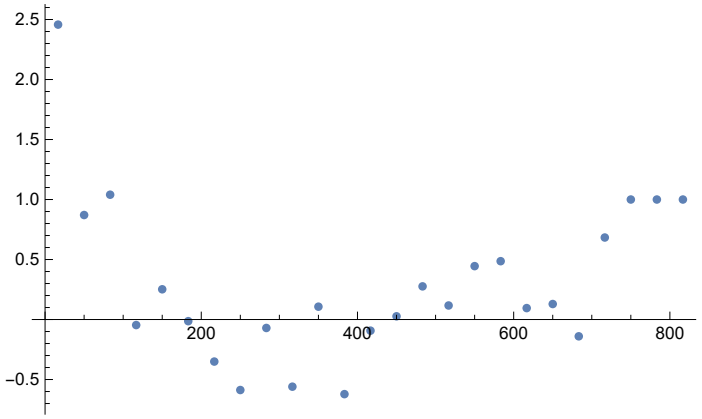
```
30
```

Out[•]=
$$\left\{\left\{\frac{50}{3}, 2.33333\right\}, \{50, 0.746032\}, \left\{\frac{250}{3}, 1.02128\right\}, \left\{\frac{350}{3}, 0.442308\right\}, \{150, 0.794872\}, \left\{\frac{550}{3}, 1.23214\right\}, \left\{\frac{650}{3}, 0.785714\right\}, \{250, -0.0977444\}, \left\{\frac{850}{3}, 0.14094\right\}, \left\{\frac{950}{3}, -0.0728477\right\}, \{350, 0.0687023\}, \left\{\frac{1150}{3}, -0.673203\right\}, \left\{\frac{1250}{3}, -0.663866\right\}, \{450, -0.76\}, \left\{\frac{1450}{3}, -0.911504\right\}, \left\{\frac{1550}{3}, -1.\right\}, \{550, -0.761905\}, \left\{\frac{1750}{3}, -0.767442\right\}, \left\{\frac{1850}{3}, -1.\right\}, \{650, -1.\}, \left\{\frac{2050}{3}, -1.\right\}, \left\{\frac{2150}{3}, -1.\right\}, \{750, -1.\}, \left\{\frac{2350}{3}, \text{Indeterminate}\right\}, \left\{\frac{2450}{3}, \text{Indeterminate}\right\}, \{850, \text{Indeterminate}\}, \left\{\frac{2650}{3}, \text{Indeterminate}\right\}, \left\{\frac{2750}{3}, \text{Indeterminate}\right\}, \{950, \text{Indeterminate}\}, \left\{\frac{2950}{3}, \text{Indeterminate}\right\}\right\}$$

Out[•]=



Out[•]=



In[•]:=
```
w3[rand2, "ShowBins" → False, "XRange" → {0, 3}, "YRange" → {0, 3},
  "BinNumber" → 30, "MinDistance" → 10^-3, "MaxDistance" → 5] // Quiet
```

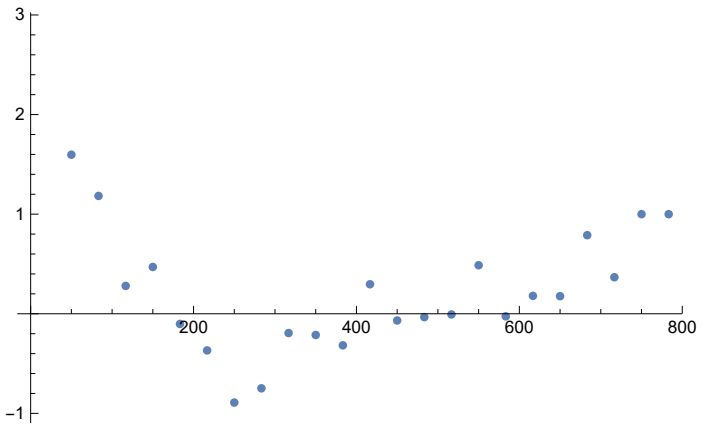
Out[•]=
$$\left\{\left\{\frac{50}{3}, 4.33846\right\}, \{50, 1.01471\}, \left\{\frac{250}{3}, 1.32078\right\}, \left\{\frac{350}{3}, 0.211712\right\}, \{150, 0.490942\}, \left\{\frac{550}{3}, 0.120161\right\}, \left\{\frac{650}{3}, -0.428409\right\}, \{250, -1.1663\}, \left\{\frac{850}{3}, -0.350904\right\}, \left\{\frac{950}{3}, -0.393077\right\}, \{350, -0.313929\}, \left\{\frac{1150}{3}, -0.391154\right\}, \left\{\frac{1250}{3}, 0.310385\right\}, \{450, 0.0254386\}, \left\{\frac{1450}{3}, -0.138554\right\}, \left\{\frac{1550}{3}, 0.02625\right\}, \{550, 0.451724\}, \left\{\frac{1750}{3}, 0.39\right\}, \left\{\frac{1850}{3}, 0.502381\right\}, \{650, 0.720588\}, \left\{\frac{2050}{3}, 0.366667\right\}, \left\{\frac{2150}{3}, 0.864286\right\}, \{750, 1.\}, \left\{\frac{2350}{3}, 1.\right\}, \left\{\frac{2450}{3}, \text{Indeterminate}\right\}, \{850, \text{Indeterminate}\}, \left\{\frac{2650}{3}, \text{Indeterminate}\right\}, \left\{\frac{2750}{3}, \text{Indeterminate}\right\}, \{950, \text{Indeterminate}\}, \left\{\frac{2950}{3}, \text{Indeterminate}\right\}\right\}$$

$$w_1 = \frac{r(r-1)}{n(n-1)}\frac{\text{DD}}{\text{RR}} - 1$$

In[•]:=
```
wLS[rand2, "XRange" → {0, 3}, "YRange" → {0, 3},
  "BinNumber" → 30, "MinDistance" → 10^-3, "MaxDistance" → 5] // Quiet
ListPlot[
 %]
```

Out[•]=
$$\left\{\left\{\frac{50}{3}, \frac{497}{120}\right\}, \left\{50, \frac{313}{196}\right\}, \left\{\frac{250}{3}, \frac{491}{415}\right\}, \left\{\frac{350}{3}, \frac{144}{515}\right\}, \left\{150, \frac{1043}{2220}\right\}, \left\{\frac{550}{3}, -\frac{101}{990}\right\}, \left\{\frac{650}{3}, -\frac{287}{780}\right\}, \left\{250, -\frac{2639}{2960}\right\}, \left\{\frac{850}{3}, -\frac{1871}{2500}\right\}, \left\{\frac{950}{3}, -\frac{689}{3560}\right\}, \left\{350, -\frac{82}{385}\right\}, \left\{\frac{1150}{3}, -\frac{843}{2660}\right\}, \left\{\frac{1250}{3}, \frac{883}{2980}\right\}, \left\{450, -\frac{38}{565}\right\}, \left\{\frac{1450}{3}, -\frac{13}{384}\right\}, \left\{\frac{1550}{3}, -\frac{11}{1680}\right\}, \left\{550, \frac{151}{310}\right\}, \left\{\frac{1750}{3}, -\frac{4}{155}\right\}, \left\{\frac{1850}{3}, \frac{79}{440}\right\}, \left\{650, \frac{53}{300}\right\}, \left\{\frac{2050}{3}, \frac{71}{90}\right\}, \left\{\frac{2150}{3}, \frac{11}{30}\right\}, \{750, 1\}, \left\{\frac{2350}{3}, 1\right\}, \left\{\frac{2450}{3}, \text{Indeterminate}\right\}, \{850, \text{Indeterminate}\}, \left\{\frac{2650}{3}, \text{Indeterminate}\right\}, \left\{\frac{2750}{3}, \text{Indeterminate}\right\}, \{950, \text{Indeterminate}\}, \left\{\frac{2950}{3}, \text{Indeterminate}\right\}\right\}$$

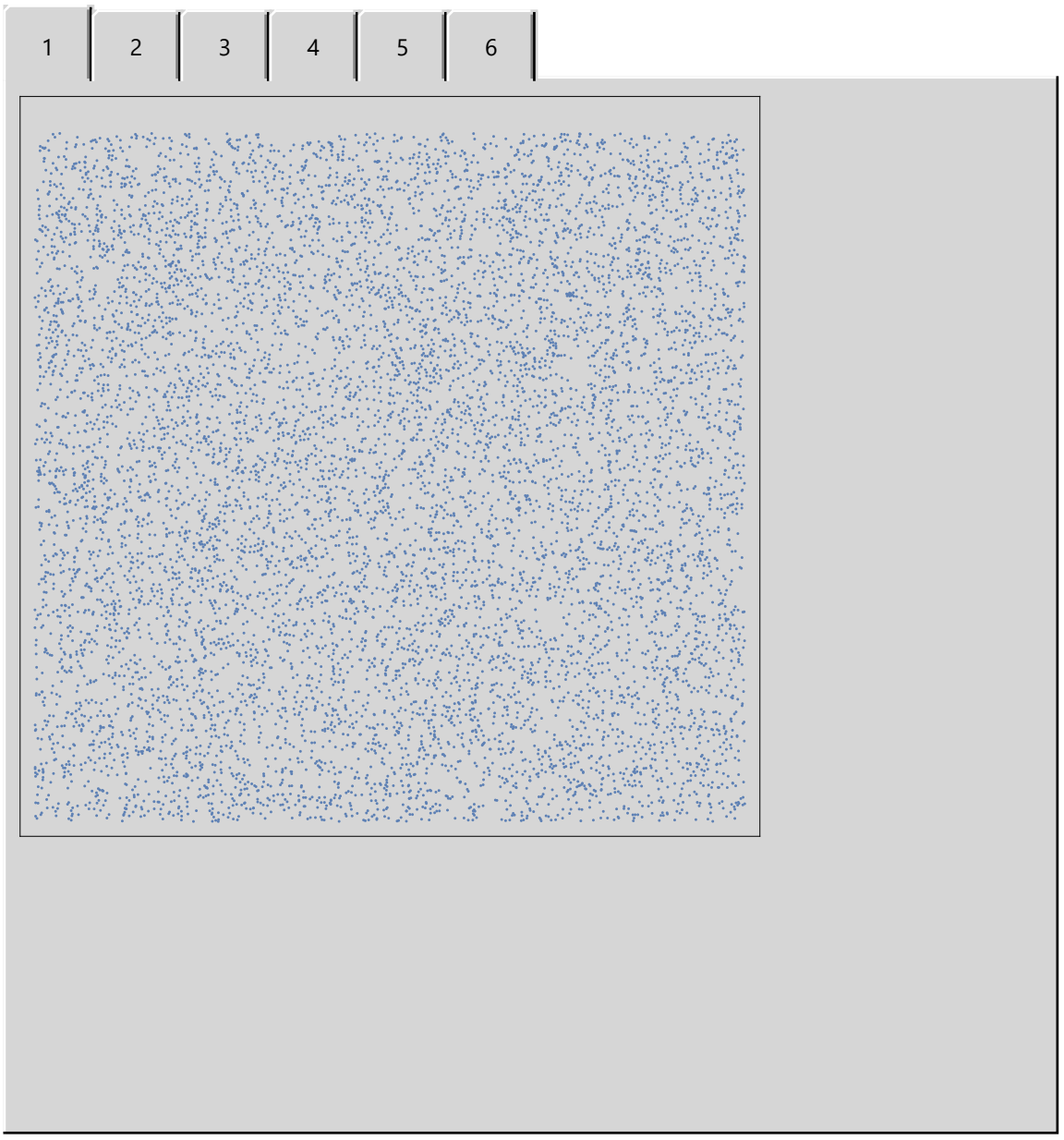
Out[•]=



Large Data Sets

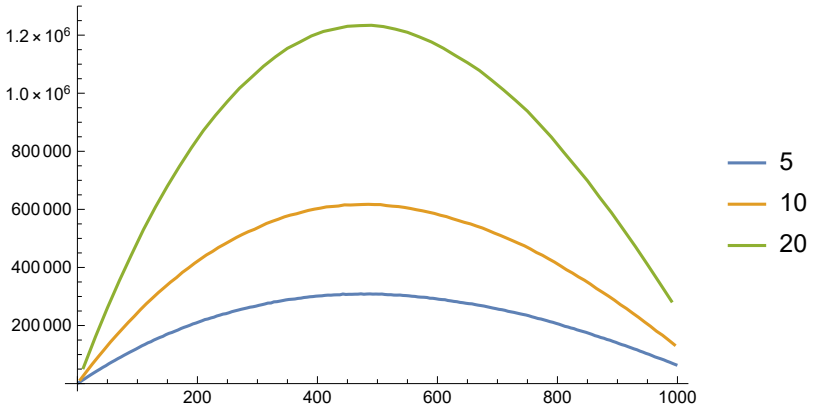
In[ ]:=
```
$dir = "C:\\Users\\roell\\OneDrive\\Dokumente_Uni\\Projekte\\Teaching\\Vorlesung\\Star
    Formation Ffm\\SS 20\\Q & A Sessions\\09.06.2020";
p1 = Import[FileNameJoin[{$dir, "P1data01.txt"}], "Table"];
p2 = Import[FileNameJoin[{$dir, "P1data02.txt"}], "Table"];
p3 = Import[FileNameJoin[{$dir, "P1data03.txt"}], "Table"];
p4 = Import[FileNameJoin[{$dir, "P1data04.txt"}], "Table"];
p5 = Import[FileNameJoin[{$dir, "P1data05.txt"}], "Table"];
p6 = Import[FileNameJoin[{$dir, "P1data06.txt"}], "Table"];
```

Out[ ]=



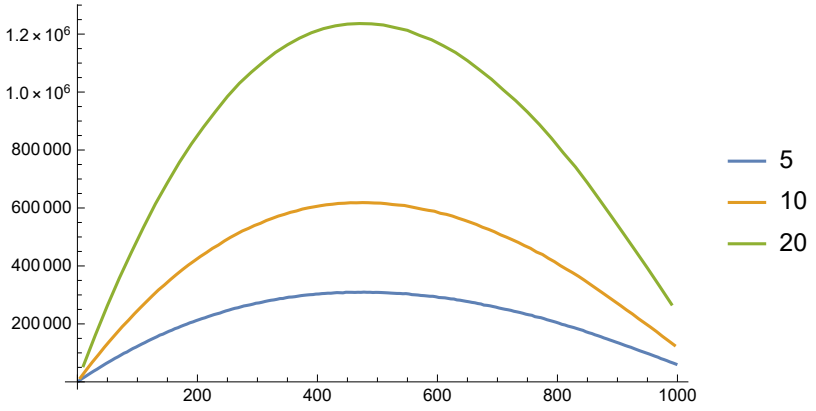
In[•]:=
```
ListLinePlot[{
  Transpose[{Mean /@ getBins[5], computeDistanceBins[p1, 5]}],
  Transpose[{Mean /@ getBins[10], computeDistanceBins[p1, 10]}],
  Transpose[{Mean /@ getBins[20], computeDistanceBins[p1, 20]}]},
 PlotLegends → {5, 10, 20}]
```

Out[•]=



In[•]:=
```
ListLinePlot[{
  Transpose[{Mean /@ getBins[5], computeDistanceBins[p2, 5]}],
  Transpose[{Mean /@ getBins[10], computeDistanceBins[p2, 10]}],
  Transpose[{Mean /@ getBins[20], computeDistanceBins[p2, 20]}]},
 PlotLegends → {5, 10, 20}]
```

Out[•]=



In[ ]:=
```
SeedRandom[1];
w11 = w1[p1, "RandomFieldNumber" → 10];
SeedRandom[1];
w31 = w3[p1, "RandomFieldNumber" → 10];
SeedRandom[1];
w12 = w1[p2, "RandomFieldNumber" → 10];
SeedRandom[1];
w32 = w3[p2, "RandomFieldNumber" → 10]; SeedRandom[1];
w13 = w1[p3, "RandomFieldNumber" → 10];
SeedRandom[1];
w33 = w3[p3, "RandomFieldNumber" → 10];
SeedRandom[1];
w14 = w1[p4, "RandomFieldNumber" → 10];
SeedRandom[1];
w34 = w3[p4, "RandomFieldNumber" → 10]; SeedRandom[1];
w15 = w1[p5, "RandomFieldNumber" → 10];
SeedRandom[1];
w35 = w3[p5, "RandomFieldNumber" → 10];
SeedRandom[1];
w16 = w1[p6, "RandomFieldNumber" → 10];
SeedRandom[1];
w36 = w3[p6, "RandomFieldNumber" → 10];
```

In[ ]:=
```
SeedRandom[1];
wLS1 = wLS[p1, "RandomFieldNumber" → 10, "Scaling" → "Log"];
SeedRandom[1];
wLS2 = wLS[p2, "RandomFieldNumber" → 10, "Scaling" → "Log"];
SeedRandom[1];
wLS3 = wLS[p3, "RandomFieldNumber" → 10, "Scaling" → "Log"];
SeedRandom[1];
wLS4 = wLS[p4, "RandomFieldNumber" → 10, "Scaling" → "Log"];
SeedRandom[1];
wLS5 = wLS[p5, "RandomFieldNumber" → 10, "Scaling" → "Log"];
SeedRandom[1];
wLS6 = wLS[p6, "RandomFieldNumber" → 10, "Scaling" → "Log"];
```

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

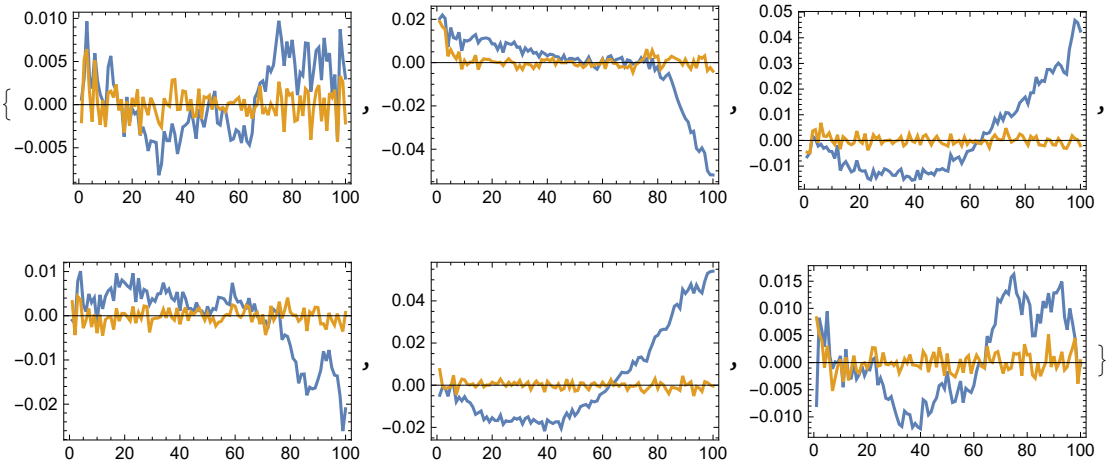
... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

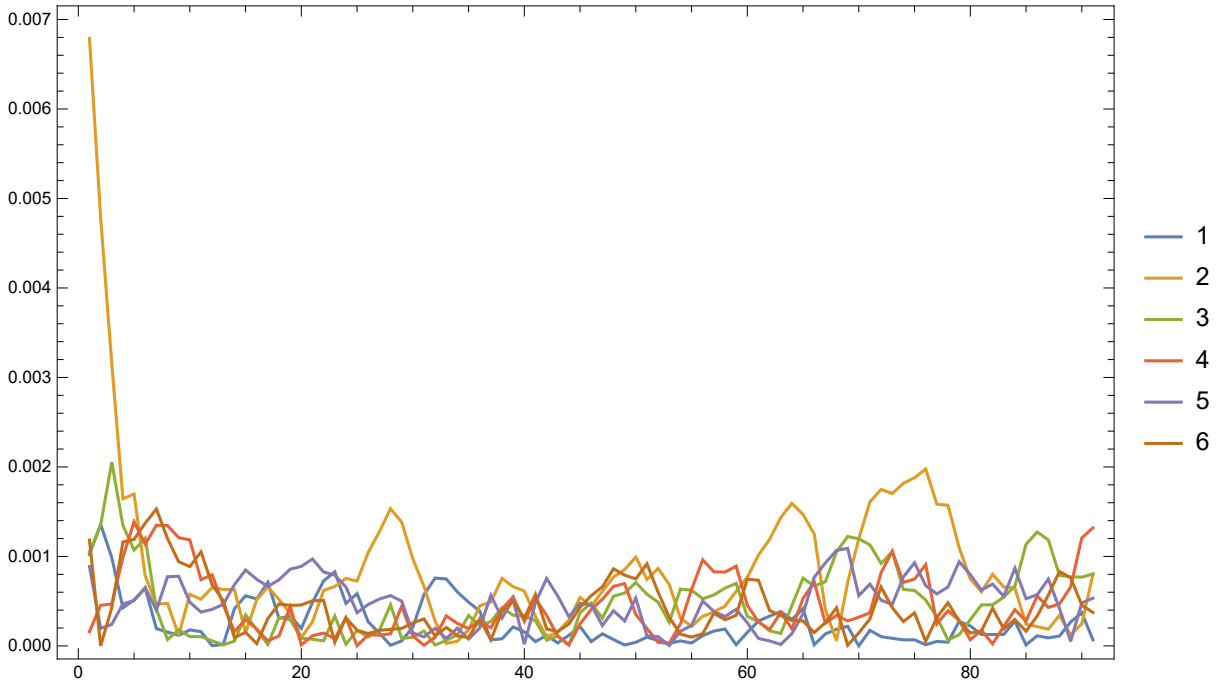
In[ ]:=
```
ListPlot[#, Joined → True, Frame → True, PlotRange → All] & /@
 {{w11, w31}, {w12, w32}, {w13, w33}, {w14, w34}, {w15, w35}, {w16, w36}}
```

Out[ ]=



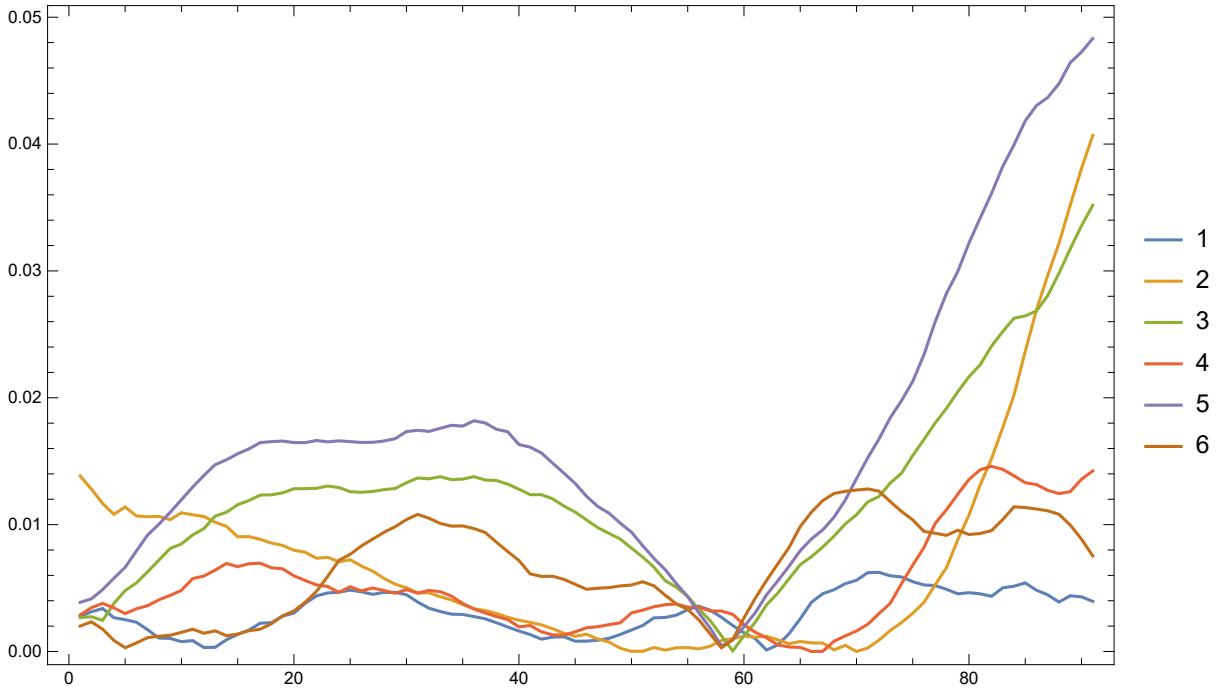
In[ ]:=
```
ListPlot[Abs@MovingAverage[#, 10] & /@ {w31, w32, w33, w34, w35, w36}, Joined → True,
 Frame → True, PlotRange → All, PlotLegends → Range[6], ImageSize → Large]
```

Out[ ]=



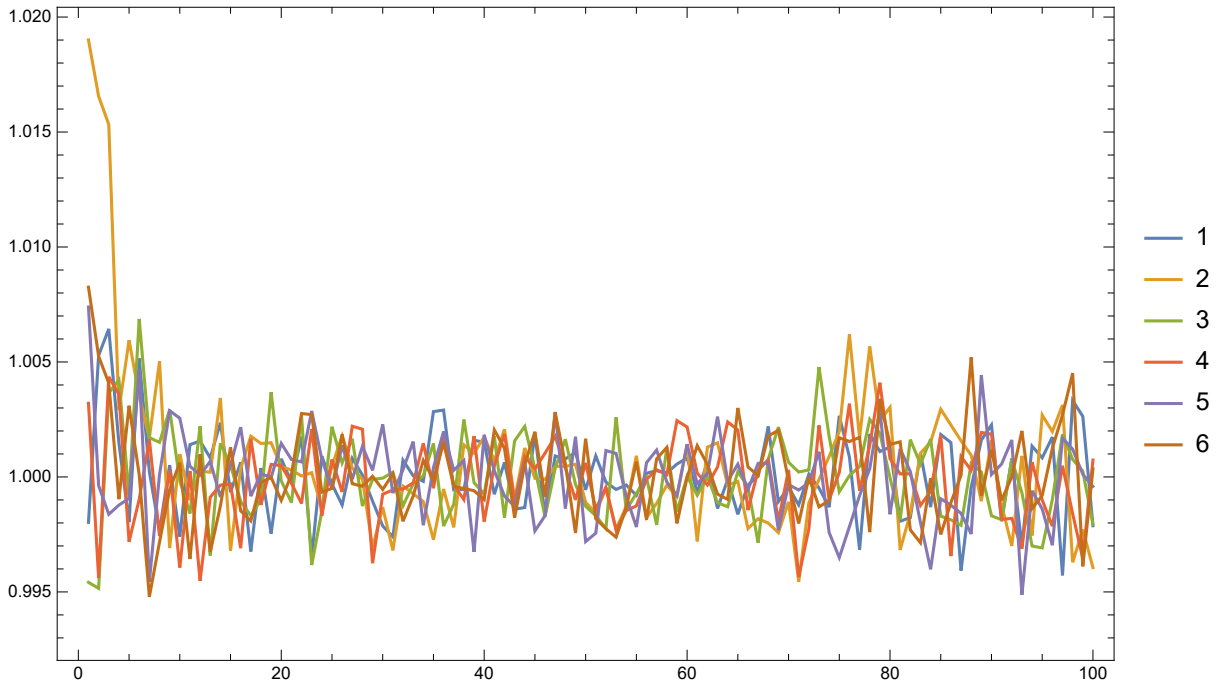
In[ ]:=
```
ListPlot[Abs@MovingAverage[#, 10] & /@ {w11, w12, w13, w14, w15, w16}, Joined → True,
 Frame → True, PlotRange → All, PlotLegends → Range[6], ImageSize → Large]
```

Out[ ]=



In[ ]:=
```
ListPlot[1 + {wLS1, wLS2, wLS3, wLS4, wLS5, wLS6}, Joined → True,
 Frame → True, PlotRange → All, PlotLegends → Range[6], ImageSize → Large]
```

Out[ ]=



In[•]:=
```
wLS1 = wLS[p1, "RandomFieldNumber" → 10, "Scaling" → "Log", "MinDistance" → 1];
```

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... Power: Infinite expression $\frac{1}{0}$ encountered.

... General: Further output of Power::infy will be suppressed during this calculation.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... Infinity: Indeterminate expression 0 ComplexInfinity encountered.

... General: Further output of Infinity::indet will be suppressed during this calculation.

In[•]:=
```
ListLogLogPlot[
 Transpose[{Mean /@ Partition[Power[10, #] & /@ findDivisions[{1, 1000}, 30], 2, 1],
   1 + wLS1}], PlotRange → All]
```

Out[•]=