FINAL PROJECT TUTORIAL

Credit Correlation (Sovereign CDS) Pseudo-Samples for Copula

Dr Richard Diamond
CQF ARPM

Techniques for **Credit Pricing**

- MATLAB demo of *ksdensity()*

to obtain pseudo-samples from historical data. Kernel function smoothes over **pdf**, and then **cdf** obtained.

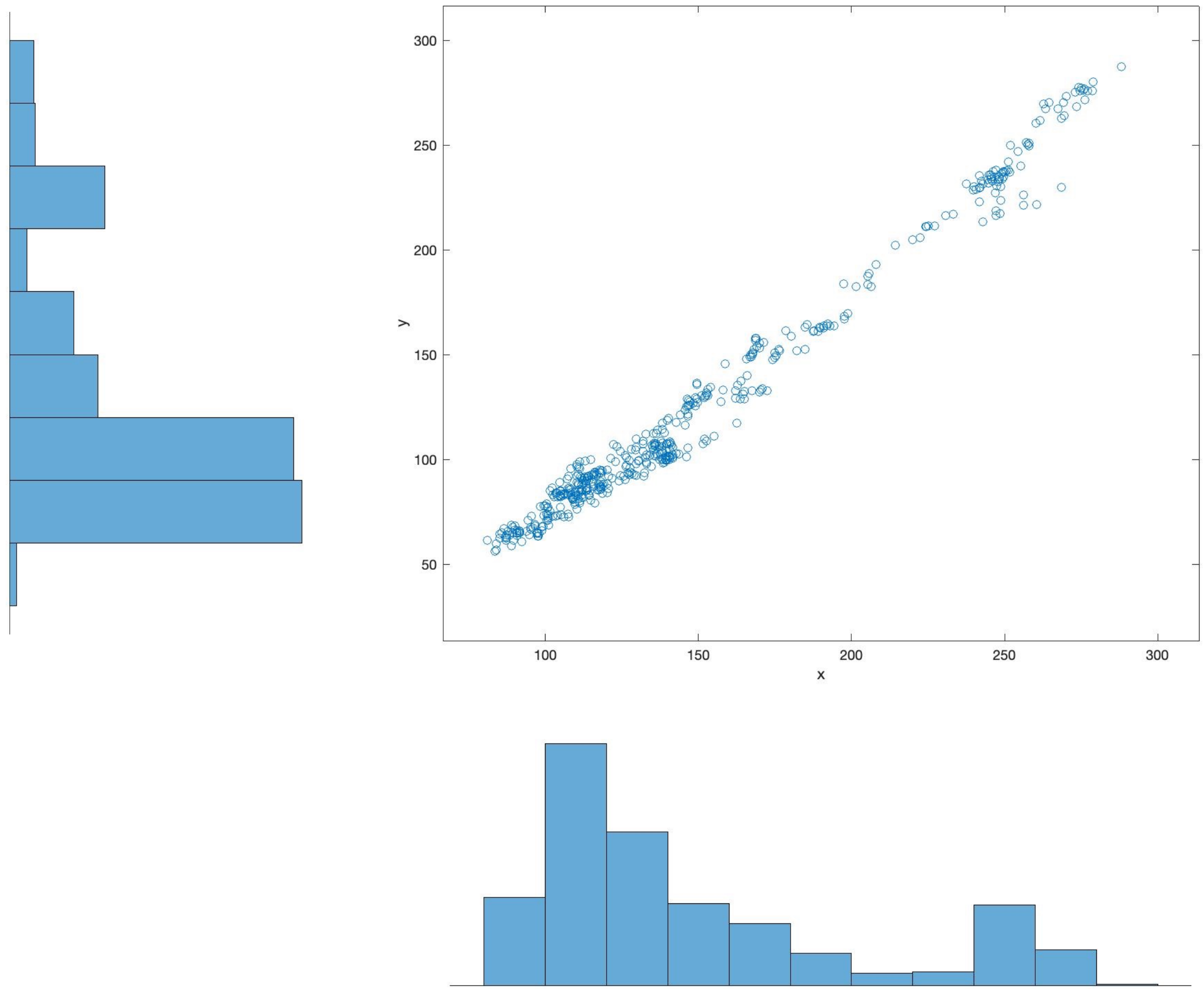
- Kernel smoothing function estimation in R and Python.

- Questions on spread pricing – as requested.

Input data are Sovereign CDS for Italy, Spain, Portugal and France.
For corporate names, using equity returns is <u>not inferior</u>.

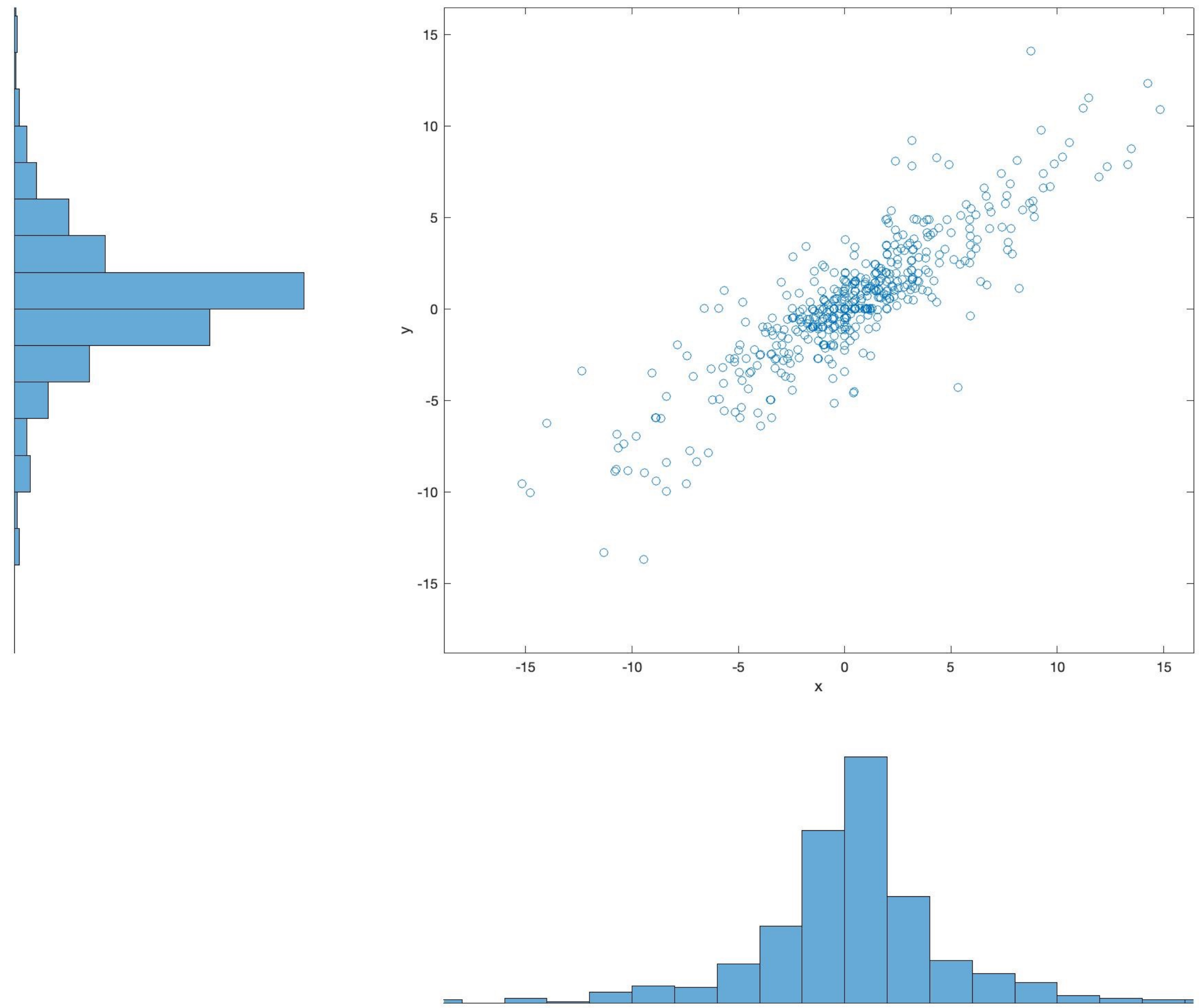
1 CDSHist_IT	2 CDSHist_ES	3 CDSHist_PT	4 CDSHist_FR
111.4700	91.8000	192.1400	47.3000
104.0300	82.2500	172.8400	48.3200
106.2510	83.5270	180.4500	49.3740
104.6690	82.4720	178.7200	49.3360
100.5800	76.7800	171.8500	49.3300
98.4400	77.6400	169.8300	49.3600
99.6300	77.6800	167.1300	49.1600
105.8100	82.8100	167.1800	49.6800
113.5100	86.4400	179.6400	49.9500
115.6500	86.9800	182.2800	46.3600
117.3100	88.5100	184.3400	45.3000
123.9900	89.8100	189.0800	45.3200
129.6400	92.4300	197.0100	44.7700
134.6500	96.6100	202.5400	45.3000
138.4800	99.7100	204.3800	45.3000
139.4500	99.7000	204.4000	45.8400
142.6500	102.9100	208.6800	46.3600

2D relationship scatterplots for CDS levels (eg, Italy vs Spain).



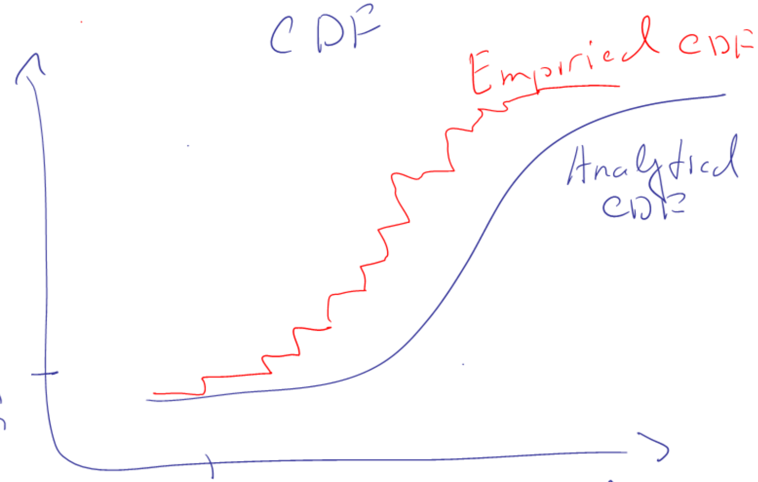
FitchLearning

2D relationship scatterplots for CDS **differences** (eg, Italy vs Spain).



Reminder from our CR Workshop drawings (Dr Richard Diamond)

Empirical CDF



1) Histogram & known analytical function is fitted
2) Integration (numerical) over that kernel function → $\hat{F}$
3) Fit over Empirical CDF

Matlab | R | Standard Scaler

Integrate over $\hat{f}_X(x)$ to get $\hat{F}_X(x)$

Empirical CDF

Matlab **ksdensity()** converts to pseudo-samples or “scores”, which are distributed uniformly.

$$U = \hat{F}(X)$$

```
%% Pseudo Samples by KS Method (via pdf and cdf estimation)

CDSPseudo = zeros(Nt_sampling-1,Nref);
for k=1:1:Nref
CDSPseudo(:,k) = ksdensity(CDSDiff(:,k),CDSDiff(:,k),'function','cdf'...
                ,'width',1.e-2);
end
```

Bandwidth 1.E-2=0.01. For your own data (CDS Diff or Equity Returns) regulate this parameter within the range:

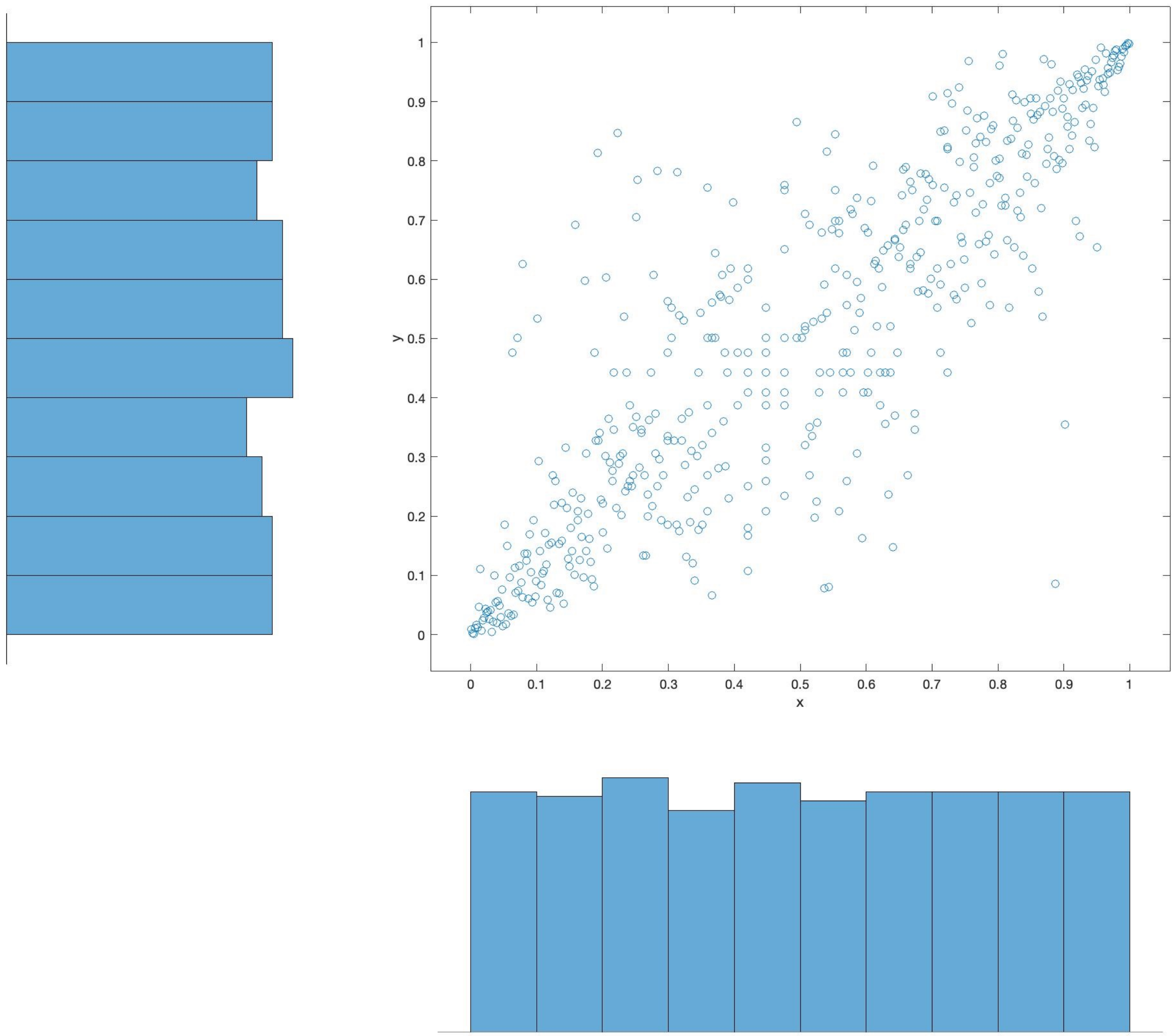
0.01 (‘smooth’ pdf) to 0.0005 (‘rough’ pdf)

- France CDS daily changes are 0 or technical 1bps, 2bps

Consider 2-day changes or more. Nobs will get reduced, however.

- KS Density will work on any variable (any distribution, including bimodal density of the typical random walk). However, check Empirical CDF.

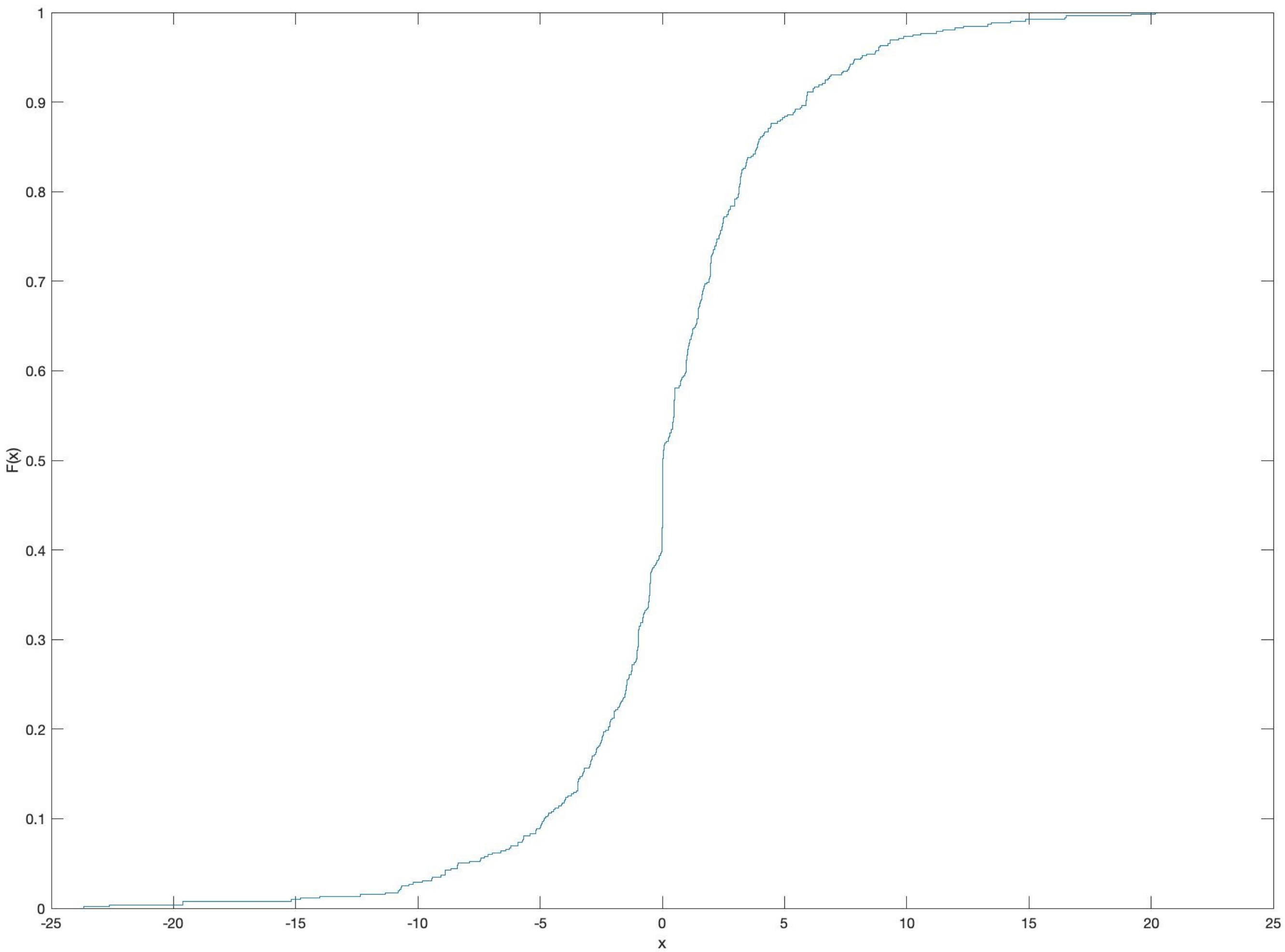
2D relationship scatterplots for **pseudo-samples** (eg, Italy vs Spain).



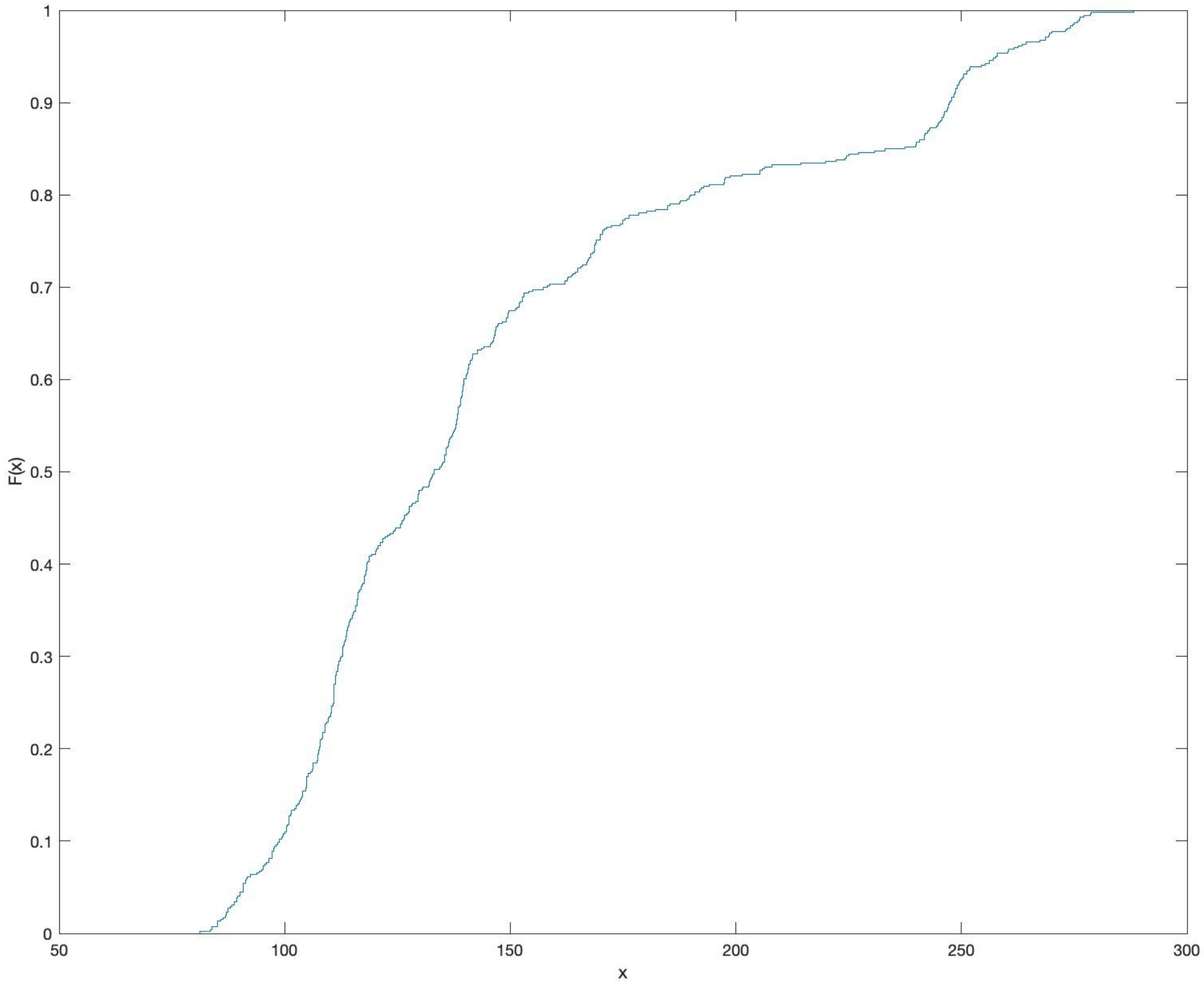
Code to obtain and plot the explicit Empirical CDF (also possible to just find the relevant plot in the PLOTS menu).

```
%% Empirical CDF Explorations - How smooth your CDF is?

k = 4 % for France -- Empirical CDF has sharp jump about zero
k =1  % for Italy -- Empirical CDF is SMOOTH and that makes all difference

[Fi,xi] = ecdf(CDSDiff(:,k));
figure()
stairs(xi,Fi,'b','LineWidth',2)
hold on

Fi_sm = ksdensity(CDSDiff(:,k), xi,'function','cdf'...
                ,'width',1.e-2);
plot(xi,Fi_sm,'r-','LineWidth',1.5)
xlabel('X1')
ylabel('Cumulative Probability')
legend('Empirical','Smoothed','Location','NW')
grid on
```

Empirical CDF for CDS differences (eg, Italy), Fhat(x) is related to x, such as +/- 5 bps, 10 bps, 15 bps… “somewhat steep but cdf-like”



Empirical CDF for historical CDS – credit spreads themselves (econometricians would say ‘levels’). **WHOA!**



Install and load **ks library in R** (free alternative to MATLAB)

```r
pseudo.uniform = function(X){
  # This function Calculates pseudo-uniform observations using ker
  # Requires 'ks' package to be loaded.

  # First we estimate the CDF
  Fhat <- kcde(X)
  # Plug in the values into the CDF to obtain pseudo-observations
  predict(Fhat, x=X)
}
```

Recipe 1. Complete solution but not quite ksdensity

sklearn.preprocessing.QuantileTransformer gives the uniform distribution as default output ('mapping to' uniform).

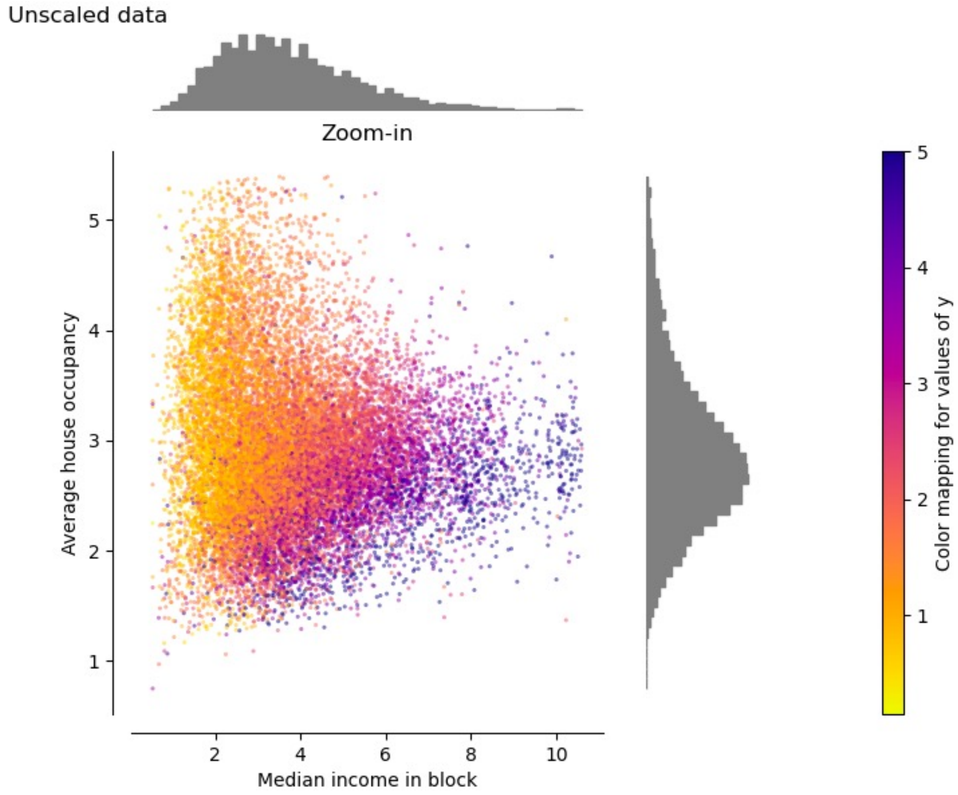
Useful for data with sparse range of values, eg "outliers that are common." Outliers are collapsed to [0,1] range, which is seen through saturation on 2D scatters.

```
trans = QuantileTransformer(n_quantiles=100, output_distribution='uniform')
data = trans.fit_transform(data)
```

Recommended to review:
https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

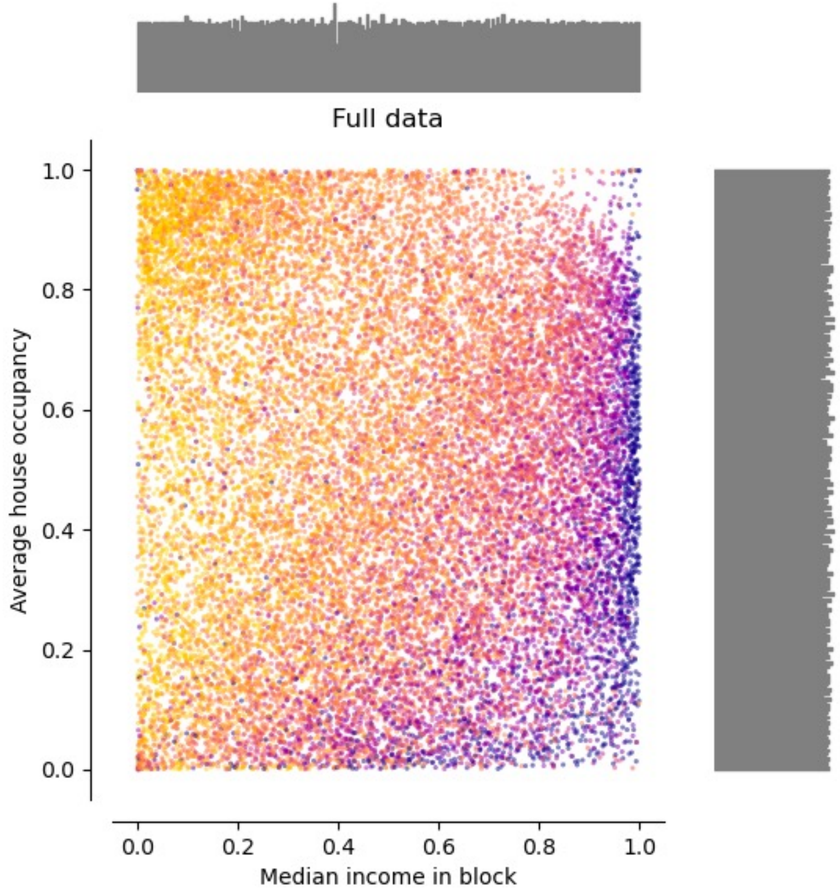
Another source (but complicated unnecessarily)
https://machinelearningmastery.com/quantile-transforms-for-machine-learning/

QuantileTransformer provides non-linear transformations in which distances between marginal outliers and inliers are shrunk.



The dataset X mapped to a uniform distribution with the range [0, 1].

Check for uniformity in histograms.



QuantileTransformer(output_distribution="uniform").fit_transform(X)

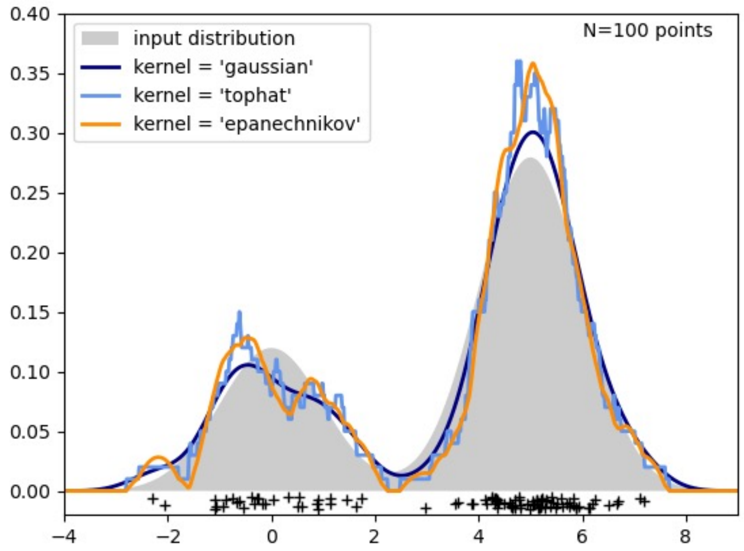
Recipe 2. Towards ksdensity, but incomplete

In terms of the kernel smoothing procedure down to **Empirical CDF**, Python ecosystem leaves the job unfinished

Kernel functions for **pdf** https://scikit-learn.org/stable/modules/density.html

```
from sklearn.neighbors import KernelDensity
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
kde = KernelDensity(kernel='gaussian', bandwidth=0.2).fit(X)
kde.score_samples(X)
```

where score_samples(X) are densities, and do not have to be between [0,1].



Alternative kde in scipy

```
import scipy.stats.gaussian_kde
```

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

Alternative on numerical integration

https://docs.scipy.org/doc/scipy/tutorial/integrate.html
Integrating using Samples section