Weights Vector $\underbrace{[1, \ldots \ldots]}_{\text{Length } \ell}$

Series : $\underbrace{[\qquad\qquad\qquad\qquad]}_{\text{Length } n}$

Transformation:

$$T : \mathbb{R}^{\ell} \Rightarrow \mathbb{R}^{n-\ell}$$

weights

$$\overline{X}_t = \sum_{k=1}^{\ell} w_k X_{t-k}$$

ℓ ℓ

Python Indices approximately:

$i - \ell$ to i (index w/o factor -1)

τ, k inequality:

$$\tau > |w_k| = \left| \prod_{i=0}^{k-1} d - i \left(\frac{1}{k!} \right) \right|$$

By inspection of recursive formula.

Thus, $k = k(d, \tau)$.

Fix $\tau \Rightarrow$ Find k.

Speed: $O(n) \Rightarrow \prod_{i=0}^{k-1}$ loop

<u>Recall</u>: Indices iterated over for X_t:

$$i\text{-width} \rightarrow i$$

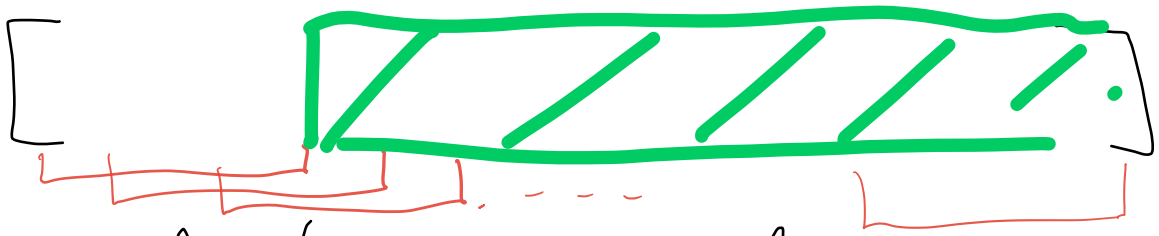
$$i \mapsto [\text{length(weights)}, \text{length(series)}]$$

i.e. Index Bounds for Dot Prod X_t:

$l = \text{length (weights)}$, $n = \text{length (series)}$

$$[l-l, l]$$
$$\vdots$$
$$[l+3-l, l+3]$$
$$\vdots$$
$$[n-l, n]$$

Series:



window size l (how many l-sized windows in n-sized series)

Forget about ± 1 python index,

<u>In general</u>: To drop k data pts, keep $n-k$ data pts.

($k = l = \text{length}(\text{weights})$.

<u>Recall $\tau = \tau(k,d)$:</u>

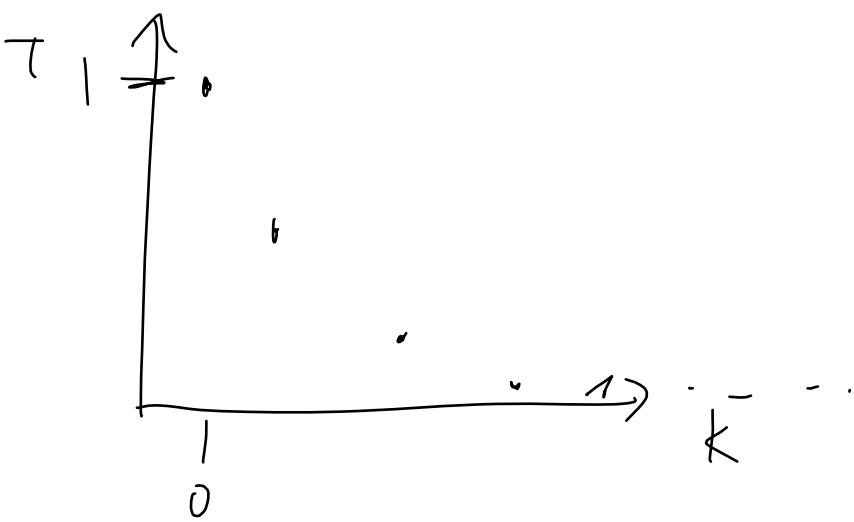
```
def calc_tau(k,d):
    val = 1
    for i in range(0,k):
        val *= (d-i)
    val *= 1/k!
    return val.
```

☆ - See code file/Test

$\Rightarrow$ Returns τ acceptable for max k bars.

Note: If k too big, for Π is too large for <u>python integer</u>!

Effects of k vs τ:



Variable Computation:

Fix k $\rightarrow$ thresh $\rightarrow$ changes adfuller test?

informs d

??

From previous trials: **Trend:**

$d \uparrow \rightarrow k \downarrow$

Functional Process:

Generate_graphs calls udf test

calls FFD

calls weights

In our inputs:

Fix range of $d = [0, 0.1, 1]$

(+) Fix a k,

$\Rightarrow$ range of thresh ($d \xleftrightarrow{\text{isomorphism}}$ unique τ)

For adfuller fcn: Fracdiff $\rightarrow$ correlation $\rightarrow$ pick best $\boxed{\text{pVal}}$

For a fixed d, k:

$\mapsto \tau$

$\to$ adfuller $\to$ check <u>pVal.</u>

Simply, it is τ vs d to find optimal d.

Previously, d vs τ:

$\tau(k)$	0.01	0.001	0.0001
$d_{optimal}$	0.5	0.3	0.2
k-drops	12	73	503

<u>ADF < 0.05 : stationary</u>

Note: do not use τ too small:

zero vector <u>breaks code</u>!

i.e. k small $\rightarrow$ τ big

$\rightarrow$ less drops $\rightarrow$ $d_{optimal}$ bigger