STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Fractionally Differentiated Features

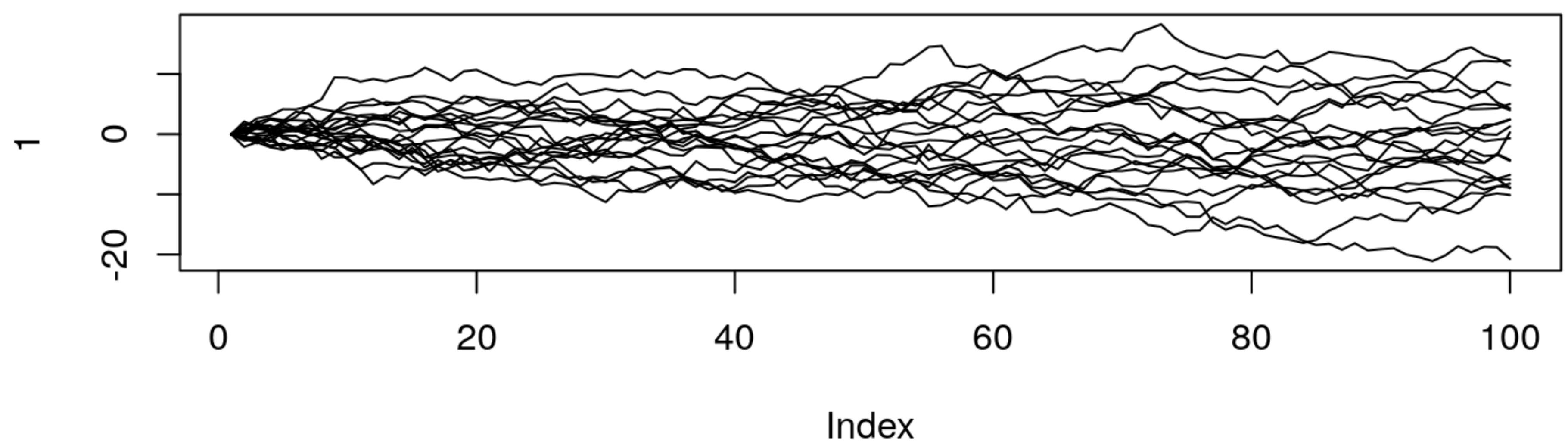
Stationarity

- Let $\{X_t\}$ be a stochastic process, if $E[X_t] = \mu$ and $E(X_t - \mu)(X_{t-j} - \mu) = \gamma_j$ for all t and j, then $\{X_t\}$ is covariance-stationary (i.e., weakly stationary). Strict stationarity means that the joint distribution of $(X_t, X_{t+j_1}, \cdots, X_{t+j_{n-1}})$ does not depend on t.

Some toy examples

- Random walk, a non-stationary process

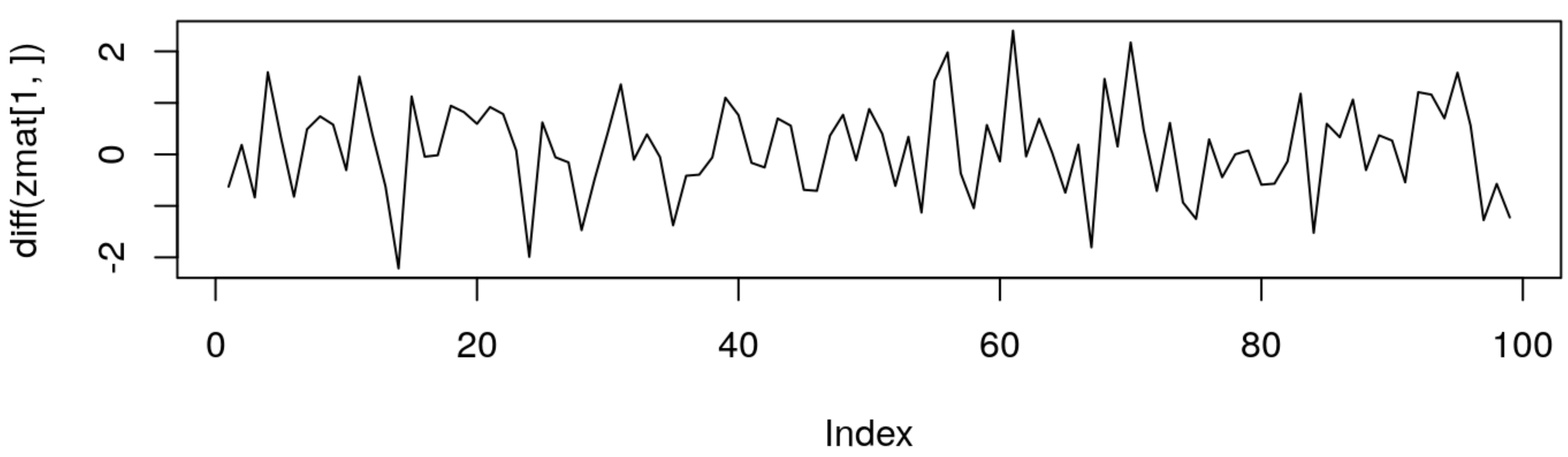
```
set.seed(1)
zmat <- matrix(0, 20, 100)
for(j in 1:20)
{  for(i in 2:100) zmat[j,i] <- zmat[j,i-1] + rnorm(1) }
plot(1, type="n", xlim=c(1,100), ylim=c(min(zmat), max(zmat)))
for(j in 1:20) lines(zmat[j,])
```



Some toy examples

- Random walk is an integrated process of order 1, denoted as $X_t \sim I(1)$. So, $Y_t := X_t - X_{t-1}$ is stationary and thus $Y_t \sim I(0)$.
  - A process X_t is integrated of order k if $(1 - L)^k X_t$ is stationary, where $(1 - L)X_t = X_t - X_{t-1}$

```
plot(diff(zmat[1,]), type="l")
```



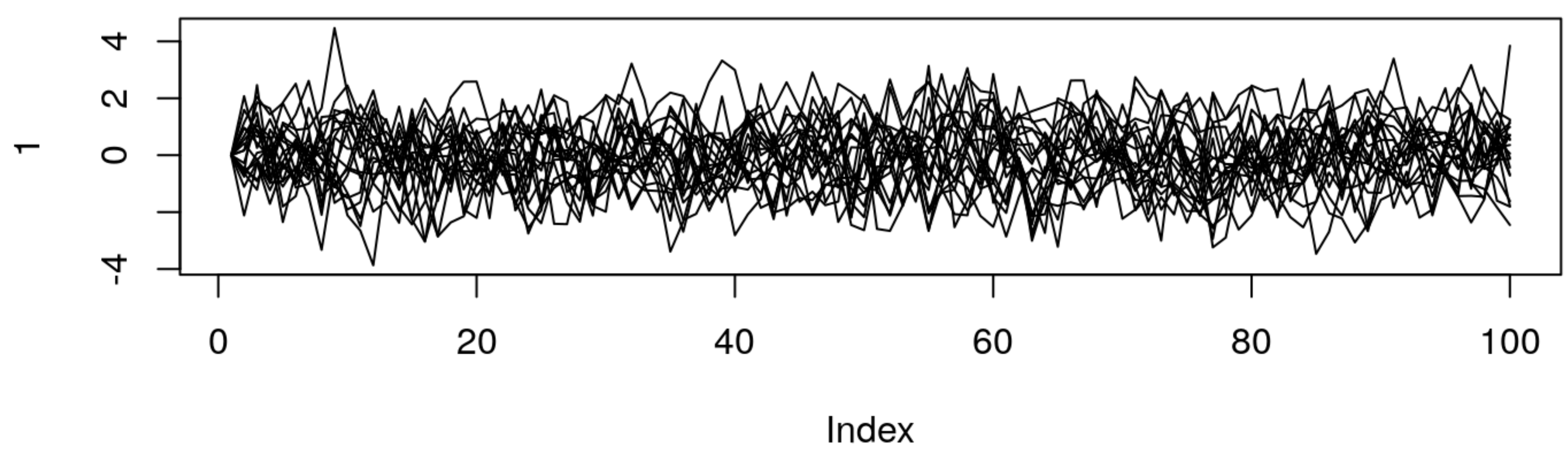
Unit root tests

- Unit root tests for (weak) stationarity of AR(p) process:
  - lag operator (L) or backshift operator (B): $BX_t = X_{t-1}$, eg., $(1-B)X_t = \Delta X_t = X_t - X_{t-1}$
  - Define characteristic function $\theta_p(B) := (1 - \alpha_1 B - \alpha_2 B^2 - \cdots - \alpha_p B^p)$, then AR(p) can be written as $\theta_p(B)X_t = w_t$, where w_t is a white noise.
  - Roots $|B_i| > 1$ if and only if AR(p) is stationary.

Examples - AR(1), stationary

- $\alpha_1 = 0.5$ so the root $B = 2 > 1$:

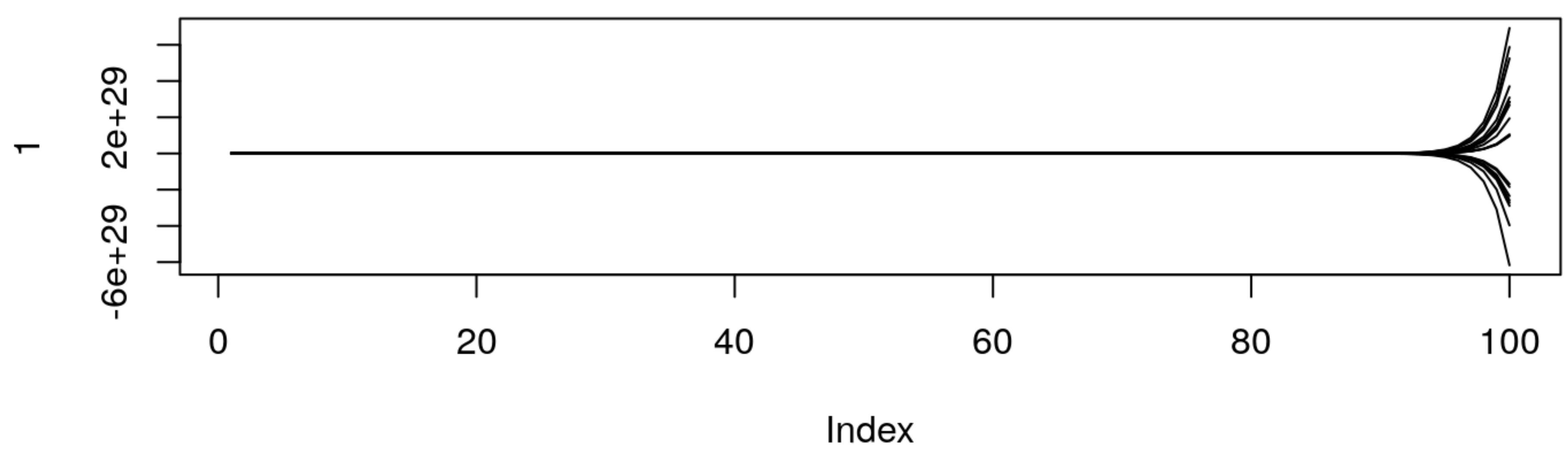
```
set.seed(1)
zmat <- matrix(0, 20, 100)
for(j in 1:20)
{  for(i in 2:100) zmat[j,i] <- (0.5)*zmat[j,i-1] + rnorm(1) }
plot(1, type="n", xlim=c(1,100), ylim=c(min(zmat), max(zmat)))
for(j in 1:20) lines(zmat[j,])
```



Examples - AR(1), non-stationary

- $\alpha_1 = 2$ so the root $B = 0.5 < 1$:

```
set.seed(1)
zmat <- matrix(0, 20, 100)
for(j in 1:20)
{  for(i in 2:100) zmat[j,i] <- (2)*zmat[j,i-1] + rnorm(1) }
plot(1, type="n", xlim=c(1,100), ylim=c(min(zmat), max(zmat)))
for(j in 1:20) lines(zmat[j,])
```



Unit root tests

- Unit root test: tests whether a process possesses a unit root and thus non-stationary.
- Augmented Dickey-Fuller Test (ADF)
- Based on AR(p)

$$\begin{aligned}\Delta X_t =& \alpha + \beta t + \gamma X_{t-1} + \\ & \delta_1 \Delta X_{t-1} + \cdots + \delta_{p-1} \Delta X_{t-p+1} + w_t\end{aligned}$$

- Drift: α; Linear trend: βt
- $H_0 : \gamma = 0, H_1 : \gamma < 0$
- The terms $\Delta X_{t-1}, \cdots, \Delta X_{t-p+1}$ are used to account for serial correlations in residuals
- No universally optimal lag p; use several different unit root tests.

Unit root tests

- R functions
  - tseries::adf.test: only linear trend model
  - CADFtest::CADFtest
  - urca::ur.df
  - fUnitRoots::adfTest

Unit root tests

- Phillips-Perron Test (PP)
  - Based on ADF, and correct for any serial correlation and heteroskedasticity in w_t
  - R functions:
    - tseries::pp.test
    - stats::PP.test
- ADF and PP tests have high Type II error when alternatives are closer to I(1)
  - e.g., $X_t = 0.95X_{t-1} + w_t$ is stationary but the null hypothesis may fail to be rejected

```r
set.seed(1)
z <- rep(0,100)
for(i in 2:100) z[i] <- (0.95)*z[i-1] + rnorm(1)
tseries::adf.test(z)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  z
## Dickey-Fuller = -2.9849, Lag order = 4, p-value = 0.1686
## alternative hypothesis: stationary
```

Unit root tests

- Elliot, Rothenberg, and Stock's efficient unit root test
  - For maximum power against very persistent alternatives
  - R functions:
    - urca::ur.ers

```
library(urca)
rst <- summary(ur.ers(z, model = "trend", lag.max = 4))

## Value of test-statistic is: -5.1622
## Critical values of DF-GLS are:
##                  1pct  5pct 10pct
## critical values -3.48 -2.89 -2.57
```

Stationarity tests

- Kwiatkowski, Phillips, Schmidt and Shin Stationarity Test (KPSS)
  - The null is I(0), unlike unit root tests
  - R functions:
    - tseries::kpss.test
- Unit root tests and Stationarity tests might lead to contradictions, which may suggests structural breaks

Motivation

- **Changes** in log prices/yields/volatilities make the series stationary while removing all memory from the original series (eg., the price process itself)
- Memory is the basis for the model's predictive power.
- Order of integration: minimum number of differences required for stationary
  - cointegrated series: I(0)
  - log prices: I(1)
- The dilemma: log returns are stationary but memory-less, and prices have memory but are non-stationary.
- What is the minimum amount of differentiation that makes price series stationary while preserving as much memory as possible?

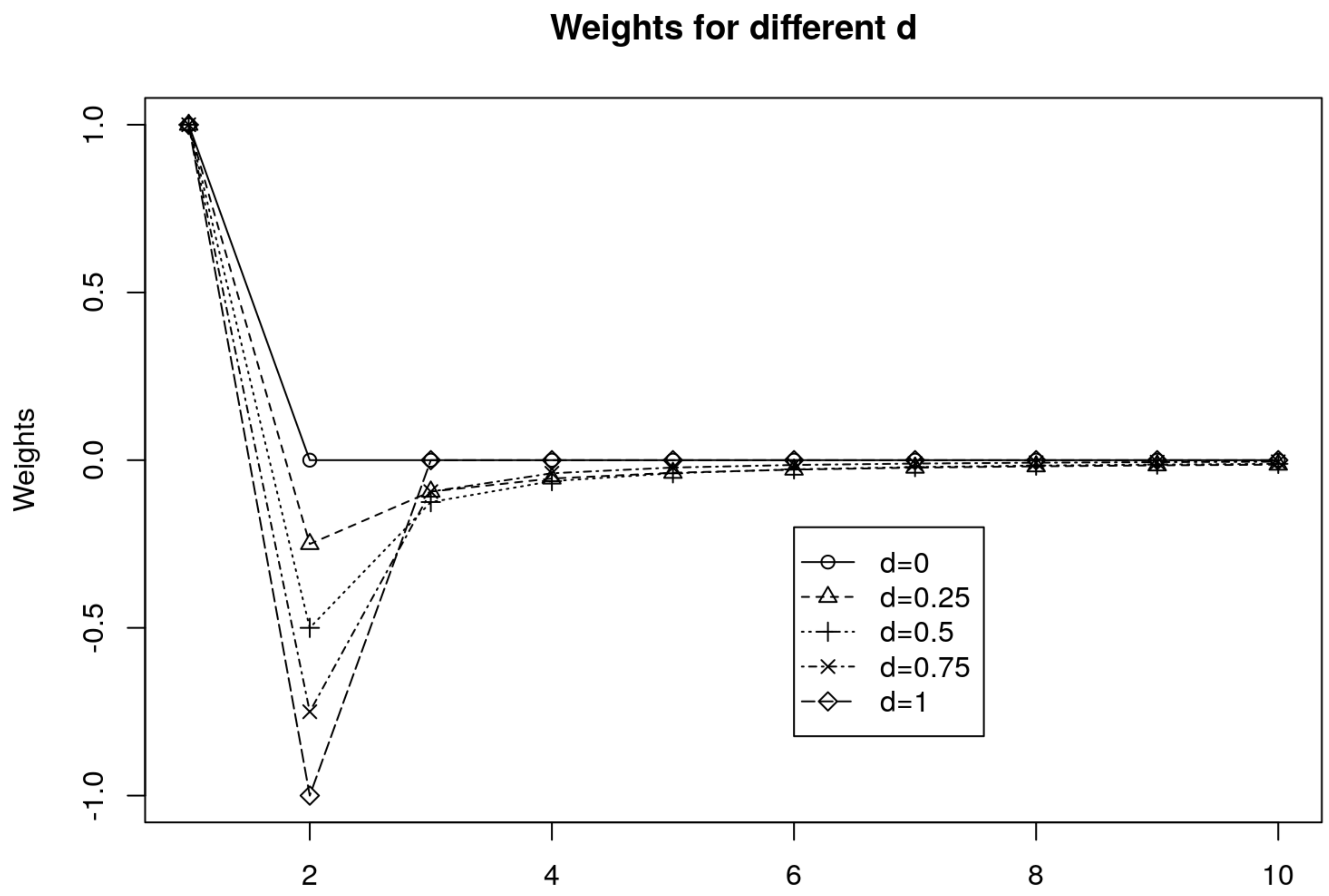
The method

- Recall: $(1-B)X_t = X_t - X_{t-1}$
- Binomial series expansion:
  - $$(1-B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k =$$
    $$1 - dB + \frac{d(d-1)}{2!}B^2 - \frac{d(d-1)(d-2)}{3!}B^3 + \cdots$$
- d is not necessarily an integer, and the same idea used for ARFIMA model

The method

- Let $X = \{X_t, X_{t-1}, \cdots, X_{t-k}, \cdots\}$, the weights $w = \{1, -d, \cdots, (-1)^k[\prod_{i=0}^{k-1}(d-i)]/k!, \cdots\}$
- Then $(1-B)^d X_t = X \cdot w = \sum_{k=0}^{\infty} w_k X_{t-k}$ preserves the memory when d is **NOT** an integer
- With d integer, memory is dropped:
  - When $d = 1$, $w = \{1, -1, 0, 0, \cdots\}$
  - When $d = 2$, $w = \{1, -2, 1, 0, 0, \cdots\}$
- The weights can be generated iteratively:
  - $w_k = -w_{k-1}(d-k+1)/k$, with $w_0 = 1$
  - When k large enough, $|(d-k+1)/k| < 1$, so $w_k \to 0$.

The method - weights comparison



How to choose the value of d

- Trade off between stationarity and memory
- Choose smallest d that just passed some unit root and/or stationarity tests.
- See Figure 5.5 in AFML

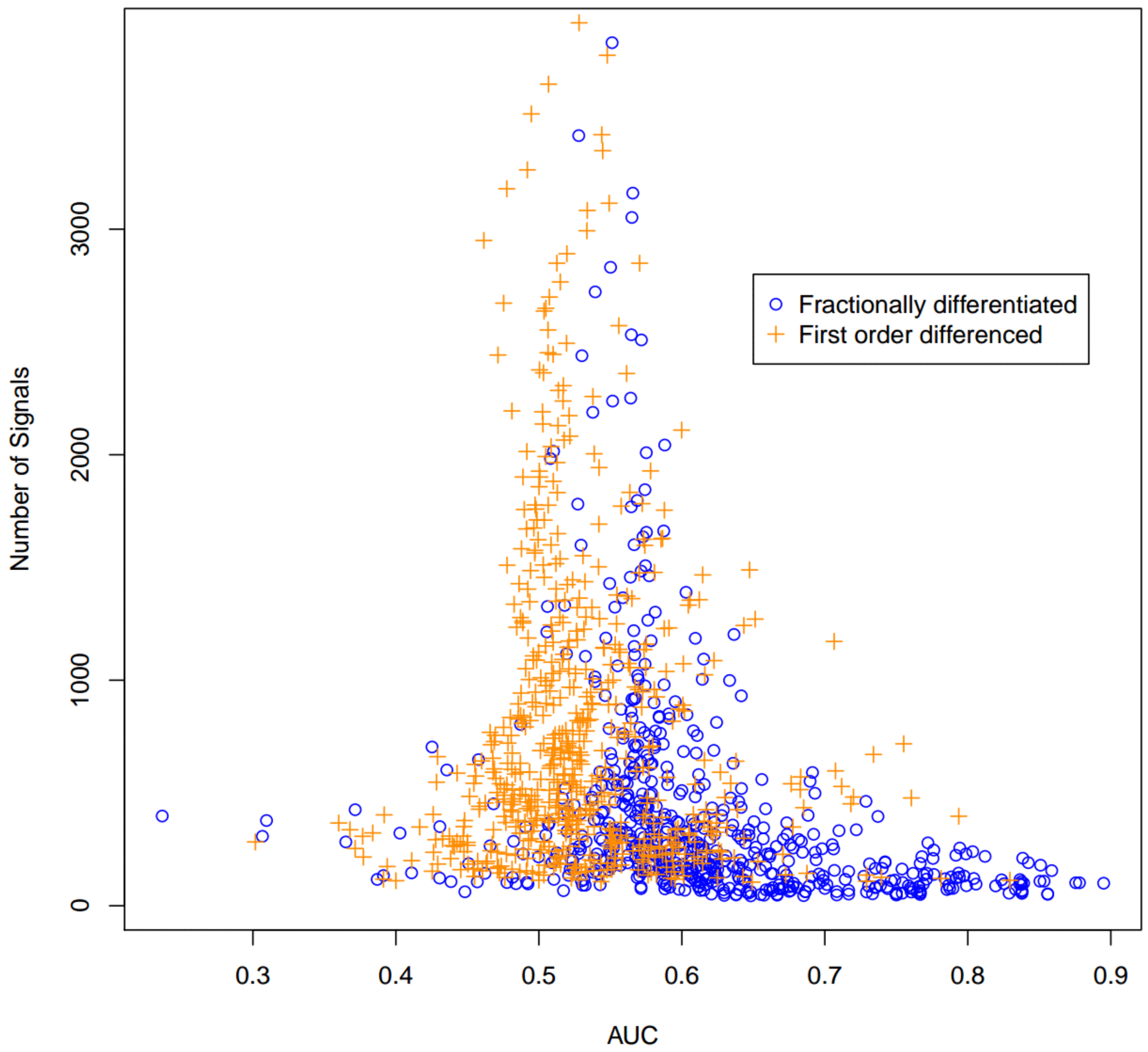
Implementation - Fixed-width window method

- Drop the weights after $|w_k|$ falls below a given threshold τ
- $\tilde{X}_t = \sum_{k=0}^{l^*} w_k X_{t-k}$ for $t = l^* + 1, \ldots, T$.

```
#' @param x a vector of time series to be fractionally differentiated
#' @param d the order for fractionally differentiated features
#' @param nWei number of weights for output
#' @param tau threshold where weights are cut off; default is NULL, if not NULL
#'            then use tau and nWei is not used
fracDiff <- function(x, d=0.3, nWei=10, tau=NULL){
  weig <- weights_fracDiff(d=d, nWei=nWei, tau=tau)
  nWei <- length(weig) # the first one in x that can use all the weights
  nx <- length(x)
  rst <- rep(NA, nx)
  rst[nWei:nx] <- sapply(nWei:nx, function(i){ sum(weig*x[i:(i-nWei+1)]) })
  return(rst)}
```

- Try R

Benefits of fracDiff - XBTUSD example



- Back to Course Scheduler