COMPUTATIONAL FINANCE & RISK MANAGEMENT
UNIVERSITY of WASHINGTON
Department of Applied Mathematics

Statistical Modeling in R

CFRM 425 (004)

R Programming for Quantitative Finance

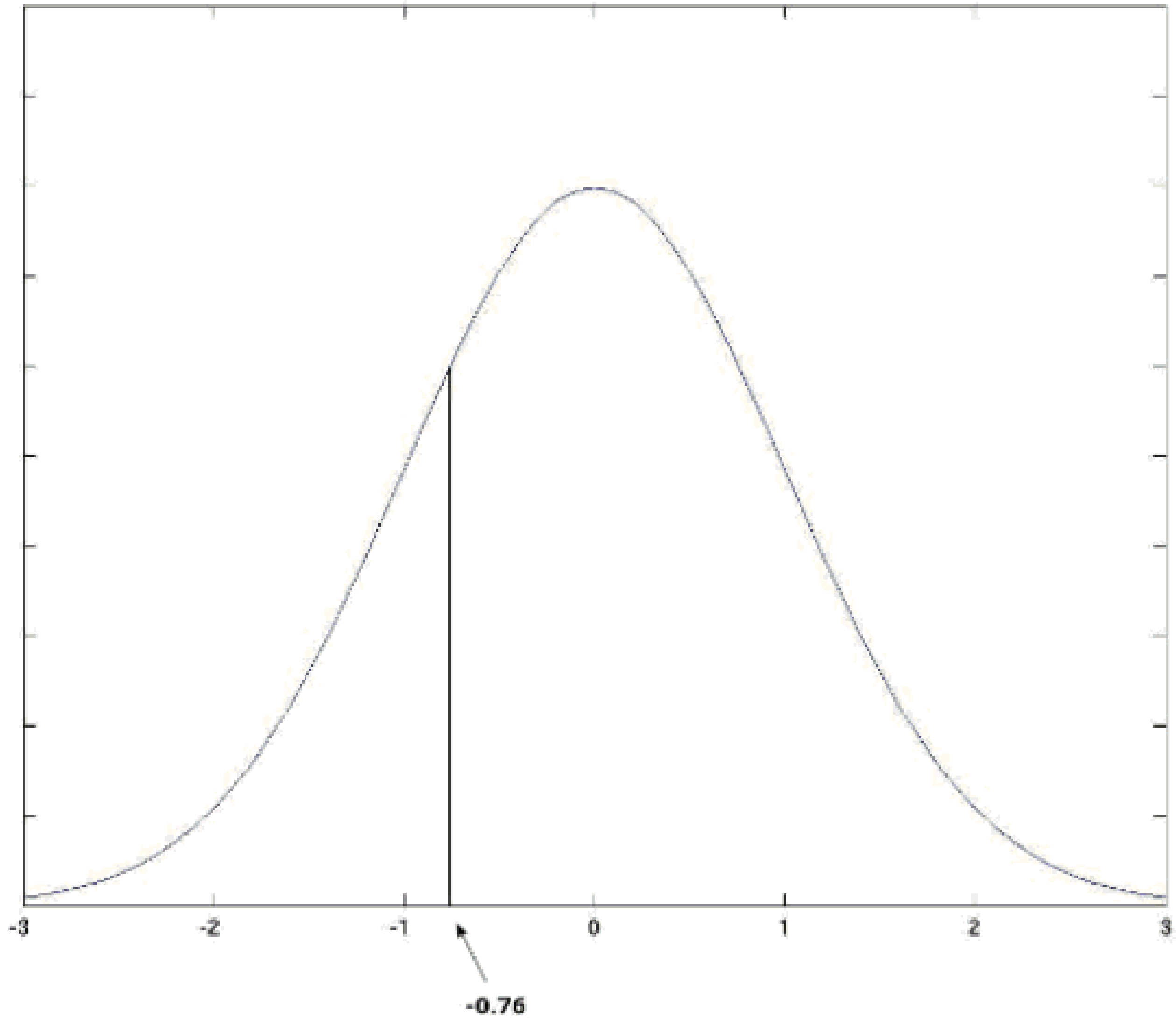
References/Reading/Topics

- Reading: Jeet and Vats, Ch 2 [JV] (selected sections)

- Topics:
  - The Normal Distribution
    - pdf
    - cdf
    - quantile
    - random sampling
  - Other Distributions
    - The Lognormal Distribtution
    - The Uniform Distribution
    - Fat-tailed Distributions (in place of EVT, but related)
      - t-distribution
      - Generalized Lambda Distribution
  - Data Sampling
  - Distribution Parameters and Moments
  - Standardization

The Normal Distribution in R

https://internal.ncl.ac.uk/ask/numeracy-maths-statistics/statistics/distributions/normal-distribution.html



- The Normal Distribution
- A very widely used probability distribution in the financial industry
- It is a bellshaped curve, with mean, median mode all the same value
- It is denoted by $N(\mu, \sigma^2)$, where
  - μ is the mean, and
  - σ^2 is the variance of the distribution
- The standard normal distribution has mean 0 and variance 1, and it is denoted by $N(0,1)$
  - Note there is a typo in the book (they put N(1, 0); should be N(0, 1)
  - Be careful: Many programming languages (including R) use the standard deviation as the 2nd parameter rather than the variance
  - For many models and problems in theoretical finance, $N(0,1)$ is the underlying distribution used; scaling by the volatility of an asset price or return – in reality the standard deviation – is applied "after the fact"

The Normal Distribution

- A Standard Normal random variable is typically denoted by Z, viz, $Z \sim N(0, 1)$

- The cdf of the standard normal distribution is often denoted by
$$\Phi(z)$$

- The pdf of the standard normal distribution is often denoted by
$$\phi(z)$$

- We can naively attempt to fit a dataset to a normal distribution, using the mean and standard deviation of the data sample (shown in text – will discuss shortly)

The Normal Distribution

- Base R provides the following functions
  - **`dnorm(x, mean = 0, sd = 1, …)`**
  - **`pnorm(x, mean = 0, sd = 1, …)`**
  - **`qnorm(p, mean = 0, sd = 1, …)`**
  - **`rnorm(n, mean = 0, sd = 1)`**

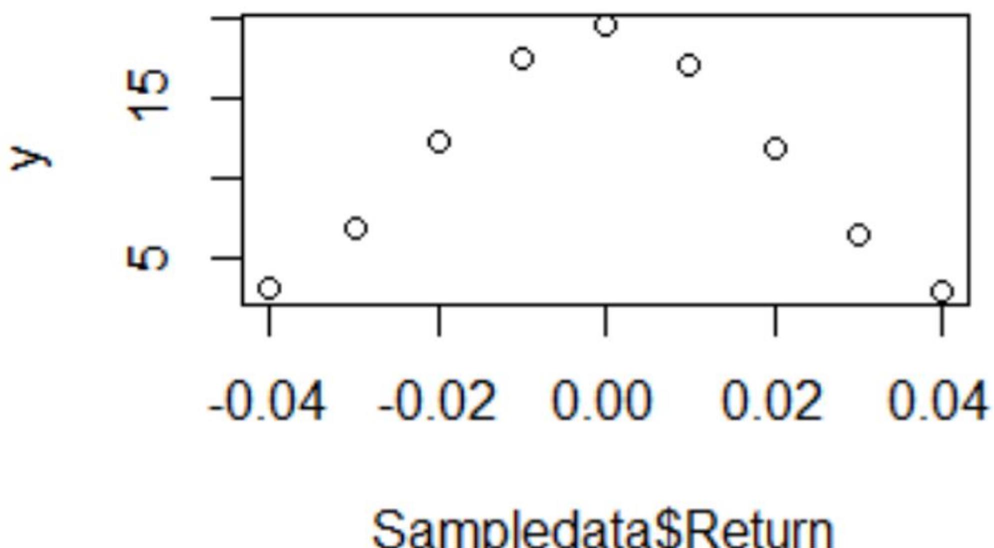
- Where, for given parameters mean and standard deviation
  - **`dnorm`** : pdf f(x)
  - **`pnorm`** : cdf F(X)
  - **`qnorm`** : Quantile function for percentile p
  - **`rnorm`** : random draw from the distribution

- In general, as we shall see, any distribution provided in R is expected to have all four functions implemented: **`dxxxx(.)`** **`pxxxx(.)`** **`qxxxx(.)`** **`rxxxx(.)`**

- Back to fitting the data to a normal distribution, per the book, we will plot the pdf as follows:
  - The x values are the returns from the data
  - The y values are the values of the pdf for each return value x
  - The mean and standard deviation parameters are taken as those of the return data:

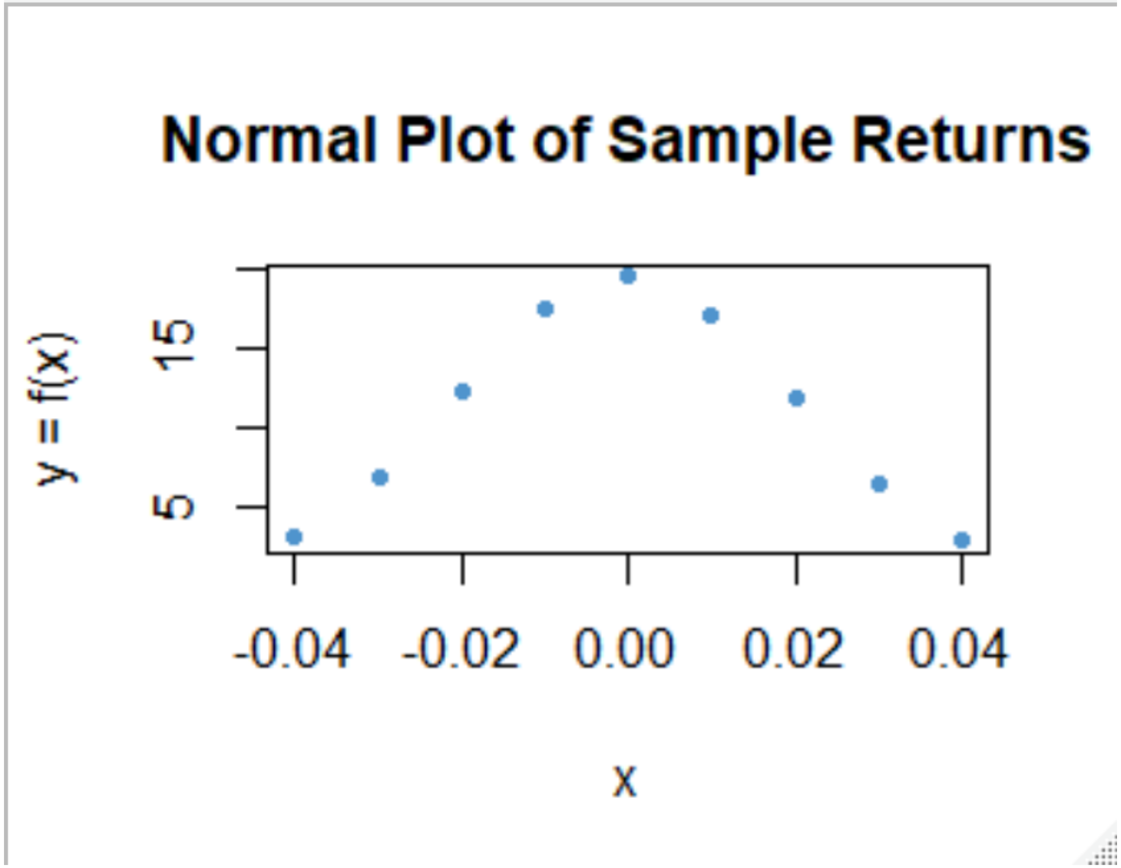
```
y <- dnorm(Sampledata$Return, mean = mean(Sampledata$Return),
           sd = sd(Sampledata$Return))

# Then plot the (x, y) pairs (we will discuss
# details about plots later):
plot(x = Sampledata$Return, y = y)
```



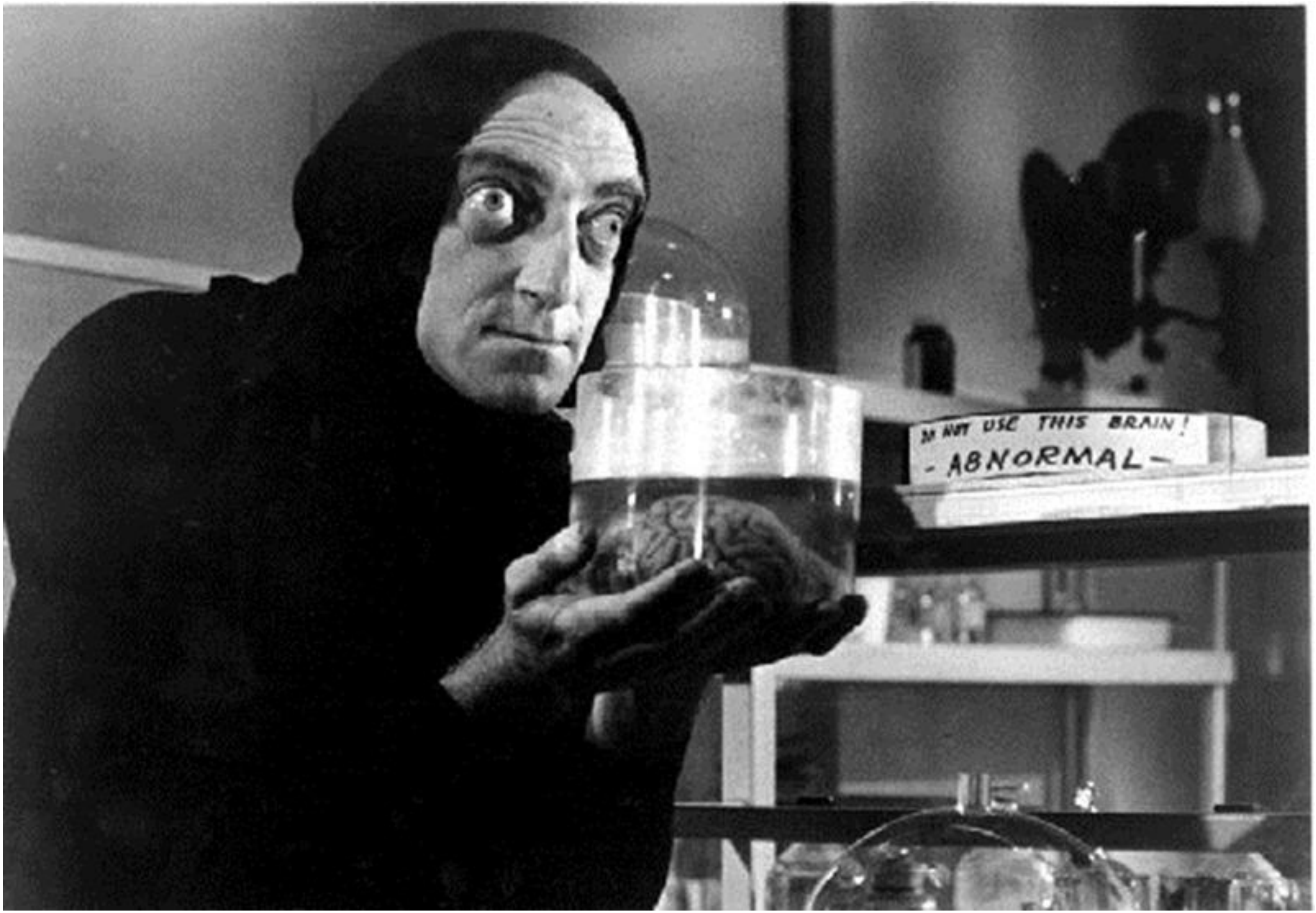
- We can improve the plot somewhat as follows:

```
plot(x = Sampledata$Return, y = y,
     pch = 16, cex = 0.75 , col = "steelblue3",
     main = "Normal Plot of Sample Returns",
     xlab = "x", ylab = "y = f(x)" )
```



- Remarks:
  - The data have been automatically binned in the previous examples
  - We will discuss the details about this later
  - We will also cover plotting in general in further detail later
    - ➢ Plotting in Base R
    - ➢ Plot functions within other packages
    - ➢ Specific plotting packages
      - plot.ly
      - ggplot2
    - ➢ Overloaded **`plot(.)`** function in other packages
      - zoo
      - xts
      - others

Other Distributions in R



The Lognormal Distribution

- Definition: A random variable Y has a lognormal distribution when the random variable $X = \ln Y$ has a normal distribution
- The book goes more into detail, but the important point in finance is that making the assumption that asset returns on a series of prices $\{S_t\}$ is lognormal ties in with the no-arbitrage pricing theory, which is at the heart of the Black-Scholes-Merton model for option pricing
- In other words, each return in the series $\left\{\frac{S_t}{S_{t-1}}\right\}$ is a lognormal random variable
- In practice, it's rare to work with the lognormal functions directly, and in fact most random sampling used in no-arbitrage pricing-theoretic models is from a normal distribution
- The no-arbitrage theory does have one particularly severe limitation: the normal distribution cannot realistically capture the fat tails present in empirical asset returns

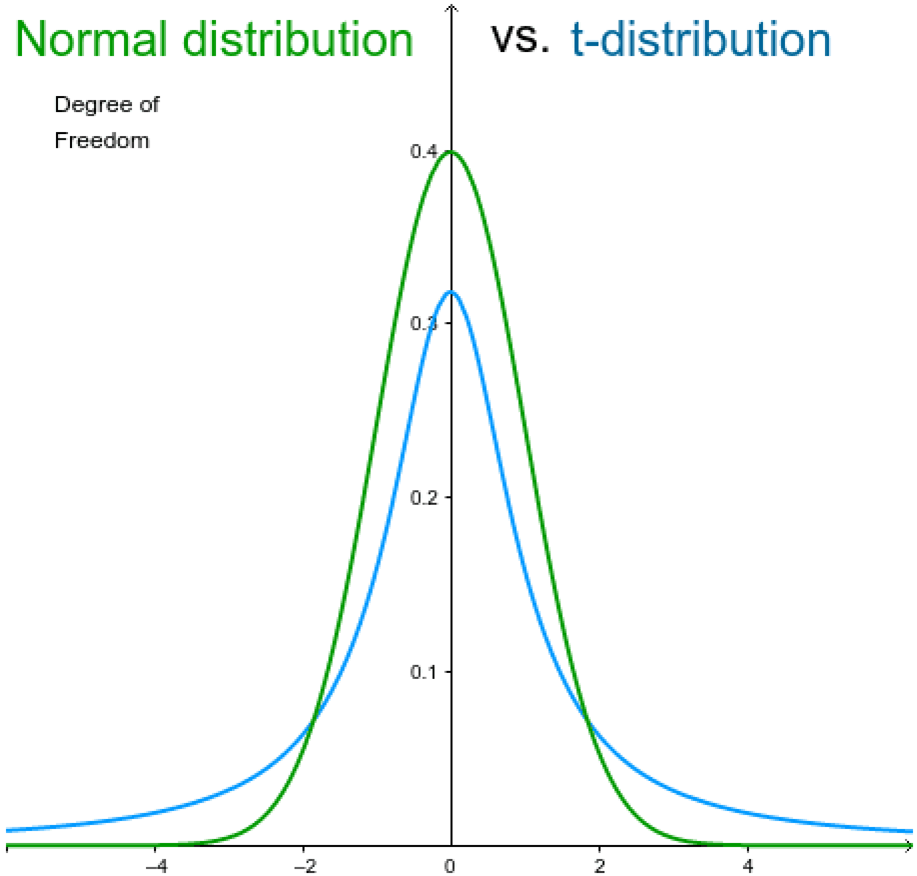
The Uniform Distribution

- The Uniform Distribution over the interval $[0, 1]$ is a distribution that is used with some frequency in practice
- One specific application is constructing a non-parametric cumulative density function (we will discuss this later in the course)
- For now:
  - **`dunif(x, min = 0, max = 1, …)`**
  - **`punif(x, min = 0, max = 1, …)`**
  - **`qunif(p, min = 0, max = 1, …)`**
  - **`runif(n, min = 0, max = 1)`**
- Remark: When generating random numbers from a distribution, if reproducible results are desired, then one should set a specific seed value; viz,

```
set.seed(106)
u <- runif(1000)          # 1000 random draws from U[0, 1]
set.seed(874)
v <- rnorm(1000)          # 1000 random draws from N(0, 1)
```

The Student t distribution

- The Student t Distribution is also bell-shaped and symmetric like the normal distribution, but it can be scaled to capture fat tails



The Student t distribution

- In Base R:

```
dt(x, df)          # pdf(x)
pt(x, df)          # cdf(x)
qt(p, df)          # quantile(p)
rt(n, df)          # random sample from distribution
```

- The t-distribution improves the fit of financial returns due to its higher kurtosis than the normal distribution (can fit fat tails better)
- However, being a symmetric distribution, it cannot capture any skewness present in returns
- Become more common as returns become more frequent

Four Parameter Distributions

- In Base R (not exhaustive):
  - Students t, with noncentrality parameter
  - Cauchy
  - Beta
  - Weibull

- In CRAN Packages (also not exhaustive)
  - Generalized Lambda
  - Generalized Hyperbolic
  - Skew-t
  - Stable Family

- Remark: Four-parameter distributions are relevant to Extreme Value Theory (EVT), although this is not explicitly covered in Ch 2. We will return to this later.

- We have covered sampling from a distribution; eg **`rnorm(.)`**

- We can also sample from a set of empirical data; eg,

```
# Without replacement:
set.seed(106)
wor <- sample(x = Sampledata$Return, size = 50, replace = FALSE)

# With replacement:
set.seed(5863)
wr <- sample(x = Sampledata$Return, size = 50, replace = TRUE)
```

- Two applications in finance (among others)
  - Monte Carlo simulations of trading strategies
  - Constructing empirical probability distributions

Statistical Measures

Statistical Measures

- The usual:
  - **`mean(.)`**
  - **`median(.)`**
  - **`mode(.)`**
  - **`var(.)  # Variance`**
  - **`sd(.)   # Standard Deviation`**
  - **`min(.), max(.)`**

are functions in Base R, that operate on individual vectors or columns of data (matrix, array, dataframe)

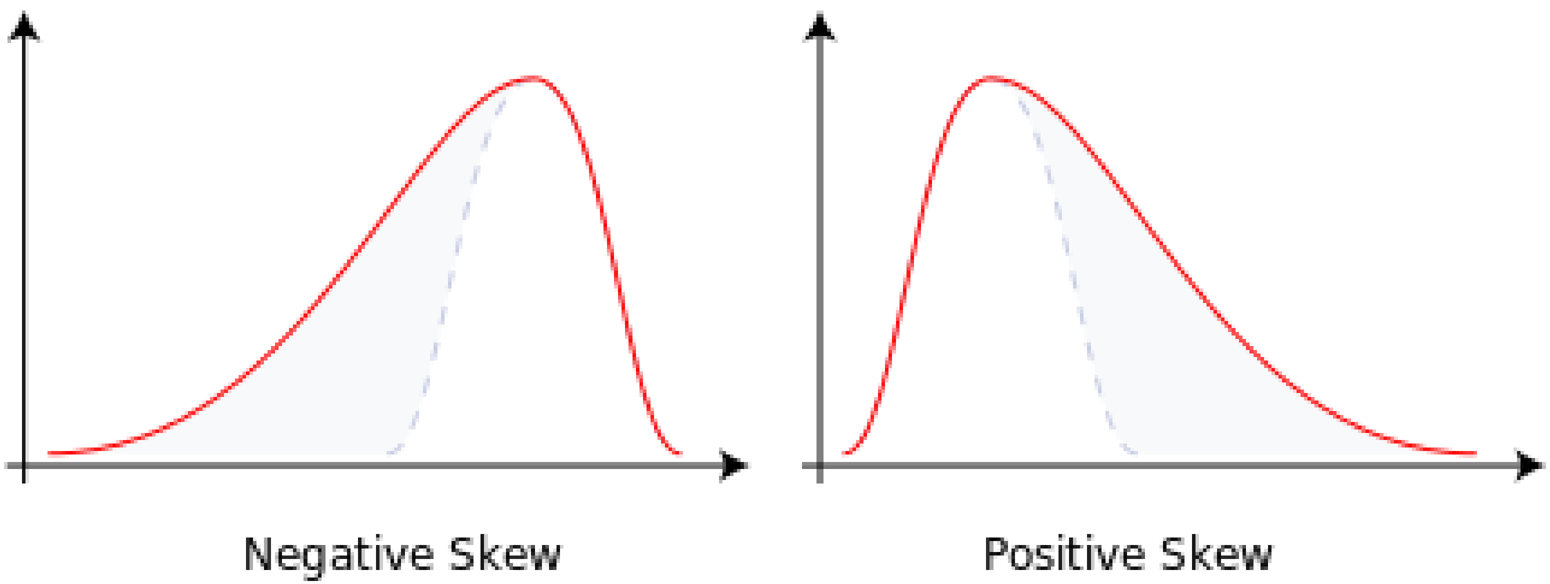
- There is also the **`summary(.)`** function that computes the mean, median, minimum, maximum, Q1, and Q3 quartiles of a vector or column

Statistical Measures

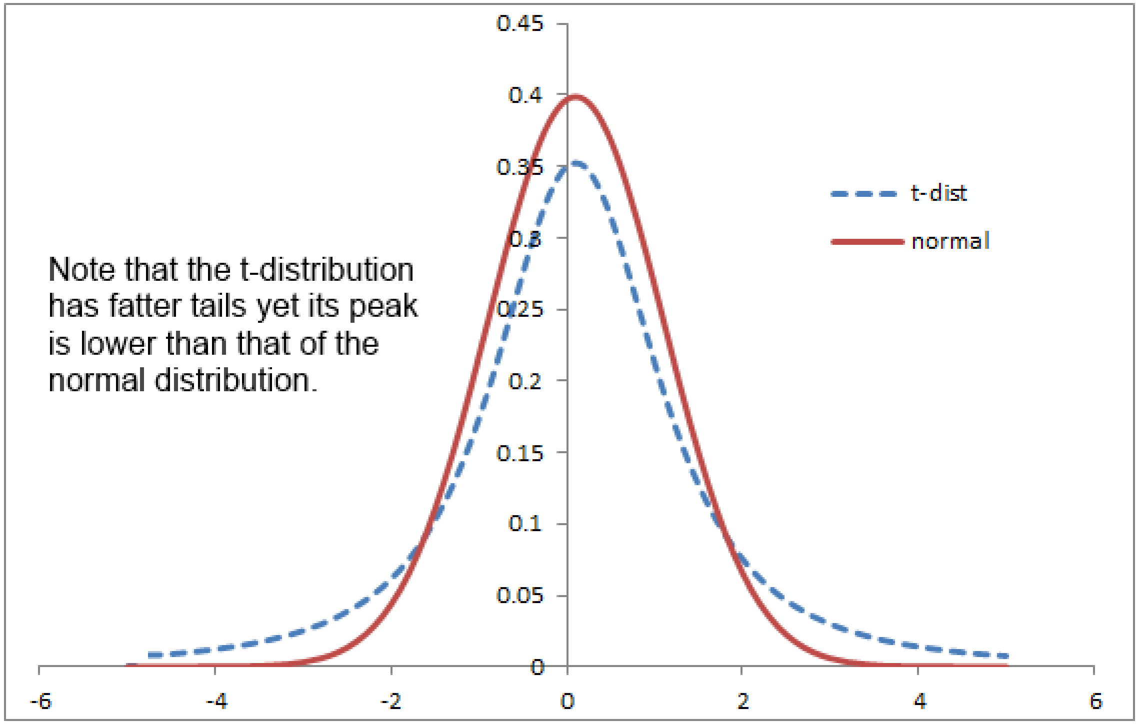
- Skewness is the measure of symmetry of the distribution
  - If the mean of data values is less than the median, then the distribution is said to be left-skewed
  - If the mean of the data values is greater than the median, then the distribution is said to be right-skewed
  - Based on the 3rd moment of the distribution

- Kurtosis measures whether the data is heavy-tailed or light-tailed relative to a normal distribution
  - Datasets with high kurtosis tend to have heavy tails, and/or outliers
  - Datasets with low kurtosis tend to have light tails, and fewer outliers
  - Based on the 4th moment of the distribution
  - Important for measuring risk in negative returns (left tail)

- For higher moments, we need to install an external package
- There are multiple packages available
- The book uses the **`e1071`** package (install this)
  - **`skewness(.)`**
  - **`kurtosis(.)`**
  - **`moments(.)`**
- Skewness – visually (http://www.deeplytrivial.com/2017/05/statistics-sunday-whats-normal-anyway.html):



- Kurtosis – visually (cannot be attributed – dead link):



- Remarks:
  - The kurtosis of any normal distribution = 3
  - In practice, this value is typically scaled to 0, so that other distributions are measured with respect to the normal for kurtosis

Correlation, Covariance, and Linear Regression

Correlation, Covariance, and Linear Regression

- Correlation and Covariance: Critical for fund management, portfolio strategies, and risk management, among other quant finance models
- Regression models: Also a critical weapon in a quant's arsenal
  - Fund tracking
  - Loan and credit trends and risk
  - Detecting correlations in data science
- In Base R:

```
cor(.)      # for correlations
cov(.)      # for covariances
cov2cor(.)  # scales a covariance matrix into the corresponding
            # correlation matrix efficiently
lm(.)       # for linear regression (lm = linear model)
```

- Remark: We will use the Base R functions for correlation and covariance, rather than **`rcorr(.)`** in the text

- Correlation and Covariance Examples:

```
# Open/High/Low/Close Prices in columns 2 to 5:
prices <- Sampledata[, 2:5]

# Covariance Matrix:
covMtx <- cov(prices)

# Correlation Matrix:
corrMtx <- cor(prices)

# Alternatively
corrFromCov <- cov2cor(covMtx)
```

```
> covMtx
          Open     High      Low    Close
Open  56.46133 50.62493 51.31672 45.31995
High  50.62493 49.04239 48.77378 45.45307
Low   51.31672 48.77378 53.44545 48.20213
Close 45.31995 45.45307 48.20213 47.12941
```

```
> corrMtx
           Open      High       Low     Close
Open  1.0000000 0.9620620 0.9341744 0.8785529
High  0.9620620 1.0000000 0.9526760 0.9454344
Low   0.9341744 0.9526760 1.0000000 0.9604281
Close 0.8785529 0.9454344 0.9604281 1.0000000
```

```
> corrFromCov
           Open      High       Low     Close
Open  1.0000000 0.9620620 0.9341744 0.8785529
High  0.9620620 1.0000000 0.9526760 0.9454344
Low   0.9341744 0.9526760 1.0000000 0.9604281
Close 0.8785529 0.9454344 0.9604281 1.0000000
```

- Simple Linear Regression example: Predict closing price from Volume:

```
# Simple Linear Regression - Regress daily volume
# on daily closing price:
Y <- Sampledata$Adj.Close
X <- Sampledata$Volume

fit <- lm(Y ~ X)
summary(fit)
```

- Results of summary:

```
Call:
lm(formula = Y ~ X)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.3053  -5.1630  -0.4186   5.9110  14.2786 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.945e+02  2.427e+00  80.146   <2e-16 ***
X           -1.880e-07  5.057e-07  -0.372    0.712    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.926 on 48 degrees of freedom
Multiple R-squared:  0.002871,	Adjusted R-squared:  -0.0179 
F-statistic: 0.1382 on 1 and 48 DF,  p-value: 0.7117
```

- Note that the fit object also holds member data that can be accessed:

```
names(fit)     # fit is an lm object
fit$coefficients
head(fit$fitted.values)    # predicted Y values
```

- Results:

```
> names(fit)     # fit is an lm object
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values"
 [6] "assign"        "qr"            "df.residual"   "xlevels"       "call"
[11] "terms"         "model"
>
> fit$coefficients
  (Intercept)             X
 1.945343e+02 -1.879999e-07
>
> head(fit$fitted.values)    # predicted Y values
       1        2        3        4        5        6
193.7552 193.2529 194.4186 194.0231 193.9351 193.5114
```

- Multiple Regression Example
- Regress High/Volume/Price on Daily Adj Close price

```
# Multiple Regression example:
fitbit <- lm(Adj.Close ~ Open + Volume + Return, data = Sampledata)
summary(fitbit)
```

- Results:

```
> summary(fitbit)

Call:
lm(formula = Adj.Close ~ Open + Volume + Return, data = Sampledata)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.1584 -1.3386 -0.0109  0.9714  7.3439 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.445e+01  7.785e+00   3.140  0.00295 ** 
Open         8.767e-01  3.999e-02  21.920  < 2e-16 ***
Volume      -2.161e-07  1.506e-07  -1.434  0.15820    
Return       1.267e+02  1.469e+01   8.628 3.58e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.057 on 46 degrees of freedom
Multiple R-squared:  0.9157,	Adjusted R-squared:  0.9102 
F-statistic: 166.6 on 3 and 46 DF,  p-value: < 2.2e-16
```

- Put the predictors with the estimated values of the response:

```
# Put data together with predictions in a separate dataframe:
predictPlusData <- data.frame(Sampledata$Date, fitbit$fitted.values, Sampledata$Open,
                              Sampledata$Volume, Sampledata$Return)

colnames(predictPlusData) <- c("Date", "Predict", "Open", "Volume", "Return")
head(predictPlusData)
```

- Results:

```
head(predictPlusData)
        Date  Predict   Open  Volume Return
12/14/2016 197.7829 198.74 4144600   0.00
12/13/2016 196.1331 193.18 6816100   0.03
12/12/2016 193.3380 192.80  615800   0.00
 12/9/2016 191.1914 190.87 2719600   0.00
 12/8/2016 192.1248 192.05 3187300   0.00
 12/7/2016 191.5344 186.15 5441400   0.04
```

Standardization

- It is a common task with financial and time series data to either de-mean, or standardize the values
  - De-mean: subtract the mean of a data column from each element, thus centering it about zero
  - Standardize: de-mean the data, as well as divide by the standard deviation so as to compare all data columns on a consistent scale
- In R, we can use the **`scale(.)`** function to perform each task:

```r
## De-meaning and Centralizing Data
volumes <- Sampledata$Volume

# De-mean
nomean <- scale(x = volumes, center = TRUE, scale = FALSE)

# Standardise
stand <- scale(x = volumes, center = TRUE, scale = TRUE)

# Remark:  Using the defaults, we could rewrite the
# standardisation more simply as
stand <- scale(volumes)
```

Standardization and Normalization

- Remark: The section following Standardization in the text is called Normalization
- Per the book, it has a different definition, and we will hold it in abeyance for now
- However, in practice, statisticians and quants also use the term *normalization* to mean what the book calls *standardization*

[END]