Package	**Function**	**Note**	**Example**	**File**
Build-in packages in R (stats, unit, base packages), no need to install and library packages	`head(x, n, ...)`	Extract the first n rows of data/matrix	`head(cars)`	L1B1
	`tail(x, n, ...)`	Extract the last n rows of data/matrix	`tail(cars)`	L1B1
	`diff(x, lag = 1, differences = 1, ...)`	Returns suitably lagged and iterated differences	`diff(x = AAPL$AAPL.Adjusted,differences = 1)`	L2B1
	`install.packages(.)`	Install packages	`install.packages("readxl")`	L1B1
	`library(.)`	prepare functions in packages	`library("readxl")`	L1B1
	`Help(.) or ?.`	Such for usage document for functions	`help("read_excel");? read_excel`	L1B1
	`read.csv(path, ...)`	Read csv file	`read.csv("AAPL.csv")`	L1B1
	`write.csv(x, file = "", ...)`	Save data to csv file	`write.csv(data_AAPL,"data.csv",row.names = TRUE)`	L1B1
	`read.table(path, header = TRUE, sep =  ""...)`	Read txt file	`read.table("AAPL.txt", header = TRUE, sep=" ")`	L1B1
	`write.table(x, file = "", ...)`	Save data to txt file	`write.table(data_AAPL, "data.txt",sep = " ", dec = ".", row.names = TRUE, col.names = TRUE)`	L1B1
	`data.frame(.)`	Create data frame	`data.frame(BJsales)`	L1B1
	`na.omit(.)`	Remove NA value from data		
	`na.approx(.)`	Create data frame	`na.approx(DEXUSEU)`	L2B3
	`merge(x, y, ...)`	Merge two data frames by common columns or row names, or do other versions of database *join* operations	`merge(AAPL$AAPL.Adjusted, CPIAUCSL, join='left')`	L2B2
	`hist(.)`	computes a histogram of the given data values and plot.	`hist(logreturn_AAPL)`	L1B1
	`summary(.)`	summary is a generic function used to produce result summaries of the results of various model fitting functions.	``summary(return_differences$`5day Return differences`)``	L2B1

	`rank(x, na.last = TRUE, ties.method)`	Returns the sample ranks of the values in a vector.	`# No. 1 is the biggest return` `rank(-as.numeric(AAPL$Return),na.last = "keep")`	L4B1
	`quantile(x, probs = seq(0, 1), na.rm = FALSE, names = TRUE, ...)`	Produces sample quantiles corresponding to the given probabilities.	`# calcaulte 80th percentage quantile return` `Quantile80 = quantile(AAPL$Return,probs=0.8, na.rm=TRUE)`	L4B1
	`which.min(x)` `which.max(x)`	Determines the index of the (first) minimum or maximum of a numeric (or logical) vector.	`# data index with closest return to Quantile80` `index80 = which.min(abs(Quantile80-AAPL$Return))`	L4B1
	`get(.)`	Search by name for an object.	`get("GE") or get(GE)`	L5B1
	`lapply(X, FUN, ...)`	Returns a list of the same length as `X`, each element of which is the result of applying `FUN` to the corresponding element of `X`.	`lapply(tickers, function(x) Ad(get(x)))`	L5B1
	`do.call(what, args)`	Constructs and executes a function and a list of arguments to be passed to it.	`do.call(merge, lapply(tickers, function(x) Ad(get(x))))`	L5B1
	`subset(x, subset, ...)`	Return subsets of vectors, matrices or data frames which meet conditions.	`subset(Asset_df, Date>="2012-01-01" & Date<"2013-01-01")`	L5B1
	`var(x, y = NULL, ...)` `cov(x, y = NULL, ...)` `cor(x, y = NULL, ...)`	Compute the variance of x and the covariance or correlation of x and y if these are vectors.	`cor(Return)`	L5B1
	`colSums(x, ...)` `rowSums(x, ...)` `colMeans(x, ...)` `rowMeans(x, ...)`	Form row and column sums and means for numeric arrays (or data frames).	`rowSums(Equityv)`	L5B2
	`cumsum(x)` `cumprod(x)` `cummax(x)` `cummin(x)`	Returns cumulative sums, products, minima or maxima of the elements of the argument.	`cumsum(Port$Port.Return)` `cummax(PortReturnCumu)`	L5B2
	`dnorm(x,mean,sd, ...)` `pnorm(q,mean,sd, lower.tail=TRUE, ...)` `qnorm(p, mean,sd, lower.tail=TRUE, ...)` `rnorm(n,mean,sd, ...)`	Density, distribution function, quantile function and random generation for the normal distribution.	`pnorm(x,mean = mu, sd = sigma)`	L5B3

	`dt(x, df, ncp)`
	`pt(q, df,ncp,`
	` lower.tail=TRUE)`
	`qt(p, df,ncp,`
	` lower.tail=TRUE)`
	`rt(n, df, ncp)`
		Density, distribution function, quantile function and random generation for the t distribution with df degrees of freedom (and optional non-centrality parameter ncp).	`pvalue = pt(q=(tstat),df = reg$df,lower.tail = TRU`
			`E)`
				L8B1

	`set.seed(.)`	Set random seed so that we can replicate the result	`set.seed(567)`	L6B1

	`sample(x, size,`
	` replace = FALSE,`
	` prob = NULL)`
		sample takes a sample of the specified size from the elements of `x` using either with or without replacement.	`sample(NormalRN$SimuReturn, size = N,replace=FALSE)`	L6B1

	`lm(formula, data, ...)`	`lm` is used to fit linear models	`lm(UMCSENT~1+UNRATE, data = Train)`	L7B1
	`predict(model,newdata)`	predictions from the results of various model fitting functions.	`predict(reg1, new_data)`	L7B1
	`decompose(x, type, ...)`	Decompose a time series into seasonal, trend and irregular components using moving averages.	`decompose(HSNG)`	L9B0

	`acf(x, ...)`
	`pacf(x, ...)`
		`acf` is for autocovariance or autocorrelation. `pacf` is for partial autocorrelations.	`acf(RGDP$FirstDifLog, main='')`
			`pacf(RGDP$FirstDifLog, main='')`
				L12B2

	`arima(x,`
	`order=c(0, 0, 0), ...)`
		Fit an ARIMA model to a univariate time series.	`arima(RGDP$FirstDifLog, order = c(1, 0, 0),`
			`seasonal=c(0, 0, 0))`
				L12B2

	`qqnorm(y, ...)`	Produces a normal QQ plot of the values in y.	`qqnorm(residuals(opt_fit))`	L12B2
	`qqline(y, ...)`	Adds a line to a "theoretical", by default normal, quantile-quantile plot	`qqline(residuals(opt_fit))`	L12B2
	`ks.test(x, y, ...)`	Perform a Kolmogorov-Smirnov test.	`ks.test(residuals(opt_fit),pnorm)`	L12B2
readxl	`read_excel(path, ...)`	Read excel file	`read_excel("AAPL.xlsx")`	L1B1

ggplot2	`ggplot(data,`
	`aes(x,y,group, ...)+`
	`geom_line()+ ...`
		Create elegant data visualizations		

quantmod	`getSymbols(Symbols,`
	` from,to,`
	` src = "yahoo", ...)`
		Load data via API.
		Note: argument from and to do not work when `src ="FRED"`
			`getSymbols("GE",from='2017-01-01',to="2021-03-01")`
			`getSymbols('CPIAUCSL', src='FRED')`
				L1B1
				
				L2B2

	`Op(x)`
	`Hi(x)`
	`Lo(x)`
	`Cl(x)`
	`Vo(x)`
	`Ad(x)`
		Extract (transformed) data from a suitable OHLC object.	`Ad(AAPL) # return adjusted closed price of AAPL`	

	`periodReturn(x,`
	` period='monthly',`
	` type='arithmetic',...)`
		Given a set of prices, return periodic returns.	`periodReturn(AAPL,period='daily',type='arithmetic')`	

Quandl	`Quandl.api_key(.)`	Set API key	`Quandl.api_key("abcd")`	L3B4

	`Quandl(code,`
	` start_date, end_date,`
	` type, transform,`
	` collapse,order, ...)`
		Retrieves Data from the Quandl Dataset endpoint and formats	`Quandl(code = c('WIKI/GOOGL.11','WIKI/IBM.11'),`
			`start_date = "2011-12-30",end_date = "2013-01-01",`
			`collapse="daily", type="xts")`
				L3B4

PerformanceAnalytics	`chart_Series(.)`	Plot time series data	`chart_Series(AAPL$AAPL.Close)`	L1B1
	`chart.Histogram(.)`	Create a histogram of returns, with optional curve fits for density and normal.	`chart.Histogram(AAPL$Return, main = "Plain")`	L4B1

	`chart.Drawdown(.)`	A time series chart demonstrating drawdowns from peak equity attained through time, calculated from periodic returns.	`chart.Drawdown(Return_AAPL,`
			`    legend.loc = "bottomleft", ylab = "Drawdowns",`
			`    date.format = "%b/%Y",main = "Drawdown Plot",`
			`    las = 2,color="red")`
				L2B1

	`chart.Boxplot(.)`	A wrapper to create box and whiskers plot with some defaults useful for comparing distributions.	`chart.Boxplot(AAPL$Return,`
			`    outlier.symbol="*",`
			`    symbol.color =c("darkblue"),outcol="red")`
				L4B2

	`charts.PerformanceSummary(.)`	For a set of returns, create a wealth index chart, bars for per-period performance, and underwater chart for drawdown	`charts.PerformanceSummary(AAPL$Return)`	L4B1
	`chart.Correlation(.)`	Visualization of a Correlation Matrix.	`chart.Correlation(Return)`	L5B1

	`chart.VaRSensitivity(`
	`  R, methods, ...)`
			`chart.VaRSensitivity(Portfolio.R$BuyHold,`
			`          methods=c("HistoricalVaR","GaussianVaR"))`
				L14B1

	`table.Stats(.)`	Returns a basic set of statistics that match the period of the data passed in.	`table.Stats(AAPL$Return)`	L4B1

	`table.Distributions(.)`	Table of distribution stats.	`table.Distributions(AAPL$Return)`	L4B1
	`Return.calculate( prices, method = c("log", "discrete", "difference"))`	calculate simple or compound returns from prices	`Return.calculate(AAPL$AAPL.Adjusted, method ="discrete")`	L4B1
	`apply.rolling(R, width, gap = 12, by = 1, FUN = "mean", ...)`	Creates a results timeseries of a function applied over a rolling window.	`width_n = 250 # window length = 1 year` `by_n = 1 # update frequency = everyday` `apply.rolling(AAPL$Return, width = width_n, by = by_n,FUN="sum")`	L4B3
	`VaR(R, p = 0.95, Method,...)`	Calculates Value-at-Risk(VaR) for univariate, component, and marginal cases using a variety of analytical methods.	`VaR(Portfolio.R, p=.95, method="historical")`	L14B1
zoo	`runSum(x, n = 10, cumulative = FALSE)`	Calculate data sums over a n-period moving window	`runSum(x=Return_AAPL, n = 5, cumulative = FALSE)`	L2B1
xts	`as.xts(x, ...)`	Covert data to xts data		
	`endpoints(x, on="months", k=1)`	Extract index values of a given xts object corresponding to the *last* observations given a period specified by on	`endpoints(R,on = Rebalance_freq ,k=k)`	L14B1
	`to.period(x,period, period = c("days", "weeks", "months", "quarters", "years"), OHLC=TRUE)`	Convert an OHLC or univariate object to a specified periodicity lower than the given data object.	`to.period(port, period= "months")`	L3B2
Histogram Tools	`PlotRelativeFrequency(x , ylab="Relative Frequency", ...)`	Produces a relative frequency histogram. `x` is "histogram" object (created by hist) .	`PlotRelativeFrequency( hist(na.omit(AAPL$Return),breaks = 40,plot=FALSE), xlab="Return", main="Relative Frequency Histogram")`	L4B1
corrplot	`corrplot(corr, method,...)`	A graphical display of a correlation matrix, confidence interval.	`corrplot(ReturnCorM, method="circle")`	L5B1
roll	`roll_cor(x, y = NULL, width, ...)`	Computes the rolling and expanding correlations of time-series data.	`roll_cor(x=Return$AAPL.Return, y = Return$GSPC.Return, width=20)`	L5B1
fitdistrplus	`descdist(data, ...)`	Computes descriptive parameters of an empirical	`descdist(as.numeric(AAPL$Return),obs.col = colors)`	L5B3

		distribution and provides a skewness-kurtosis plot.		
DistributionUtils	`skewness(.)` `kurtosis(.)`	Computes the sample skewness and sample kurtosis.		
rollRegres	`roll_regres( formula, data, width, do_compute, ...)`	Method for fast rolling and expanding regression models.	`roll_regres(data = "Returns",width = 24L, formula = ExcessReturn.Stock~1+ExcessReturn.Market,do_compute = c("sigmas", "r.squareds")`	L8B2
TTR	`SMA(x, n = 10, ...)`	Calculate various moving averages (MA) of a series.	`SMA(HSNG,n=24)`	L9B0
lubridate	`month(x, ...)` `year(x)` `quarter(x)`	Return month/quarter/year number	`month(index(Asset))`	L9B1
car	`linearHypothesis(model, hypothesis.matrix,...)`	Generic function for testing a linear hypothesis.	`linearHypothesis(model2, c("betasLogFactorRetMkt.RF=0", "betasLogFactorRetSMB=0", "betasLogFactorRetHML=0"))`	L10B2
lmtest	`coeftest(x, vcov. = NULL, df = NULL, ...)`	A generic function for performing z and (quasi-)t Wald tests of estimated coefficients.	`# t-test for coefficients. Argument of vcovHC accommodates for potential heteroscedasticity.` `coeftest(model2,vcov=vcovHC(model2))`	L10B2
tseries	`adf.test(x, alternative = c("stationary", "explosive"), ...)`	Computes the Augmented Dickey-Fuller test for the null that x has a unit root.	`adf.test(RGDP$Level, alternative = "stationary")`	L12B2
forecast	`auto.arima(y, d, D, stationary = FALSE, seasonal = TRUE,...)`	Returns best ARIMA model according to either AIC, AICc or BIC value.	`auto.arima(RGDP$LogLevel, seasonal=FALSE)`	L12B2
	`tsdisplay(x, lag.max,...)`	Plots a time series along with its acf and either its pacf, lagged scatterplot or spectrum.	`tsdisplay(residuals(opt_fit), lag.max=45, main='Optimal Model Residuals')`	L12B2
	`Arima(y,order,seasonal, include.mean = TRUE, include.drift = FALSE, include.constant, ...)`	Largely a wrapper for `arima` function in the stats package. The main difference is that this function allows a drift term.	`Arima(Train$Level, order=c(2,1,0), seasonal = FALSE, include.drift = TRUE)`	L12B3
PortfolioAnalytics	`portfolio.spec(.)` `add.constraint(.)` `add.objective(.)` `optimize.portfolio(.)`	Search for efficient portfolio with respect to constraints and objectives.		L13B0

	chart.Weights(.)	Charts the optimal weights of a portfolio.	chart.Weights(opt.portf, neighbors = NULL,main = "Weights")	L13B1
	chart.RiskReward(.)	This function charts risk-return	chart.RiskReward(opt.portf, risk.col="StdDev", return.col="mean",chart.assets = TRUE)	L13B1
	extractWeights(.)	Extract assets weights	Weights = extractWeights(opt.portf)	L13B1

bizdays	create.calendar(
	  name, holidays,
	  weekdays, ...)
		Creates calendars and stores them in the calendar register.	cal = create.calendar("America",
			      holidays=holidayNYSE(2000:2022),
			      weekdays=c("saturday", "sunday"))	

	adjust.next(dates,cal)
	following(.)
	adjust.none(.)
	modified.following(.)
	adjust.previous(.)
	preceding(.)
	modified.preceding(.)
		Rolls the given date to the next or previous business day, unless it is a business day.	modified.preceding(Rebalance.Date,cal)	L14B1

riskParityPortfolio	riskParityPortfolio(
	  Sigma, ...)
		Designs risk parity portfolios to equalize/distribute the risk contributions of the different assets.	Sigma<-cov(Return)
			portfolio.parity <- riskParityPortfolio(Sigma)
				L14B2

fPortfolio	efficientPortfolio(.)
	maxratioPortfolio(.)
	tangencyPortfolio(.)
	minriskPortfolio(.)
	minvariancePortfolio(.)
	maxreturnPortfolio(.)
		Returns efficient portfolios.	portfolio.tangency <- tangencyPortfolio(
			                      as.timeSeries(Return),
			                      constraints = "LongOnly")
				L14B2

	getCov(.)
	getCovRiskBudgets(.)
	getWeights(.)
		Extractor functions to get information from portfolio	getWeights(portfolio.tangency)
			getCovRiskBudgets(portfolio.tangency)
				L14B2

	rollingWindows(x, period, by = "1m")	Returns a list of rolling window frames	rollingWindows(Return,period="3m",by="1m")	L14B2

Note: LmBn in the "File" column indicates Lesson m Breakout n. For example, "L14B2" is Lesson 14 breakout 2 (see "BreakoutContents.pdf" for details)