Homework 6

Ze Yang (zey@andrew.cmu.edu)

Due Thursday, October 12 at 3:00 PM

You should submit the Rmd file for your analysis. Name the file as `YOURANDREWID_HW6.Rmd` and submit it via Canvas. Also submit the `.pdf` file that is produced.

Part 1:

Suppose we are interested in building a training set of features that we hope could be useful in a prediction problem. In particular, the idea we want to explore is the use of the percentiles of the daily closing stock price over the past n weeks.

Write R code to perform this task.

Use smoothing to estimate the percentiles. You can use the code that was provided and described in lecture.

The sample should be constructed by randomly sampling NYSE ticker symbols. An R package/function for obtaining the list of all NYSE ticker symbols was described in lecture.

Market data can be read in using the `quantmod` package. Again, this was presented in lecture.

Your code should assume that n is a positive integer, and $n \geq 2$. The following example illustrates how easy `quantmod` makes it to get the n most recent weeks of data:

```
foo = getSymbols("AAPL", from=Sys.Date()-n*7, to=Sys.Date(),
                 auto.assign=FALSE)
```

Choose the bandwidth for the kernel density estimators using the Sheather-Jones approach.

Use the "Adjusted" price to calculate the quantiles.

Sample 100 NYSE ticker symbols at random for this purpose.

The final result should be a data frame. Each row should correspond to a different ticker symbol. The first column should be filled with the ticker symbol. The following columns should be the percentiles. For this exercise, estimate the 10^{th}, 20^{th}, ..., 90^{th} percentiles. Hence, the data frame should have 100 rows and ten columns.

Solution:

```
stock.sample = function(tickers, n_weeks,
                        n_samples=100, seed=42, ...) {
  #' Download adjusted close prices for sample tickers.
  #' @param tickers: vector, the whole tickers set.
  #' @param n_weeks: int, number of weeks to look at.
  #' @param n_samples: int, number of tickers to draw.
  #' @param seed: int, the random seed.
  #' @return data.frame of prices, where each row is a date
```

```r
  #' and each column is a ticker.
  require(quantmod)
  set.seed(seed)
  tickers.shuffle = sample(tickers)
  # i: current ticker, i.got: number of good tickers.
  # (good: not 404 error, more than 1 rows)
  i = 1; i.got = 1
  price.list = list()
  downloaded.tickers = c()
  while (i.got <= n_samples && i <= length(tickers)) {
    out = tryCatch({
      # fetch stock data.
      df = getSymbols(
        tickers.shuffle[i],
        from=Sys.Date()-n_weeks*7, to=Sys.Date(),
        auto.assign=FALSE)
      # raise error(continue)
      # if the downloaded column has only 1 row.
      if (length(df[,1]) == 1) {
        stop("Abnormal data: Only one row.\n")
      }
      # increment i.got only if getting a "good" ticker.
      i.got = i.got + 1
      downloaded.tickers = c(
        downloaded.tickers, tickers.shuffle[i])
      Ad(df)
    },
    # error handling: continue
    error = function(e){message(e)},
    # warning handling: continue
    warning = function(e){message(e)})
    price.list[[tickers.shuffle[i]]] = out
    # increment i
    i = i+1
    # print progress
    if (i.got %% 10 == 0) {
      print(sprintf(
        "[quantmod] %d tickers downloaded.", i.got))
    }
  }
  return(price.list)
}

kde.cdf = function(x, bw='SJ', res=100, ...) {
  #'Get the cdf and inverse cdf of a kde distribution.
  #'@param x: vector, the data set over which the kde is computed.
```

```r
  #'@param bw: string, the method used to compute kde binwidth.
  #'@return List of two functions, the linear interpolated
  #'cdf and inverse cdf of the kernel density estimation.
  kde = density(x, bw=bw, ...)
  interp.kde = approxfun(
    kde$x, kde$y, yleft=0, yright=0)
  lb = min(kde$x)
  ub = max(kde$x)
  x.grid = seq(lb, ub, length=res)
  cdf.grid = numeric(res)
  for (i in 1:res) {
    cdf.grid[i] = integrate(
      interp.kde, lower=lb, upper=x.grid[i],
      stop.on.error=F)$value
  }
  cdf.fun = approxfun(
    x.grid, cdf.grid, yleft=0, yright=1)
  invcdf.fun = approxfun(
    cdf.grid, x.grid, yleft=NA, yright=NA)
  return(list(cdf=cdf.fun, inv.cdf=invcdf.fun))
}

stock.kde.quantiles = function(
  tickers, probs=c(0, 0.25, 0.5, 0.75, 1), n_weeks=2,
  n_samples=100, bw='SJ', res=100, seed=42) {
  #' Wrapper function of stock.sample and kde.cdf.
  #' @param tickers: vector, the whole tickers set.
  #' @param probs: vector, the prob level of quantiles to calculate.
  #' @param n_weeks: int, number of weeks to look at.
  #' @param n_samples: int, number of tickers to draw.
  #' @return data.frame, where each row is a ticker,
  #' and each column is a probability level.
  for (p in probs) {
    if (p > 1 || p < 0) {stop("Invalid probability.")}
  }
  quantiles.list = list()
  prices = stock.sample(tickers, n_weeks, n_samples, seed)
  ticker.sample = names(prices)
  for (t in ticker.sample) {
    cdf.holdout = kde.cdf(prices[[t]], bw, res)
    inv.cdf = cdf.holdout$inv.cdf
    quantiles.list[[t]] = inv.cdf(probs)
  }
  df = t(data.frame(quantiles.list))
  colnames(df) = paste(probs, '.quantile', sep='')
  return(df)
```

```r
}
```

```r
# Get all NYSE symbols
tickers = stockSymbols()
```

```
## Fetching AMEX symbols...

## Fetching NASDAQ symbols...

## Fetching NYSE symbols...
```

```r
NYSE = tickers[tickers$Exchange=='NYSE','Symbol']
probs = seq(0, 0.9, len=10)

qtl.df = stock.kde.quantiles(NYSE, probs, n_weeks=5)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).

## [1] "[quantmod] 10 tickers downloaded."
## [1] "[quantmod] 20 tickers downloaded."
## [1] "[quantmod] 30 tickers downloaded."
## [1] "[quantmod] 40 tickers downloaded."
## [1] "[quantmod] 50 tickers downloaded."
## [1] "[quantmod] 50 tickers downloaded."
## [1] "[quantmod] 60 tickers downloaded."
## [1] "[quantmod] 60 tickers downloaded."
## [1] "[quantmod] 70 tickers downloaded."
## [1] "[quantmod] 80 tickers downloaded."
## [1] "[quantmod] 90 tickers downloaded."
## [1] "[quantmod] 100 tickers downloaded."
```

```r
str(qtl.df)
```

```
##  num [1:100, 1:10] 55.76 31.8 8.81 202.93 1.18 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:100] "UAL" "VET" "KMM" "PSA" ...
```

```
##   ..$ : chr [1:10] "0.quantile" "0.1.quantile" "0.2.quantile" "0.3.quantile" ...
```

```r
head(qtl.df)
```

```
##      0.quantile 0.1.quantile 0.2.quantile 0.3.quantile 0.4.quantile
## UAL   55.763736    58.005082    58.602037    59.225309    59.878343
## VET   31.803004    33.563284    34.132169    34.709029    35.237450
## KMM    8.807154     8.854180     8.866622     8.879266     8.892561
## PSA  202.925763   207.414648   211.403864   212.380443   213.006287
## BORN   1.180447     1.225050     1.236732     1.245624     1.253175
## PEI    9.160610     9.893264    10.069217    10.204792    10.323313
##      0.5.quantile 0.6.quantile 0.7.quantile 0.8.quantile 0.9.quantile
## UAL     60.407620    60.846918    61.285870    61.947254    64.320858
## VET     35.712801    36.135758    36.485460    36.804384    37.167496
## KMM      8.902661     8.910885     8.918732     8.927363     8.939206
## PSA    213.483215   213.868307   214.239467   214.698844   215.532950
## BORN     1.260097     1.266832     1.273762     1.281563     1.292697
## PEI     10.433909    10.543228    10.659183    10.796313    10.997004
```

Part 2:

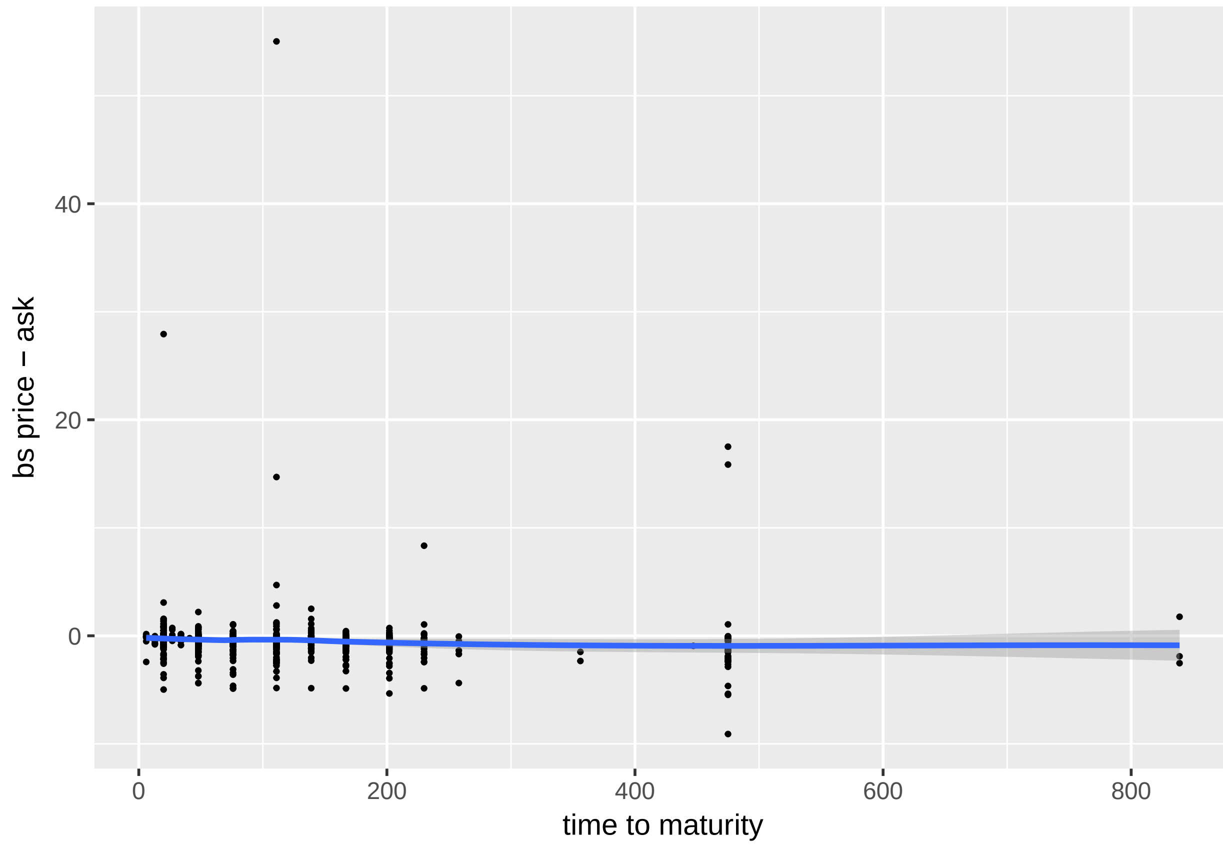
Read in the options sample that I presented in lecture. **(Read in the sample I created, do not generate your own.)**

Create a plot that compares (1) the difference between the Black-Scholes price and the ask price with (2) the time to expiration. Smooth the relationship using local linear regression, and show this on the plot. Is there evidence of a relationship between these two quantities? Can you guess as to why you are seeing this?

Solution:

```r
# read option sample
options.data = read.table("optionssample09302017.txt", sep=',', header=T)
# difference between Black-Scholes price and market price.
options.data$price.diff = options.data$bsval  - options.data$ask

ggplot(options.data, aes(x=timetoexpiry, y=price.diff)) +
  geom_point(size=0.5) + geom_smooth(
    method = "loess", method.args=list(degree=1)) +
  labs(x='time to maturity', y='bs price - ask')
```



Comments:

- There is no significant pattern between (1) the difference between Black-Scholes price and the market ask price and (2) the time to maturity of the option. The slope of tge fitted local linear regression line is almost zero, and the price differences are just scattered around the line very randomly.
- As a result, we may conclude that the Black-Scholes price matches with the real market price very well on the T (time to maturity) dimension.
- T is an input to Black-Scholes model, it is likely that the variations in option prices that attributes to T is well captured by the Black-Scholes model. The remaining errors are caused by other factors that are not included in BS model. So the errors have little correlation with T, and hence are identically distributed for different T's.