Audit Report for a renowned Retailer

Mughundhan Chandrasekar

3/20/2017

Question 1.

Compute the Trial Balance account entries from the datasets provided by management for LSI???s accounts at December 31, 2016 for the following six accounts: *Sales Revenue* Cost of Goods Sold *Unpaid Accounts Receivable*
Allowance for Doubtful Accounts *Inventory on hand at 12/31/2016

Solution - Work done using the following R code which is self-explanatory

Creating an appropriate environment

```
library(formattable)
library(scales)
library(lubridate)
library(pastecs)
setwd("/Users/Mughundhan/UIC/UIC Academics/SPRING 2017/AUDIT/Mid Term")
collections <- read.csv("collections.csv", header = T)
credit <- read.csv("credit.csv", header = T)
inventory <- read.csv("inventory.csv", header = T)
purchases <- read.csv("purchases.csv", header = T)
sales <- read.csv("sales.csv", header = T)
```

Sales Revenue

```
audityear = interval(ymd(20160101), ymd(20161231)) #Data inferred from Summary
sales$datep = as_date(sales$date) #Type Casting for computational ease
sales$year = year(sales$datep)
sales2016 = split(sales,sales$year)$'2016'
sales.audit.year = split(sales, sales$year)
summary(sales.audit.year)

##      Length Class      Mode
## 2015 11     data.frame list
## 2016 11     data.frame list
## 2017 11     data.frame list

head(sales2016) #Top 6 entries

##   X invoice  sku qty cashtrue       date unitprice   total cust.no
## 1 1       1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2 2       2  170  59        0 2016-01-12     22.09 1303.31     467
## 3 3       3 1449   5        1 2016-07-05     10.06   50.30     273
## 4 4       4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5 5       5  739  60        0 2016-11-15     31.64 1898.40     387
## 6 6       6  189  44        0 2016-03-28     24.81 1091.64     418
##        datep year
## 1 2016-01-23 2016
## 2 2016-01-12 2016
## 3 2016-07-05 2016
```

```
## 4 2016-03-26 2016
## 5 2016-11-15 2016
## 6 2016-03-28 2016
```

```
total_2016_Sales = sum(sales2016$total)
total_2016_Sales
```

```
## [1] 960030574
```

Inventory

```
inventoryCost = sum(inventory$unitcost*inventory$endstock)
inventoryCost
```

```
## [1] 152765109
```

Cost of Goods Sold

```
COGS1=merge(sales2016,inventory,by="sku")
head(COGS1)
```

```
##   sku    X.x invoice qty cashtrue       date unitprice.x total cust.no
## 1   1 505903  505903   4        1 2016-10-12         5.7  22.8     373
## 2   1 278696  278696 122        0 2016-08-02         5.7 695.4     606
## 3   1 962588  962588  12        0 2016-07-22         5.7  68.4     106
## 4   1 454907  454907   2        0 2016-05-24         5.7  11.4     882
## 5   1 688592  688592  39        0 2016-12-25         5.7 222.3     427
## 6   1 917373  917373 104        0 2016-06-18         5.7 592.8     527
##        datep year X.y unitcost unitprice.y beginstock endstock defective
## 1 2016-10-12 2016   1     3.73         5.7       6714    12175       100
## 2 2016-08-02 2016   1     3.73         5.7       6714    12175       100
## 3 2016-07-22 2016   1     3.73         5.7       6714    12175       100
## 4 2016-05-24 2016   1     3.73         5.7       6714    12175       100
## 5 2016-12-25 2016   1     3.73         5.7       6714    12175       100
## 6 2016-06-18 2016   1     3.73         5.7       6714    12175       100
##   returns
## 1      12
## 2      12
## 3      12
## 4      12
## 5      12
## 6      12
```

```
sum (COGS1$qty*COGS1$unitcost)
```

```
## [1] 350802594
```

Work done related to "Cash"

```
cash_Temp = sales[sales$cashtrue =='1',]
head(cash_Temp)
```

```
##     X invoice  sku qty cashtrue       date unitprice   total cust.no
## 3   3       3 1449   5        1 2016-07-05     10.06   50.30     273
## 13 13      13 1612   1        1 2017-01-12      6.80    6.80     255
## 16 16      16  589  49        1 2016-07-16     13.72  672.28     219
## 20 20      20 1848   9        1 2016-05-04     20.37  183.33     412
## 24 24      24 1875   1        1 2016-03-30     21.49   21.49     591
```

```
## 34 34      34  804  47        1 2016-09-27     29.83 1402.01     369
##         datep year
## 3  2016-07-05 2016
## 13 2017-01-12 2017
## 16 2016-07-16 2016
## 20 2016-05-04 2016
## 24 2016-03-30 2016
## 34 2016-09-27 2016

sum(cash_Temp$total)

## [1] 67299374
```

Generic computations related to purchases

```
audityear = interval(ymd(20160101), ymd(20161231))
purchases$datep = as_date(purchases$date)
purchases$year = year(purchases$datep)
purchases2016 = split(purchases,purchases$year)$'2016'
head(purchases2016)

##   X  sku unitcost quantity       date PO.no      datep year
## 1 1    1     3.73      976 2016-01-05  9343 2016-01-05 2016
## 2 2   10     1.88     3256 2016-01-05 19769 2016-01-05 2016
## 3 3  100     8.07     2210 2016-01-05  5419 2016-01-05 2016
## 4 4 1000     8.29     2983 2016-01-05 10812 2016-01-05 2016
## 5 5 1001     5.62     3491 2016-01-05 21359 2016-01-05 2016
## 6 6 1002     8.78     3406 2016-01-05  7490 2016-01-05 2016
```

Generic computations related to collections

```
audityear = interval(ymd(20160101), ymd(20161231))
collections$datep = as_date(collections$date)
collections$year = year(collections$datep)
collections2016 = split(collections,collections$year)$'2016'
head(collections2016)

##          X collected invoice       date receipt.no      datep year
## 7069   498    112.06     498 2016-01-01     286992 2016-01-01 2016
## 7070  8818   1167.67    8818 2016-01-01     239197 2016-01-01 2016
## 7071 17744    627.75   17744 2016-01-01     895122 2016-01-01 2016
## 7072 18042   1060.80   18042 2016-01-01     781039 2016-01-01 2016
## 7073 19573     74.58   19573 2016-01-01     524633 2016-01-01 2016
## 7074 23222    376.32   23222 2016-01-01     537192 2016-01-01 2016
```

Work done related to "Unpaid Accounts Receivable"

```
AccountsReceivable <- merge(sales2016,collections2016,by="invoice",all.x = TRUE)
head(accountsReceivable)

##   invoice X.x  sku qty cashtrue     date.x unitprice   total cust.no
## 1       1   1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2       2   2  170  59        0 2016-01-12     22.09 1303.31     467
## 3       3   3 1449   5        1 2016-07-05     10.06   50.30     273
## 4       4   4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5       5   5  739  60        0 2016-11-15     31.64 1898.40     387
## 6       6   6  189  44        0 2016-03-28     24.81 1091.64     418
##      datep.x year.x X.y collected     date.y receipt.no    datep.y year.y
```

```
## 1 2016-01-23   2016   1   2394.44 2016-05-21     496082 2016-05-21   2016
## 2 2016-01-12   2016   2   1303.31 2016-09-09     515154 2016-09-09   2016
## 3 2016-07-05   2016   3     50.30 2016-08-22     789087 2016-08-22   2016
## 4 2016-03-26   2016   4   1170.78 2016-06-17     590135 2016-06-17   2016
## 5 2016-11-15   2016   5   1898.40 2016-12-20     967507 2016-12-20   2016
## 6 2016-03-28   2016   6   1091.64 2016-10-15     624062 2016-10-15   2016
```

```
accountsReceivable$collected[is.na(accountsReceivable$collected)]<-0
accountsReceivable$collected[accountsReceivable$collected<0]<-0

cash_Collected = sum(accountsReceivable$collected)
cash_Collected
```

```
## [1] 524843737
```

```
cash_Total = sum(accountsReceivable$total)
cash_Total
```

```
## [1] 960030574
```

```
diff_Amt = cash_Total - cash_Collected
diff_Amt
```

```
## [1] 435186837
```

Work done related to "Allowance for Doubtful Accounts"

```
allowancetable = accountsReceivable
head(allowancetable)
```

```
##   invoice X.x  sku qty cashtrue     date.x unitprice   total cust.no
## 1       1   1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2       2   2  170  59        0 2016-01-12     22.09 1303.31     467
## 3       3   3 1449   5        1 2016-07-05     10.06   50.30     273
## 4       4   4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5       5   5  739  60        0 2016-11-15     31.64 1898.40     387
## 6       6   6  189  44        0 2016-03-28     24.81 1091.64     418
##      datep.x year.x X.y collected     date.y receipt.no    datep.y year.y
## 1 2016-01-23   2016   1   2394.44 2016-05-21     496082 2016-05-21   2016
## 2 2016-01-12   2016   2   1303.31 2016-09-09     515154 2016-09-09   2016
## 3 2016-07-05   2016   3     50.30 2016-08-22     789087 2016-08-22   2016
## 4 2016-03-26   2016   4   1170.78 2016-06-17     590135 2016-06-17   2016
## 5 2016-11-15   2016   5   1898.40 2016-12-20     967507 2016-12-20   2016
## 6 2016-03-28   2016   6   1091.64 2016-10-15     624062 2016-10-15   2016
```

```
allowancetable$ar<-(allowancetable$total-allowancetable$collected)
head(allowancetable)
```

```
##   invoice X.x  sku qty cashtrue     date.x unitprice   total cust.no
## 1       1   1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2       2   2  170  59        0 2016-01-12     22.09 1303.31     467
## 3       3   3 1449   5        1 2016-07-05     10.06   50.30     273
## 4       4   4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5       5   5  739  60        0 2016-11-15     31.64 1898.40     387
## 6       6   6  189  44        0 2016-03-28     24.81 1091.64     418
##      datep.x year.x X.y collected     date.y receipt.no    datep.y year.y
## 1 2016-01-23   2016   1   2394.44 2016-05-21     496082 2016-05-21   2016
## 2 2016-01-12   2016   2   1303.31 2016-09-09     515154 2016-09-09   2016
```

```
## 3 2016-07-05   2016   3     50.30 2016-08-22     789087 2016-08-22   2016
## 4 2016-03-26   2016   4   1170.78 2016-06-17     590135 2016-06-17   2016
## 5 2016-11-15   2016   5   1898.40 2016-12-20     967507 2016-12-20   2016
## 6 2016-03-28   2016   6   1091.64 2016-10-15     624062 2016-10-15   2016
##   ar
## 1  0
## 2  0
## 3  0
## 4  0
## 5  0
## 6  0
```

```
allowancetable$ardueperiod<-as.Date(as.character("2016-12-31"),format="%Y-%m-%d")-as.Date
(as.character(allowancetable$date.x),format="%Y-%m-%d")
allowancetableRefined <- allowancetable[allowancetable$ar>0,]
head(allowancetableRefined)
```

```
##    invoice X.x  sku qty cashtrue     date.x unitprice   total cust.no
## 13      17  17  666  56        0 2016-10-08     33.78 1891.68      72
## 14      18  18  456  45        0 2016-10-13     18.11  814.95     319
## 17      21  21 1703  19        0 2016-05-23     10.54  200.26     144
## 18      22  22  220 160        0 2016-09-19     31.21 4993.60     871
## 23      28  28  506 116        0 2016-07-05     14.30 1658.80     561
## 24      29  29 1403  15        0 2016-12-29     31.56  473.40     133
##       datep.x year.x X.y collected date.y receipt.no datep.y year.y
## 13 2016-10-08   2016  NA         0   <NA>         NA    <NA>     NA
## 14 2016-10-13   2016  NA         0   <NA>         NA    <NA>     NA
## 17 2016-05-23   2016  NA         0   <NA>         NA    <NA>     NA
## 18 2016-09-19   2016  NA         0   <NA>         NA    <NA>     NA
## 23 2016-07-05   2016  NA         0   <NA>         NA    <NA>     NA
## 24 2016-12-29   2016  NA         0   <NA>         NA    <NA>     NA
##         ar ardueperiod
## 13 1891.68     84 days
## 14  814.95     79 days
## 17  200.26    222 days
## 18 4993.60    103 days
## 23 1658.80    179 days
## 24  473.40      2 days
```

Since the execution of the conditional for loop takes a long time, the value was computed and printed as below. The code used to infer this solution is included.

```
for (i in 1:nrow(allowancetableRefined))
{
  if(allowancetableRefined$ardueperiod[i]>180){
    allowancetableRefined$allowanceamt[i] = 0.4*allowancetableRefined$ar[i]
  }else if(allowancetableRefined$ardueperiod[i]>=90 && allowancetableRefined$ardueperiod[
i]<=180){
    allowancetableRefined$allowanceamt[i] = 0.2*allowancetableRefined$ar[i]
  }else{
    allowancetableRefined$allowanceamt[i] = 0
  }
}

sum(allowancetableRefined$allowanceamt)
69,273,535
```

TRIAL BALANCE ENTRIES	CALCULATED VALUE	ORIGINAL VALUE	REPLACEMENT
Sales Revenue	$960,030,574	$969,139,000	Required
Inventory	$152,765,109	$152,765,000	Not Required
Cost of Goods Sold	$350,802,594	$353,803,000	Required
Accounts Receivable	$435,186,837	$432,418,000	Required
Allowance for Doubtful Accounts	$69,273,535	$73,458,000	Required

NOTE: Replacement is NOT REQUIRED when the difference is less than one million (Only Inventory satisfies this condition).

Question 2.

All of LSI???s transaction documents and journal entries are sequentially numbered with a unique identifier (Sales Invoice Number; Purchase Order Number; Cash Receipt Number, SKU). Perform the following audit program tests of the Revenue Cycle for all transactions during the 2016 fiscal year:

2.1. Foot (total) and agree to Trial Balance

```
sum(sales2016$total) #Gives the foot (total)
```

```
## [1] 960030574
```

Based on the value of foot(total), we shall disagree with the Trial Balance

2.2. Statistically summarize the transactions in the datasets This gives a quick and simple description of the data which includes mean, median, mode, minimum value, maximum value, range, standard deviation, etc.

```
summary(collections)
```

```
##        X             collected          invoice                date       
##  Min.   :      1   Min.   : -719.1   Min.   :      1   2016-12-13:  2885  
##  1st Qu.: 324816   1st Qu.:  175.8   1st Qu.: 324816   2016-12-26:  2865  
##  Median : 649489   Median :  520.0   Median : 649489   2016-12-05:  2848  
##  Mean   : 649746   Mean   :  876.1   Mean   : 649746   2016-12-08:  2847  
##  3rd Qu.: 974770   3rd Qu.: 1193.0   3rd Qu.: 974770   2016-12-22:  2841  
##  Max.   :1299998   Max.   :15002.3   Max.   :1299998   2016-12-10:  2840  
##                                                        (Other)   :679470  
##    receipt.no         datep                year        
##  Min.   :     2   Min.   :2015-11-26   Min.   :2015  
##  1st Qu.:249970   1st Qu.:2016-05-27   1st Qu.:2016  
##  Median :500309   Median :2016-08-19   Median :2016  
##  Mean   :499685   Mean   :2016-08-07   Mean   :2016  
```

```
##  3rd Qu.:748970   3rd Qu.:2016-10-28   3rd Qu.:2016
##  Max.   :999999   Max.   :2017-01-01   Max.   :2017
##
```

```
summary(credit)
```

```
##        X             customer          limit
##  Min.   :   1.0   Min.   :   1.0   Min.   :109000
##  1st Qu.: 250.8   1st Qu.: 250.8   1st Qu.:223750
##  Median : 500.5   Median : 500.5   Median :231000
##  Mean   : 500.5   Mean   : 500.5   Mean   :230393
##  3rd Qu.: 750.2   3rd Qu.: 750.2   3rd Qu.:238000
##  Max.   :1000.0   Max.   :1000.0   Max.   :261000
```

```
summary(inventory)
```

```
##        X               sku            unitcost        unitprice
##  Min.   :   1.0   Min.   :   1.0   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 500.8   1st Qu.: 500.8   1st Qu.: 3.940   1st Qu.: 9.838
##  Median :1000.5   Median :1000.5   Median : 5.965   Median :15.095
##  Mean   :1000.5   Mean   :1000.5   Mean   : 6.061   Mean   :16.572
##  3rd Qu.:1500.2   3rd Qu.:1500.2   3rd Qu.: 8.070   3rd Qu.:22.260
##  Max.   :2000.0   Max.   :2000.0   Max.   :15.710   Max.   :54.160
##    beginstock       endstock       defective         returns
##  Min.   : 5007   Min.   : 5002   Min.   :  53.0   Min.   :  7.0
##  1st Qu.: 8857   1st Qu.: 8719   1st Qu.: 154.8   1st Qu.: 25.0
##  Median :12576   Median :12602   Median : 225.0   Median : 41.5
##  Mean   :12543   Mean   :12530   Mean   : 313.4   Mean   : 61.8
##  3rd Qu.:16218   3rd Qu.:16305   3rd Qu.: 384.0   3rd Qu.: 74.0
##  Max.   :19996   Max.   :20000   Max.   :1813.0   Max.   :485.0
```

```
summary(purchases)
```

```
##        X              sku            unitcost        quantity
##  Min.   :    1   Min.   :   1.0   Min.   : 0.000   Min.   : 976
##  1st Qu.: 6001   1st Qu.: 500.8   1st Qu.: 3.940   1st Qu.:2518
##  Median :12000   Median :1000.5   Median : 5.965   Median :2884
##  Mean   :12000   Mean   :1000.5   Mean   : 6.061   Mean   :2887
##  3rd Qu.:18000   3rd Qu.:1500.2   3rd Qu.: 8.070   3rd Qu.:3268
##  Max.   :24000   Max.   :2000.0   Max.   :15.710   Max.   :4215
##
##          date           PO.no           datep                 year
##  2016-01-05: 2000   Min.   :    1   Min.   :2016-01-05   Min.   :2016
##  2016-01-31: 2000   1st Qu.: 6101   1st Qu.:2016-03-25   1st Qu.:2016
##  2016-03-02: 2000   Median :11939   Median :2016-06-17   Median :2016
##  2016-04-02: 2000   Mean   :12006   Mean   :2016-06-17   Mean   :2016
##  2016-05-02: 2000   3rd Qu.:17987   3rd Qu.:2016-09-08   3rd Qu.:2016
##  2016-06-02: 2000   Max.   :23999   Max.   :2016-12-02   Max.   :2016
##  (Other)   :12000
```

```
summary(sales)
```

```
##        X              invoice             sku            qty
##  Min.   :      1   Min.   :      1   Min.   :   1   Min.   :  0.00
##  1st Qu.: 325001   1st Qu.: 325001   1st Qu.: 501   1st Qu.: 15.00
##  Median : 650000   Median : 650000   Median :1001   Median : 40.00
##  Mean   : 650000   Mean   : 650000   Mean   :1001   Mean   : 53.46
```

```
##  3rd Qu.: 975000   3rd Qu.: 975000   3rd Qu.:1500   3rd Qu.: 77.00
##  Max.   :1300000   Max.   :1300000   Max.   :2000   Max.   :433.00
##
##     cashtrue              date           unitprice         total
##  Min.   :0.0000   2016-08-05:   3417   Min.   : 0.00   Min.   :    0.0
##  1st Qu.:0.0000   2016-06-12:   3379   1st Qu.: 9.84   1st Qu.:  180.6
##  Median :0.0000   2016-05-18:   3373   Median :15.14   Median :  526.1
##  Mean   :0.1538   2016-10-14:   3362   Mean   :16.58   Mean   :  886.2
##  3rd Qu.:0.0000   2016-07-01:   3360   3rd Qu.:22.26   3rd Qu.: 1202.4
##  Max.   :1.0000   2016-07-24:   3360   Max.   :54.16   Max.   :15174.1
##                   (Other)   :1279749
##     cust.no           datep               year
##  Min.   :   1.0   Min.   :2015-11-26   Min.   :2015
##  1st Qu.: 251.0   1st Qu.:2016-03-14   1st Qu.:2016
##  Median : 500.0   Median :2016-07-02   Median :2016
##  Mean   : 500.5   Mean   :2016-07-01   Mean   :2016
##  3rd Qu.: 750.0   3rd Qu.:2016-10-20   3rd Qu.:2016
##  Max.   :1000.0   Max.   :2017-02-06   Max.   :2017
##
```

```
str(collections)
```

```
## 'data.frame':    696596 obs. of  7 variables:
##  $ X         : int  517247 804718 848053 1064009 15894 66206 101111 113753 122276 1485
24 ...
##  $ collected : num  405 657 1310 1974 157 ...
##  $ invoice   : int  517247 804718 848053 1064009 15894 66206 101111 113753 122276 1485
24 ...
##  $ date      : Factor w/ 403 levels "2015-11-26","2015-11-27",..: 1 1 1 1 2 2 2 2 2 2
...
##  $ receipt.no: num  818083 421568 938254 847682 25042 ...
##  $ datep     : Date, format: "2015-11-26" "2015-11-26" ...
##  $ year      : num  2015 2015 2015 2015 2015 ...
```

```
str(credit)
```

```
## 'data.frame':    1000 obs. of  3 variables:
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ customer: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ limit   : int  109000 237000 252000 232000 238000 239000 224000 234000 224000 22100
0 ...
```

```
str(inventory)
```

```
## 'data.frame':    2000 obs. of  8 variables:
##  $ X         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sku       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ unitcost  : num  3.73 4.13 6.35 7.6 6.27 5.67 9.62 9.17 8.73 1.88 ...
##  $ unitprice : num  5.7 12.2 19.6 12.6 17.6 ...
##  $ beginstock: int  6714 11954 12676 13267 16358 13051 11230 14289 7397 13325 ...
##  $ endstock  : int  12175 11354 13096 12354 11049 9138 18720 12873 7709 11346 ...
##  $ defective : int  100 158 669 260 188 169 128 730 176 789 ...
##  $ returns   : int  12 21 72 53 29 28 24 205 34 86 ...
```

```
str(purchases)
```

```
## 'data.frame':    24000 obs. of  8 variables:
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sku     : int  1 10 100 1000 1001 1002 1003 1004 1005 1006 ...
##  $ unitcost: num  3.73 1.88 8.07 8.29 5.62 ...
##  $ quantity: int  976 3256 2210 2983 3491 3406 3180 3791 2624 2704 ...
##  $ date    : Factor w/ 12 levels "2016-01-05","2016-01-31",..: 1 1 1 1 1 1 1 1 1 ...
##  $ PO.no   : int  9343 19769 5419 10812 21359 7490 6063 12206 7496 8680 ...
##  $ datep   : Date, format: "2016-01-05" "2016-01-05" ...
##  $ year    : num  2016 2016 2016 2016 2016 ...
```

```
str(sales)
```

```
## 'data.frame':    1300000 obs. of  11 variables:
##  $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ invoice  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sku      : int  1739 170 1449 1480 739 189 1643 436 445 560 ...
##  $ qty      : int  124 59 5 38 60 44 60 55 42 142 ...
##  $ cashtrue : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ date     : Factor w/ 439 levels "2015-11-26","2015-11-27",..: 59 48 223 122 356 124
197 53 22 62 ...
##  $ unitprice: num  19.3 22.1 10.1 30.8 31.6 ...
##  $ total    : num  2394.4 1303.3 50.3 1170.8 1898.4 ...
##  $ cust.no  : int  595 467 273 589 387 418 411 93 307 274 ...
##  $ datep    : Date, format: "2016-01-23" "2016-01-12" ...
##  $ year     : num  2016 2016 2016 2016 2016 ...
```

QUESTION 3.

Determine the range of dates of sales, purchases and collections and compute:

```
salesRange <- range(as_date(sales$date))
collectionsRange <- range(as_date(collections$date))
purchasesRange <- range(as_date(purchases$date))
salesRange
```

```
## [1] "2015-11-26" "2017-02-06"
```

```
collectionsRange
```

```
## [1] "2015-11-26" "2017-01-01"
```

```
purchasesRange
```

```
## [1] "2016-01-05" "2016-12-02"
```

3.1. Compute the minimum, maximum, 1st and 3rd quartiles for the markup percentages on LSI???s sales, purchasing and collections transactions

Minimum, Maximum, 1st and 3rd Quartiles for Sales

```
#Pertaining to the Date
summary(as_date(sales$date))
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2015-11-26" "2016-03-14" "2016-07-02" "2016-07-01" "2016-10-20"
##         Max.
## "2017-02-06"
```

```
#Markup Perentages Summary
sales_markup=merge(sales,inventory,by="sku")
sales_markup$markup_value = (sales_markup$unitprice.x/sales_markup$unitcost)-1
summary (sales_markup$markup_value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.5026  1.0780  1.7450  1.7390  2.3750  3.0000     640
```

Minimum, Maximum, 1st and 3rd Quartiles for Collections

```
#Pertaining to the Date
summary(as_date(collections$date))
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2015-11-26" "2016-05-27" "2016-08-19" "2016-08-07" "2016-10-28"
##         Max.
## "2017-01-01"
```

```
#Markup Perentages Summary
collections_markup <- merge(collections,sales,by="invoice")
collections_markupT <- merge(collections_markup,inventory, by="sku")
collections_markupT$markup_value <- (collections_markupT$unitprice.x/collections_markupT$
unitcost)-1
summary (collections_markupT$markup_value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.5026  1.0790  1.7450  1.7390  2.3740  3.0000     336
```

Minimum, Maximum, 1st and 3rd Quartiles for Purchases

```
#Pertaining to the Date
summary(as_date(purchases$date))
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2016-01-05" "2016-03-25" "2016-06-17" "2016-06-17" "2016-09-08"
##         Max.
## "2016-12-02"
```

```
#Markup Perentages Summary
purchases_markup <- merge(purchases,inventory,by="sku")
purchases_markup$markup_value <- (purchases_markup$unitprice/purchases_markup$unitcost.x)
-1
summary (purchases_markup$markup_value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.5026  1.0770  1.7400  1.7380  2.3740  3.0000      12
```

3.2. Compute the daily averages for sales, purchases and collections transactions

```
sales$amt = sales$qty * sales$unitprice
salesagg = aggregate(amt~date,sales,sum)
head(salesagg)
```

```
##         date     amt
## 1 2015-11-26 2438867
## 2 2015-11-27 2911767
## 3 2015-11-28 2929302
## 4 2015-11-29 1842058
```

```
## 5 2015-11-30 2977050
## 6 2015-12-01 3035096

summary(salesagg$amt) #Average number of sales per day

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1758000 2022000 2843000 2624000 2912000 3099000

purchases$amt = purchases$unitcost*purchases$quantity
purchasesagg = aggregate(amt~date,purchases,sum)
summary(purchasesagg$amt)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 34880000 34880000 34880000 34880000 34880000 34880000

collectionsagg = aggregate(collected~date,collections,sum)
summary(collectionsagg$collected)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2631  917800 1705000 1514000 2148000 2597000
```

3.3. Do the ranges of dates of sales, purchases and collections lie within the fiscal year (2016) being audited?

```
#To prove that different year datas are present
sales.audit.year <- split(sales,sales.audit.year)
summary(sales.audit.year)
Length Class      Mode
2015 12     data.frame list
2016 12     data.frame list
2017 12     data.frame list

purchases.audit.year <- split(purchases,purchases$year)
purchases.audit.year <- split(purchases,purchases.audit.year)
summary(purchases.audit.year)
Length Class      Mode
2016   9      data.frame list

collections.audit.year <- split(collections,collections$year)
collections.audit.year <- split(collections,collections.audit.year)
summary(collections.audit.year)
Length Class      Mode
2015   7      data.frame list
2016   7      data.frame list
2017   7      data.frame list
```

It is pretty much evident that the ranges of dates of sales, purchases and collections DONOT lie within the fiscal year (2016)

3.4. If not, what corrections do you need to make to properly conduct the audit calculations you have made previously? We have to get the data of the audit year 2016 alone and compute the values. To extract the 2016 data of sales, we need to make use of the following R code :

```
sales$datep = as_date(sales$date) #Type Casting for computational ease
sales$year = year(sales$datep)
sales2016 = split(sales,sales$year)$'2016'
head(sales2016)
```

```
##   X invoice  sku qty cashtrue       date unitprice   total cust.no
## 1 1       1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2 2       2  170  59        0 2016-01-12     22.09 1303.31     467
## 3 3       3 1449   5        1 2016-07-05     10.06   50.30     273
## 4 4       4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5 5       5  739  60        0 2016-11-15     31.64 1898.40     387
## 6 6       6  189  44        0 2016-03-28     24.81 1091.64     418
##        datep year     amt
## 1 2016-01-23 2016 2394.44
## 2 2016-01-12 2016 1303.31
## 3 2016-07-05 2016   50.30
## 4 2016-03-26 2016 1170.78
## 5 2016-11-15 2016 1898.40
## 6 2016-03-28 2016 1091.64
```

3.5. Would any of your computed account balances in the Trial Balance change because of your findings?

Yes, the value of records (pertaining to computed account balances in the Trial Balance) in sales, accounts receivable, inventory and allowance for doubtful accounts would change.

QUESTION 4.

Perform the following audit program tests to determine the integrity of internal controls in LSI???s Revenue Cycle for all Sales transactions during the 2016 fiscal year:

4.1. Find any Duplicate transactions (i.e., where an Invoice sequence number appears more than once)

```
anyDuplicated(sales2016$invoice, incomparables = FALSE, fromLast = FALSE)
```

```
## [1] 0
```

Thus, we can infer that there are no duplications.

4.2. Find any Omitted transactions (i.e., where one or more Invoice sequence numbers have been skipped)

```
checklist <- seq(1, 1300000, by = 1) #In-order to verify whether all 1300000 entries are present
head(checklist[!match(checklist,salesA$invoice,nomatch=FALSE)],20)
```

```
##  [1]   9  12  13  15  26  38  46  59  60  75  84  85  87  99 100 103 116
## [18] 119 121 123
```

```
head(salesA$invoice,20)
```

```
##  [1]  1  2  3  4  5  6  7  8 10 11 14 16 17 18 19 20 21 22 23 24
```

Thus, we can infer that there are many (over 0.25 million) ommitted entries. For eg : The entries 9,12,13 (to name a few) are not found in the given dataset.

4.3. Perform a Sales Cutoff test, i.e., sales transactions listed as sales in on audit period (2016) but where ownership changed in another (reflected by a date of sale not in 2016)

```
sales$datep = as_date(sales$date) #Type Casting for computational ease
sales$year = year(sales$datep)
sales2015 <- split(sales,sales$year)$'2015'
```

```
sales2016 <- split(sales,sales$year)$'2016'
sales2017 <- split(sales,sales$year)$'2017'
```

Number of sales transactions NOT listed as sales in audit period (2016)

```
salesNOT2016 <- nrow(sales2015) + nrow(sales2017)
salesNOT2016
```

```
## [1] 216533
```

Number of sales transactions listed as sales in audit period (2016)

```
salesIN2016 <- nrow(sales2016)
salesIN2016
```

```
## [1] 1083467
```

```
head(sales2016) #To view first 6 rows
```

```
##   X invoice  sku qty cashtrue       date unitprice   total cust.no
## 1 1       1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2 2       2  170  59        0 2016-01-12     22.09 1303.31     467
## 3 3       3 1449   5        1 2016-07-05     10.06   50.30     273
## 4 4       4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5 5       5  739  60        0 2016-11-15     31.64 1898.40     387
## 6 6       6  189  44        0 2016-03-28     24.81 1091.64     418
##        datep year     amt
## 1 2016-01-23 2016 2394.44
## 2 2016-01-12 2016 1303.31
## 3 2016-07-05 2016   50.30
## 4 2016-03-26 2016 1170.78
## 5 2016-11-15 2016 1898.40
## 6 2016-03-28 2016 1091.64
```

The Sales Cutoff test

```
cbind(salesIN2016, salesNOT2016)
```

```
##      salesIN2016 salesNOT2016
## [1,]     1083467       216533
```

```
collections2017 <- split(collections,collections$year)$'2017'
collections_sales <- merge(collections2017,sales2016,by="invoice")
head(collections_sales)
```

```
##   invoice X.x collected     date.x receipt.no    datep.x year.x X.y sku
## 1     133 133   2631.09 2017-01-01     911259 2017-01-01   2017 133 649
##   qty cashtrue     date.y unitprice   total cust.no    datep.y year.y
## 1 187        0 2016-03-05     14.07 2631.09     780 2016-03-05   2016
##       amt
## 1 2631.09
```

QUESTION 5.

Compute and explain how you computed LSI???s Cost of Goods Sold?

5.1. What accounting principal was important in accurately making this calculation?

The accounting principle which is important in accurately making this calculation is the **Matching Principle**. In accrual accounting, the matching principle states that expenses should be recorded during the period in which they are incurred, regardless of when the transfer of cash occurs.

5.2. What is the average markup on LSI???s inventory items? The solution below displays the summary of LSI's inventory items with average markup as well as without average markup.

```
CGS=merge(salesA,inventory,by="sku")
#sum(CGS$qty*CGS$unitcost)
#sum (CGS$amt)
sum (CGS$amt)/ sum(CGS$qty*CGS$unitcost)
0.1736
```

We can infer that the Markup is 173.6 percent from the above code chunk

5.3. Compute the minimum, maximum, 1st and 3rd quartiles for the markup percentages on LSI???s inventory items

```
inventory$amt=inventory$unitcost*inventory$endstock
View (inventory)
summary (inventory$amt)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0   40890   67980   76380  103800  261600

#Based on markup percentages
inventory$markupvalue = ((inventory$unitprice/(inventory$unitcost))-1)
summary(inventory$markupvalue)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
0.5026  1.0770  1.7400  1.7380  2.3730  3.0000       1
```

QUESTION 6.

6.1. Compute the balance of unpaid Accounts Receivables (A/R) at 12/31/2016 from the datasets given to you. The following code computes the balance of unpaid

```
accountsReceivable=merge(sales2016,collections2016,by="invoice",all.x = TRUE)
head(accountsReceivable)

##   invoice X.x  sku qty cashtrue     date.x unitprice   total cust.no
## 1       1   1 1739 124        0 2016-01-23     19.31 2394.44     595
## 2       2   2  170  59        0 2016-01-12     22.09 1303.31     467
## 3       3   3 1449   5        1 2016-07-05     10.06   50.30     273
## 4       4   4 1480  38        0 2016-03-26     30.81 1170.78     589
## 5       5   5  739  60        0 2016-11-15     31.64 1898.40     387
## 6       6   6  189  44        0 2016-03-28     24.81 1091.64     418
##      datep.x year.x     amt X.y collected     date.y receipt.no    datep.y
## 1 2016-01-23   2016 2394.44   1   2394.44 2016-05-21     496082 2016-05-21
## 2 2016-01-12   2016 1303.31   2   1303.31 2016-09-09     515154 2016-09-09
## 3 2016-07-05   2016   50.30   3     50.30 2016-08-22     789087 2016-08-22
## 4 2016-03-26   2016 1170.78   4   1170.78 2016-06-17     590135 2016-06-17
## 5 2016-11-15   2016 1898.40   5   1898.40 2016-12-20     967507 2016-12-20
## 6 2016-03-28   2016 1091.64   6   1091.64 2016-10-15     624062 2016-10-15
##   year.y
## 1   2016
## 2   2016
## 3   2016
## 4   2016
```

```
## 5   2016
## 6   2016

accountsReceivable$collected[is.na(accountsReceivable$collected)]<-0
accountsReceivable$collected[accountsReceivable$collected<0]<-0

cash_Collected <- sum(accountsReceivable$collected)
cash_Collected

## [1] 524843737

cash_Total <- sum(accountsReceivable$total)
cash_Total

## [1] 960030574

diff_Amt = cash_Total - cash_Collected
diff_Amt

## [1] 435186837
```

6.2. Age the Accounts Receivables. The current balance in Allowance for Uncollectable Accounts Receivable is zero. Add to Allowance for Uncollectable Accounts Receivable the following percentages of Unpaid A/R: 1. **<90 days old 0%** 2. **90-180 days old 20% of A/R** 3. **>180 days old 40% of A/R**

```
for (i in 1:nrow(allowancetableRefined))
{
  if(allowancetableRefined$arddueperiod[i]>180){
    allowancetableRefined$allowanceamt[i] = 0.4*allowancetableRefined$ar[i]
  }else if(allowancetableRefined$arddueperiod[i]>=90 && allowancetableRefined$arddueperiod[
i]<=180){
    allowancetableRefined$allowanceamt[i] = 0.2*allowancetableRefined$ar[i]
  }else{
    allowancetableRefined$allowanceamt[i] = 0
  }
}

sum(allowancetableRefined$allowanceamt)
69,273,535
```

QUESTION 7

LSI, Inc is a high-end retailer, and preapproves customers for credit sale. Find any customers who have exceeded their limit (Customer Credit Limit dataset) at any time during the audit year (2016), and report the date and amount by which the limit is exceeded. How does this information influence your audit report, and where would you write-up this problem?

Whenever a customer makes a purchase, the details pertaining to the purchase (the credit and the purchase date) is recorded or stored.

The purchase details of each customer is appended by the subsequent purchase details. The credit Limit for the each cutomer is tracked and all the instances when the customer exceeds his/her credit limit in 2016 is noted down.

Further, the dates pertaining to the transaction when the credit limit is exceeded is also recorded.

```
auidityear <- interval(ymd(20160101), ymd(20161231))
sales$datep <- as_date(sales$date)
sales$year <- year(sales$datep)
Audit_Sales2016 <- split(sales,sales$year)$'2016'

auidityear <- interval(ymd(20160101), ymd(20161231))
collections$datep <- as_date(collections$date)
collections$year <- year(collections$datep)
Audit_Collections2016 <- split(collections,collections$year)$'2016'
```

Renaming the Columns and Merging the datasets for computational ease

```
colnames(Audit_Sales2016)[6]<- "Sales_Date"
colnames(Audit_Collections2016)[4]<-"Collection_Date"
colnames(Audit_Credit)[2] <- "cust.no"


merged1 <- merge(Audit_Collections, Audit_Sales, by="invoice",all.y = TRUE)
merged1 <-arrange(merged1,cust.no,Collection_Date)

merged2 <-merge(Audit_Credit, Audit_Sales, by="cust.no",all.y = TRUE)
merged2 <-arrange(m2,cust.no,Sales_Date)
```

Running iterations for computing the transactions

```
transactions<- m1[0,]
uniqueCustIds<-unique(m1$cust.no)
exceededBy<-data.frame(cust.no=0,invoice=0,currentCredit=0)
m1$collected[m1$collected<0]<-0

for(c in 1:nrow(tempCust1)) {
  tempCust1<-m1[(m1$cust.no==uniqueCustIds[c]),]
  tempCust2<-m2[(m2$cust.no==uniqueCustIds[c]),]

  tempCust1<-arrange(tempCust1,Collection_Date)
  tempCust2<-arrange(tempCust2,Sales_Date)

  creditLimitForTheParticularCustomer<-tempCust2[1,]$limit
  maxCredit<-creditLimitForTheParticularCustomer

  prevSalesDate<-tempCust2[1,]$Sales_Date
  previousSalesDate<-as.POSIXct(prevSalesDate)

  for(i in 1:nrow(tempCust1)) {

    creditLimitForTheParticularCustomer<-creditLimitForTheParticularCustomer-tempCust2[i,
]$total



    currentSalesDate<-as.POSIXct(tempCust2[i,]$Sales_Date)
    tempCust1$Collection_Date<-as.POSIXct(tempCust1$Collection_Date)
    if(is.na(tempCust1[i,]$Collection_Date)){
      collection<- tempCust1[0,]
    }else{
      collection<-tempCust1[(tempCust1[i,]$Collection_Date>previousSalesDate&&tempCust1[i
```

```
,]$Collection_Date<=currentSalesDate), ]
    }

    if(nrow(collection)>0){
      colAmt<-sum(collection$collected)
    }else{
      colAmt<-0
    }

    creditLimitForTheParticularCustomer<-creditLimitForTheParticularCustomer+colAmt

    if(creditLimitForTheParticularCustomer>maxCredit){
      creditLimitForTheParticularCustomer<-maxCredit
    }

    if(creditLimitForTheParticularCustomer<0) {
      transactions <- rbind(transactions, tempCust2[i,])

      exceededBy<-rbind(exceededBy,c(tempCust2[i,]$cust.no,tempCust2[i,]$invoice,creditLi
mitForTheParticularCustomer))
    }
    previousSalesDate<-currentSalesDate
  }
}
```

Merging datasets and performing appropriate arrangements in-order to create the output file

The spreadsheet included along with this word doc provides the details of the customers who have exceeded their credit limit in 2016.

```
exceededBy<-exceededBy[-1,]
trans = merge(transactions, exceededBy, by=c("invoice","cust.no"),all.x = TRUE)
trans<-arrange(trans,cust.no,Sales_Date)

write.csv(trans, file = "trans.csv")
```

QUESTION 8.

When an inventory item is not available in stock, LSI salespeople will complete the sale and place the item on backorder to be delivered to the customer when the stock arrives.

8.1. Has LSI, Inc. ???stocked out??? of any Inventory SKUs during the fiscal year (i.e., sold the item but had to backorder it since it was not in inventory at the time of the sale)? *Yes, LSI, Inc. has been "stocked out" for many Inventory SKUs during the fiscal year 2016.

8.2. Write a list of SKUs??? that have stocked out, when they stocked out and how much was the excess demand over inventory before the next shipment of inventory was received. Shipments are received at the beginning of the month, and the Purchase Order is dated on the date that inventory orders are received into inventory.

Computing the number of unique instances of Sales (based on sku's aggregate)

```
sales2016$month <- month(sales2016$date)
summary(sales2016)
```

```
##        X              invoice             sku            qty          
##  Min.   :      1   Min.   :      1   Min.   :   1   Min.   :  0.00  
##  1st Qu.: 325200   1st Qu.: 325200   1st Qu.: 501   1st Qu.: 15.00  
##  Median : 650363   Median : 650363   Median :1001   Median : 40.00  
##  Mean   : 650261   Mean   : 650261   Mean   :1001   Mean   : 53.44  
##  3rd Qu.: 975510   3rd Qu.: 975510   3rd Qu.:1500   3rd Qu.: 77.00  
##  Max.   :1300000   Max.   :1300000   Max.   :2000   Max.   :433.00  
##                                                                     
##     cashtrue              date          unitprice         total        
##  Min.   :0.0000   2016-08-05:   3417   Min.   : 0.00   Min.   :    0.0  
##  1st Qu.:0.0000   2016-06-12:   3379   1st Qu.: 9.84   1st Qu.:  180.6  
##  Median :0.0000   2016-05-18:   3373   Median :15.14   Median :  526.0  
##  Mean   :0.1538   2016-10-14:   3362   Mean   :16.58   Mean   :  886.1  
##  3rd Qu.:0.0000   2016-07-01:   3360   3rd Qu.:22.26   3rd Qu.: 1202.1  
##  Max.   :1.0000   2016-07-24:   3360   Max.   :54.16   Max.   :15174.1  
##                   (Other)   :1063216                                    
##     cust.no           datep                year           amt          
##  Min.   :   1.0   Min.   :2016-01-01   Min.   :2016   Min.   :    0.0  
##  1st Qu.: 250.0   1st Qu.:2016-04-01   1st Qu.:2016   1st Qu.:  180.6  
##  Median : 500.0   Median :2016-07-01   Median :2016   Median :  526.0  
##  Mean   : 500.2   Mean   :2016-07-01   Mean   :2016   Mean   :  886.1  
##  3rd Qu.: 750.0   3rd Qu.:2016-10-01   3rd Qu.:2016   3rd Qu.: 1202.1  
##  Max.   :1000.0   Max.   :2016-12-31   Max.   :2016   Max.   :15174.1  
##                                                                        
##      month       
##  Min.   : 1.000  
##  1st Qu.: 4.000  
##  Median : 7.000  
##  Mean   : 6.512  
##  3rd Qu.:10.000  
##  Max.   :12.000  
##                  
```

```
sales2016$sku <- as.character(sales2016$sku)

salesbySkuAggregate <- aggregate(qty~sku+month, sales2016, sum)
nrow(salesbySkuAggregate)
```

```
## [1] 24000
```

```
colnames(salesbySkuAggregate)[3] <- "quantity_sold" #Renaming for computational ease
```

Computing the Stockout/backorder as well as the Number of distinct instances

```
purchases <- purchases[,-c(1)]
purchases$month <- month(purchases$date) + 1
head(purchases)
```

```
##    sku unitcost quantity       date PO.no      datep year      amt month
## 1    1     3.73      976 2016-01-05  9343 2016-01-05 2016  3640.48     2
## 2   10     1.88     3256 2016-01-05 19769 2016-01-05 2016  6121.28     2
## 3  100     8.07     2210 2016-01-05  5419 2016-01-05 2016 17834.70     2
## 4 1000     8.29     2983 2016-01-05 10812 2016-01-05 2016 24729.07     2
## 5 1001     5.62     3491 2016-01-05 21359 2016-01-05 2016 19619.42     2
## 6 1002     8.78     3406 2016-01-05  7490 2016-01-05 2016 29904.68     2
```

```
purchases$sku <- as.character(purchases$sku)
colnames(purchases)[3]<-"quantity_fullfilled"

#AGGREGATION

aggregateMonth<-merge(purchasesSales,y=inventory[,c('beginstock','sku','month')],by=c('sku','month'),all.x = T)
aggregateMonth$quantity_fullfilled[is.na(aggregateMonth$quantity_fullfilled)] <- 0
aggregateMonth$beginstock[is.na(aggregateMonth$beginstock)] <- 0
aggregateMonth<-aggregateMonth[with(aggregateMonth,order(aggregateMonth[,2])),]

#Calculating stockout at the end of each month
for(i in 1:length(aggregateMonth$sku)){
  eachMonth<- aggregateMonth$month[i]
  newSku<-aggregateMonth$sku[i]
  nrow(aggregateMonth)
  aggregateMonth$onhand[i]<-aggregateMonth$beginstock[i]+aggregateMonth$quantity_fullfilled[i]-aggregateMonth$quantity_sold[i]
  aggregateMonth<- within(aggregateMonth,beginstock[sku==newSku & month == eachMonth+1]<-aggregateMonth$onhand[i])
}
stockout<-aggregateMonth[aggregateMonth$onhand<0,c(1,2,6)]
nrow(stockout)

## [1] 354

length(unique(stockout$sku))

## [1] 86
```

Thus we can infer that there are *354 stockout/bordorder and 86 unique instances of stockout. There are almost 30 SKUs. The excess demand is calculated below. The generated output file is included in the dropbox.*

Computing the Excess Demand in each month and returning it as a table for better understandability

```
stockout$month[stockout$month == 1] <- 'Jan'
stockout$month[stockout$month == 2] <- 'Feb'
stockout$month[stockout$month == 3] <- 'Mar'
stockout$month[stockout$month == 4] <- 'Apr'
stockout$month[stockout$month == 5] <- 'May'
stockout$month[stockout$month == 6] <- 'Jun'
stockout$month[stockout$month == 7] <- 'Jul'
stockout$month[stockout$month == 8] <- 'Aug'
stockout$month[stockout$month == 9] <- 'Sep'
stockout$month[stockout$month == 10] <- 'Oct'
stockout$month[stockout$month == 11] <- 'Nov'
stockout$month[stockout$month == 12] <- 'Dec'
table(stockout$month)

##
##  Apr  Aug  Dec  Jul  Jun  May  Nov  Oct  Sep
##  2    35   84   31   16   7    74   60   45
```