Week 2 – R Data Modeling

Previously on Financial Analytics...

- Processing existing data into `R`
- String manipulation, scraping and collecting data
- Arrays, matrices, vectors

This week

- Pivot tables in `R`
- VLOOKUP in `R`
- Functions and control

Try this. . .

Define

1. Pivot table
2. VLOOKUP

Thinking. . .

Results

Pivot table

- Data summarization tool that can automatically sort, count, total, or give the average of the data stored in one table or spreadsheet, displaying the results in a second table showing the summarized data
- Tool that transforms a flat table of fields with rows of data into a table with grouping row values and column header values that rotate the flat table's data rows into the intersection of the row and column labels

VLOOKUP

- "V" or "vertical" stands for the looking up of a value in a column

Pivot tables and Vertical Lookups

- Two of the most-used Excel functions
- Here made easier and less apt to crash on large data sets
- Start with pivot tables

Whitman

SCHOOL *of* MANAGEMENT

SYRACUSE UNIVERSITY

MBA@SYRACUSE

Pivot and Parry

Credit Card Applicant business questions:

1. What is the income risk across applicant pools?
2. Are there differences in applicant income?
3. Does age matter?
4. Is there a pattern of dependents across applicant pools?
5. How much income per dependent?

Dimensions

1. Card status
2. Ownership
3. Employment

```r
CreditCard <- read.csv("data/CreditCard.csv")
str(CreditCard)
```

```
## 'data.frame':    1319 obs. of  13 variables:
##  $ card       : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ reports    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age        : num  37.7 33.2 33.7 30.5 32.2 ...
##  $ income     : num  4.52 2.42 4.5 2.54 9.79 ...
##  $ share      : num  0.03327 0.00522 0.00416 0.06521 0.06705 ...
##  $ expenditure: num  124.98 9.85 15 137.87 546.5 ...
##  $ owner      : Factor w/ 2 levels "no","yes": 2 1 2 1 2 1 1 2 2 1 ...
##  $ selfemp    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ dependents : int  3 3 4 0 2 0 2 0 0 0 ...
##  $ months     : int  54 34 58 25 64 54 7 77 97 65 ...
##  $ majorcards : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ active     : int  12 13 5 7 5 1 5 3 6 18 ...
##  $ state      : Factor w/ 3 levels "CT","NJ","NY": 3 3 3 3 3 3 3 3 3 3 ...
```

```r
head(CreditCard, 3)
```

```
##   card reports      age income       share expenditure owner selfemp
## 1  yes       0 37.66667   4.52 0.033269910  124.983300   yes      no
## 2  yes       0 33.25000   2.42 0.005216942    9.854167    no      no
## 3  yes       0 33.66667   4.50 0.004155556   15.000000   yes      no
##   dependents months majorcards active state
## 1          3     54          1     12    NY
## 2          3     34          1     13    NY
## 3          4     58          1      5    NY
```

```r
summary(CreditCard)
```

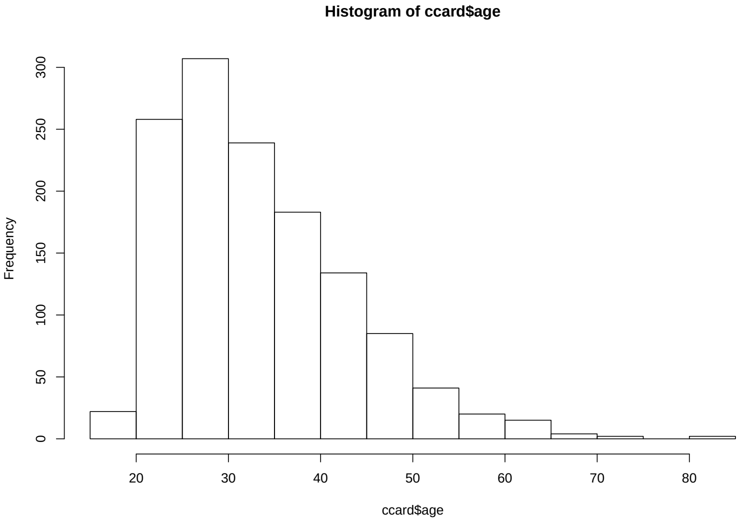
```
##   card          reports              age                income
##  no : 296   Min.   : 0.0000   Min.   : 0.1667   Min.   : 0.210
##  yes:1023   1st Qu.: 0.0000   1st Qu.:25.4167   1st Qu.: 2.244
##             Median : 0.0000   Median :31.2500   Median : 2.900
##             Mean   : 0.4564   Mean   :33.2131   Mean   : 3.365
##             3rd Qu.: 0.0000   3rd Qu.:39.4167   3rd Qu.: 4.000
##             Max.   :14.0000   Max.   :83.5000   Max.   :13.500
##      share            expenditure        owner     selfemp
##  Min.   :0.0001091   Min.   :   0.000   no :738   no :1228
##  1st Qu.:0.0023159   1st Qu.:   4.583   yes:581   yes:  91
##  Median :0.0388272   Median : 101.298
##  Mean   :0.0687322   Mean   : 185.057
##  3rd Qu.:0.0936168   3rd Qu.: 249.036
##  Max.   :0.9063205   Max.   :3099.505
##    dependents         months         majorcards          active
##  Min.   :0.0000   Min.   :  0.00   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 12.00   1st Qu.:1.0000   1st Qu.: 2.000
##  Median :1.0000   Median : 30.00   Median :1.0000   Median : 6.000
##  Mean   :0.9939   Mean   : 55.27   Mean   :0.8173   Mean   : 6.997
##  3rd Qu.:2.0000   3rd Qu.: 72.00   3rd Qu.:1.0000   3rd Qu.:11.000
##  Max.   :6.0000   Max.   :540.00   Max.   :1.0000   Max.   :46.000
##  state
##  CT:442
##  NJ:472
##  NY:405
```

Age minimum is 0.2? Let's filter the data for ages greater than 18:

```
ccard <- CreditCard[CreditCard$age >=
    18, ]
```

... and look at the distribution of ages of applicants:

```r
hist(ccard$age)
```



Try this

What is the basic design of this inquiry?

1. Business questions?
2. Dimensions?
3. Taxonomy and metrics?

Thinking...

Results

1 and 2. Our business questions require answers along the lines of indicator variables:

- Card issued (`card`)
- Own or rent (`owner`)
- Self-employed or not (`selfemp`)

3. So our basic taxonomy is:

1. For each card issued. . . in New York
2. . . . and for each owner. . .
3. . . . who is employed. . .
4. What are the range of income, average dependents, age, and income per dependent?

Basic 3 step pivot table design

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```r
# 1: filter to keep three states.
pvt_table <- filter(ccard, state %in%
    "NY")

# 2: set up data frame for by-group
# processing.
pvt_table <- group_by(pvt_table, card,
    owner, selfemp)

# 3: calculate the three summary
# metrics
options(dplyr.width = Inf)  # to display all columns
pvt_table <- summarise(pvt_table, income.cv = sd(income)/mean(income),
    age.avg = mean(age), income.per.dependent = sum(income)/sum(dependents))
```

```r
knitr::kable(pvt_table)
```

card	owner	selfemp	income.cv	age.avg	income.per.dependent
no	no	no	0.4941859	31.91936	3.645848
no	no	yes	0.5652634	26.38542	2.852000
no	yes	no	0.3756274	36.01786	2.157589
no	yes	yes	NaN	53.33333	Inf
yes	no	no	0.3298633	28.09311	5.313677
yes	no	yes	0.4367858	37.45238	7.062500
yes	yes	no	0.5519888	36.79503	3.154476
yes	yes	yes	0.5032180	41.91667	3.194547

Now to VLOOKUP

Load this IBRD (World Bank) data.

- The variable `life.expectancy` is the average life expectancy for each country from 2009 through 2014.
- The variable `sanitation` is the percentage of population with direct access to sanitation facilities.

```
le <- read.csv("data/life_expectancy.csv",
    header = TRUE, stringsAsFactors = FALSE)
sa <- read.csv("data/sanitation_.csv",
    header = TRUE, stringsAsFactors = FALSE)
```

```r
head(le)
```

```
##                country years.life.expectancy.avg
## 1          Afghanistan                  46.62135
## 2              Albania                  71.11807
## 3              Algeria                  61.81652
## 4               Angola                  41.65847
## 5  Antigua and Barbuda                  69.81219
## 6           Arab World                  60.93432
```

```r
head(sa)
```

```
##           country sanitation.avg
## 1     Afghanistan       25.39600
## 2         Albania       85.36154
## 3         Algeria       84.21538
## 4  American Samoa       61.73077
## 5         Andorra      100.00000
## 6          Angola       36.00769
```

The job is to join sanitation data with life expectancy data, by country.

In Excel we would typically use a `VLOOKUP(country, sanitation, 2, FALSE)` statement.

1. In this statement `country` is the value to be looked up, for example, "Australia".
2. The variable `sanitation` is the range of the sanitation lookup table of two columns of country and sanitation data, for example, `B2:C104` in Excel.
3. The `2` is the second column of the sanitation lookup table, for example column `C`.
4. `FALSE` means don't find an exact match.

In `R` we use the `merge()` function.

```
life.sanitation <- merge(le[, c("country",
    "years.life.expectancy.avg")], sa[,
    c("country", "sanitation.avg")])
```

The whole range of countries is populated by the lookup.

```r
head(life.sanitation, 3)
```

```
##       country years.life.expectancy.avg sanitation.avg
## 1 Afghanistan                  46.62135       25.39600
## 2     Albania                  71.11807       85.36154
## 3     Algeria                  61.81652       84.21538
```

Try this out

Load this data on house prices. Suppose you work for a housing developer like Toll Brothers (NYSE: TOL) and want to allocate resources to marketing and financing the building of luxury homes in major US metropolitan areas. You have data for one test market.

```
hprice <- read.csv("data/hprice.csv")
```

Questions

1. What are the most valuable (higher price) neighborhoods?
2. What housing characteristics maintain the most housing value?

Thinking...

Some results

Where and what are the most valuable houses?

One way to answer this is to build a pivot table. But first let's look at the available data:

```
hprice <- read.csv("data/hprice.csv")
head(hprice)
```

```
##   ID  Price SqFt Bedrooms Bathrooms Offers Brick Neighborhood
## 1  1 114300 1790        2         2      2    No         East
## 2  2 114200 2030        4         2      3    No         East
## 3  3 114800 1740        3         2      1    No         East
## 4  4  94700 1980        3         2      3    No         East
## 5  5 119800 2130        3         3      3    No         East
## 6  6 114600 1780        3         2      2    No        North
```

```r
summary(hprice)
```

```
##        ID             Price             SqFt         Bedrooms    
##  Min.   :  1.00   Min.   : 69100   Min.   :1450   Min.   :2.000  
##  1st Qu.: 32.75   1st Qu.:111325   1st Qu.:1880   1st Qu.:3.000  
##  Median : 64.50   Median :125950   Median :2000   Median :3.000  
##  Mean   : 64.50   Mean   :130427   Mean   :2001   Mean   :3.023  
##  3rd Qu.: 96.25   3rd Qu.:148250   3rd Qu.:2140   3rd Qu.:3.000  
##  Max.   :128.00   Max.   :211200   Max.   :2590   Max.   :5.000  
##    Bathrooms         Offers        Brick    Neighborhood
##  Min.   :2.000   Min.   :1.000   No :86   East :45    
##  1st Qu.:2.000   1st Qu.:2.000   Yes:42   North:44    
##  Median :2.000   Median :3.000            West :39    
##  Mean   :2.445   Mean   :2.578                        
##  3rd Qu.:3.000   3rd Qu.:3.000                        
##  Max.   :4.000   Max.   :6.000                        
```



```r
library(dplyr)

# 1: filter to those houses with
# fairly high prices
pvt_table <- filter(hprice, Price > 99999)

# 2: set up data frame for by-group
# processing
pvt_table <- group_by(pvt_table, Brick,
    Neighborhood)

# 3: calculate the summary metrics
options(dplyr.width = Inf)  # to display all columns
pvt_table <- summarise(pvt_table, Price.avg = mean(Price),
    Price.cv = sd(Price)/mean(Price),
    SqFt.avg = mean(SqFt), Price.per.SqFt = mean(Price)/mean(SqFt))
```

```r
knitr::kable(pvt_table)
```

Brick	Neighborhood	Price.avg	Price.cv	SqFt.avg	Price.per.SqFt
No	East	121095.7	0.1251510	2019.565	59.96125
No	North	115307.1	0.0939797	1958.214	58.88382
No	West	148230.4	0.0912350	2073.478	71.48878
Yes	East	135468.4	0.0977973	2031.053	66.69863
Yes	North	118457.1	0.1308498	1857.143	63.78462
Yes	West	175200.0	0.0930105	2091.250	83.77764

Based on this data set from one metropolitan area, the most valuable properties (fetching the highest average price and price per square foot) are made of brick in the West neighborhood. Brick or not, the West neighborhood also seems have the lowest relative variation in price.

Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Why functions?

- Data structures tie related values into one object.
- Functions tie related commands into one object.
- In both cases: easier to understand, easier to work with, easier to build into larger things

For example, here is an Excel look-alike NPV function

```
# Net Present Value function Inputs:
# vector of rates (rates) with 0 as
# the first rate for time 0, vector
# of cash flows (cashflows) Outputs:
# scalar net present value
NPV.1 <- function(rates, cashflows) {
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

Generate data internal to the function

- Use seq_along to generate time index of cashflows.
- Be sure to subtract 1 from this sequence as starting cashflow is time 0.

Our functions get used just like the built-in ones:

```
rates <- c(0, 0.08, 0.06, 0.04)  # first rate is always 0.00
cashflows <- c(-100, 200, 300, 10)
NPV.1(rates, cashflows)
```

```
## [1] 361.0741
```

Go back to the declaration and look at the parts:

```
# Net Present Value function Inputs:
# vector of rates (rates) with 0 as
# the first rate for time 0, vector
# of cash flows (cashflows) Outputs:
# scalar net present value
NPV.1 <- function(rates, cashflows) {
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

Interfaces: the **inputs** or **arguments**; the **outputs** or **return value**

- Calls other functions `sum`, `seq_along()`, operators /, +, ^ and - . could also call other functions we've written
- `return()` explicitly says what the output is: good documentation . alternately, return the last evaluation; explicit returns are better documentation

Comments: Not required by R, but always a Good Idea

One-line description of purpose - Listing of arguments - Listing of outputs

What should be a function?

- Things you're going to re-run, especially if it will be re-run with changes
- Chunks of code you keep highlighting and hitting return on
- Chunks of code which are small parts of bigger analyses
- Chunks which are very similar to other chunks

Named and default arguments

```
# Internal Rate of Return (IRR)
# function Inputs: vector of cash
# flows (cashflows), scalar
# interations (maxiter) Outputs:
# scalar net present value
IRR.1 <- function(cashflows, maxiter = 1000) {
    t <- seq_along(cashflows) - 1
    # rate will eventually converge to
    # IRR
    f <- function(rate) (sum(cashflows/(1 +
        rate)^t))
    # use uniroot function to solve for
    # root (IRR = rate) of f = 0 c(-1,1)
    # bounds solution for only positive
    # or negative rates select the root
    # estimate
    return(uniroot(f, c(-1, 1), maxiter = maxiter)$root)
}
```

Default argument

- `maxiter` controls the number of iterations.
- We can eliminate this argument if we want (perhaps at our peril!)

```r
# Here are the cashflows for a 3\%
# coupon bond bought at a hefty
# premium
cashflows <- c(-150, 3, 3, 3, 3, 3, 3,
    3, 103)
IRR.1(cashflows)
```

```
## [1] -0.02554088
```

```r
IRR.1(cashflows, maxiter = 100)
```

```
## [1] -0.02554088
```

Negative Interest Rates

- We get a negative IRR or yield to maturity on this net present value $= 0$ calculation.

Shoot the trouble

Problem: Odd behavior when arguments aren't as we expect

```
NPV.1(c(0.1, 0.05), c(-10, 5, 6, 100))
```

```
## [1] 86.10434
```

We do get a solution, but...

- What does it mean? What rates correspond with what cashflows?
- *Solution*: Put sanity checks into the code.
- Use `stopifnot(some logical statment)` is TRUE.

```r
# Net Present Value function Inputs:
# vector of rates (rates) with 0 as
# the first rate for time 0, vector
# of cash flows (cashflows), length
# of rates must equal length of
# cashflows Outputs: scalar net
# present value
NPV.2 <- function(rates, cashflows) {
    stopifnot(length(rates) == length(cashflows))
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

stopifnot TRUE error handling

- Arguments to `stopifnot()` are a series of logical expressions which should all be TRUE.
- Execution halts, with error message, at *first* FALSE.

```r
NPV.2(c(0.1, 0.05), c(-10, 5, 6, 100))
```

Hit (not too hard!) the `Escape` key on your keyboard

This will take you out of `Browse[1]>` mode and back to the console prompt `>`.

What the function can see and do

- Each function has its own environment.
- Names here override names in the global environment.
- Internal environment starts with the named arguments.
- Assignments inside the function only change the internal environment.
- Names undefined in the function are looked for in the environment the function gets called from.

Try this . . .

Your company is running a £100 project in the EU. You must post 25% collateral in a Landesbank using only high-quality government securities. You find a high-quality gilt fund that will pay 1.5% (coupon rate) annually for three years.

Questions

1. How much would you pay for this collateral if the rate curve (yield to maturity of cash flows) is (from next year on...)

```r
rates <- c(-0.001, 0.002, 0.01)
```

2. Suppose a bond dealer asks for 130% of notional collateral value for this bond. What is the yield on this transaction (IRR)? Would you buy it?
3. What is the return on this collateral if you terminate the project in one year and liquidate the collateral (i.e., sell it for cash) if the yield shifts down by 0.005? This is a "parallel" shift, which is finance for: "take each rate and deduct 0.005."

Thinking...

Some answers

Build rates and cash flows across the 3-year time frame:

```
(rates <- c(0, rates))
```

```
## [1]  0.000 -0.001  0.002  0.010
```

```
collateral.periods <- 3
collateral.rate <- 0.25
collateral.notional <- collateral.rate *
    100
coupon.rate <- 0.015
cashflows <- rep(coupon.rate * collateral.notional,
    collateral.periods)
cashflows[collateral.periods] <- collateral.notional +
    cashflows[collateral.periods]
(cashflows <- c(0, cashflows))
```

```
## [1]  0.000  0.375  0.375 25.375
```

What just happened...

1. Append a `0` to the rate schedule so we can use the `NPV.2` function.
2. Parameterize the term sheet (terms of the collateral transaction),
3. `rep()` coupon cash flows.
4. Add notional value repayment to the last cash flow.

Find the present value of the bond using `NPV.2`:

```
(Value.0 <- NPV.2(rates, cashflows))
```

```
## [1] 25.3776
```

The answer is £25.378

or `Value.0 / collateral.notional` times the notional value.

The yield to maturity averages the forward rates across the bond cash flows. This is one interpretation of the Internal Rate of Return ("IRR").

```
cashflows.IRR <- cashflows
collateral.ask <- 130
cashflows.IRR[1] <- -(collateral.ask/100) *
    collateral.notional
# mind the negative sign!
(collateral.IRR.1 <- IRR.1(cashflows.IRR))
```

```
## [1] -0.07112366
```

You end up paying over 7% per annum for the privilege of owning this bond! You call up the European Central Bank, report this overly hefty haircut on your project. You send out a request for proposal to other bond dealers. They come back with an average asking price of 109 (109% of notional).

```
cashflows.IRR <- cashflows
collateral.ask <- 109
cashflows.IRR[1] <- -(collateral.ask/100) *
    collateral.notional
(collateral.IRR.1 <- IRR.1(cashflows.IRR))
```

```
## [1] -0.01415712
```

That's more like it: about 140 basis points (1.41% x 100 basis points per percentage) cost (negative sign).

Unwind the project, and the collateral transaction, in 1 year. Let's suppose the yield curve in 1 year has parallel shifted down by 0.005.

```
rate.shift <- -0.005
rates.1 <- c(0, rates[-2]) + rate.shift
cashflows.1 <- c(0, cashflows[-2])
(Value.1 <- NPV.2(rates.1, cashflows.1))
```

```
## [1] 25.37541
```

```
(collateral.return.1 <- Value.1/(-cashflows.IRR[1]) -
    1)
```

```
## [1] -0.0687923
```

Looks much more than a break-even return on the collateral transation:

```
(collateral.gainloss <- collateral.notional *
    collateral.return.1) * 1e+06
```

```
## [1] -1719807
```

```
# adjust for millions of euros
```

That's probably someone's salary. . . (in pounds sterling).

Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Mind the Interface!

- Interfaces mark out a controlled inner environment for our code;
- Interact with the rest of the system only at the interface.
- Advice: arguments explicitly give the function all the information.
  - Reduces risk of confusion and error
  - Exception: true universals like π
- Likewise, output should only be through the return value. More about breaking up tasks and about environments later

Further reading:

Herbert Simon, *The Sciences of the Artificial*

Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Making distributions

- As always, let's load some data: let's use and open data package called `pdfetch`. This is a portal to finance and government data, including Yahoo Finance.
- Let's go to the Bureau of Labor Statistics (BLS) and load the export-import price index at `http://data.bls.gov/timeseries/EIUIR?output_view=pct_1mth`
- Look up the symbols "EIUIR" and "EIUIR100".

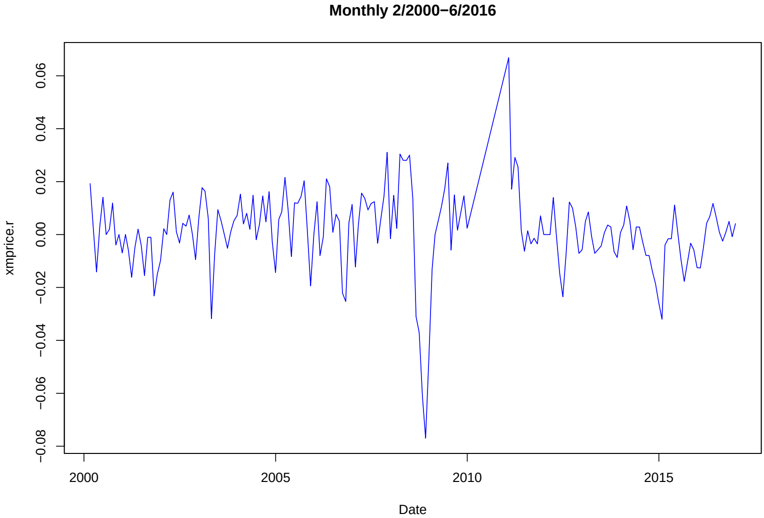
```
require(pdfetch)
```

```
## Warning: package 'pdfetch' was built under R version 3.4.3
```

```
require(xts)
require(zoo)
EIUIR <- pdfetch_BLS(c("EIUIR", "EIUIR100"),
    2000, 2016)  # start and end years
head(EIUIR)
```

```
##            EIUIR EIUIR100
## 2000-01-31  97.8     87.2
## 2000-02-29  99.7    100.2
## 2000-03-31  99.9     99.4
```

```r
plot(xmprice.r, type = "l", col = "blue",
    xlab = "Date", main = "Monthly 2/2000-6/2016")
```



```
xmprice.r.df <- data.frame(xmprice.r,
    Date = index(xmprice.r), Rate = xmprice.r[,
        1], Rate.abs = abs(xmprice.r[,
        1]))
head(xmprice.r.df)
```

```
##                    EIUIR       Date         Rate    Rate.abs
## 2000-02-29  0.019241100 2000-02-29  0.019241100 0.019241100
## 2000-03-31  0.002004009 2000-03-31  0.002004009 0.002004009
## 2000-04-30 -0.014113137 2000-04-30 -0.014113137 0.014113137
## 2000-05-31  0.003041057 2000-05-31  0.003041057 0.003041057
## 2000-06-30  0.014070584 2000-06-30  0.014070584 0.014070584
## 2000-07-31  0.000000000 2000-07-31  0.000000000 0.000000000
```

```
str(xmprice.r.df)
```

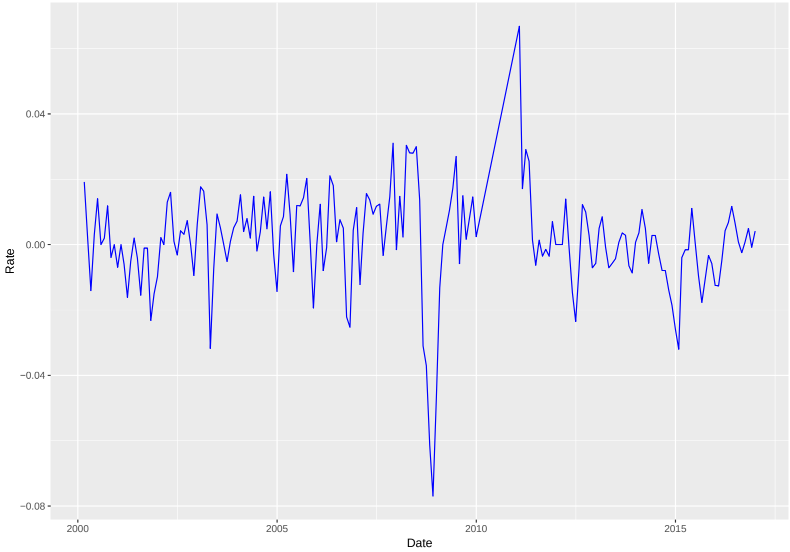
```
## 'data.frame':    191 obs. of  4 variables:
##  $ EIUIR   : num  0.01924 0.002 -0.01411 0.00304 0.01407 ...
##  $ Date    : Date, format: "2000-02-29" "2000-03-31" ...
##  $ Rate    : num  0.01924 0.002 -0.01411 0.00304 0.01407 ...
##  $ Rate.abs: num  0.01924 0.002 0.01411 0.00304 0.01407 ...
```

- A “prettier” plot with the ggplot2 package
- Use aes, “aesthetics”, to pick x (horizontal) and y (vertical) axes
- Use geom_line to build the plot

```
require(ggplot2)
ggplot(xmprice.r.df, aes(x = Date, y = Rate)) +
    geom_line(colour = "blue")
```

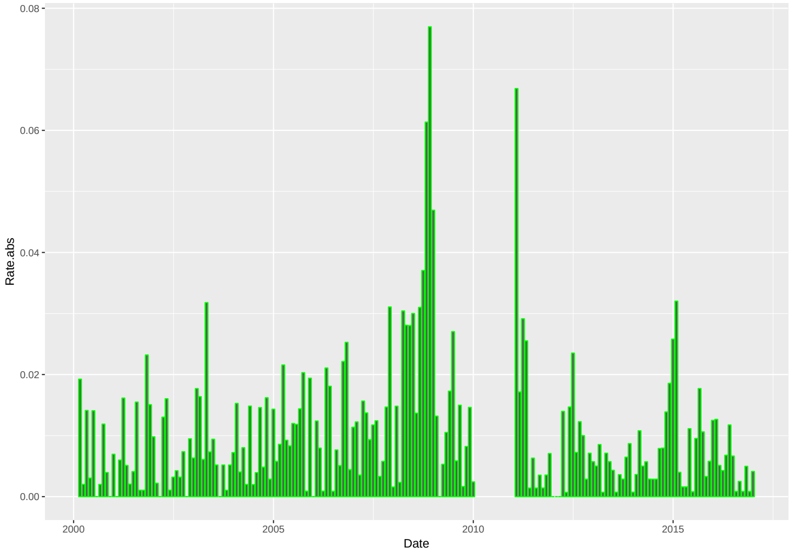


```
## Warning: package 'ggplot2' was built under R version 3.4.2
```



- Let's try a bar graph of the absolute value of price rates.
- Use `geom_bar` to build this picture.

```
require(ggplot2)
ggplot(xmprice.r.df, aes(x = Date, y = Rate.abs)) +
    geom_bar(stat = "identity", colour = "green")
```

0.08
0.06
0.04
0.02
0.00
Rate.abs
2000
2005
2010
2015
Date

Try this

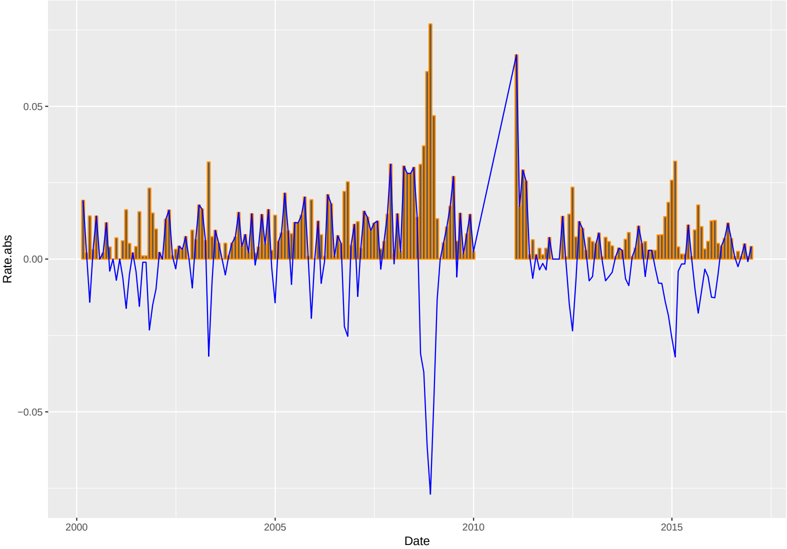
- Overlay returns (`geom_line`) and their absolute value `geom_bar`.
- `ggplot` declares the canvas using the price data frame.
- `aes` establishes the data series to be used to generate pictures.
- `geom_bar` builds bar chart.
- `geom_line` overplots bar chart with a line chart.

```
require(ggplot2)
ggplot(xmprice.r.df, aes(Date, Rate.abs)) +
    geom_bar(stat = "identity", colour = "darkorange") +
    geom_line(data = xmprice.r.df, aes(Date,
        Rate), colour = "blue")
```

By examining this chart, what business questions about your Univeral Export-Import Ltd supply chain could this help answer? Why is this helpful?

Thinking. . .

Results



1. Answers the question: When supply and demand tightens, does price volatility cluster?
2. If we are selling, we would experience strong swings in demand and thus in revenue at the customer fulfillment end of the chain.
3. If we are buying, we would experience strong swings in cost and input product utilization at the procurement end of the chain.
4. For the financial implications: we would have a tough time making the earnings we forecast to the market.

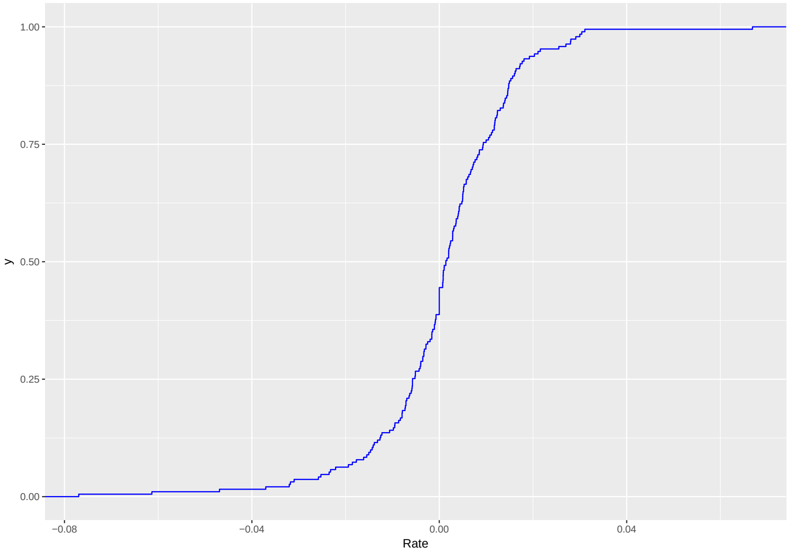
Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Picture this

- We import goods as input to our manufacturing process.
- We might want to know the odds that a very high export-import rate might occur.
- We answer this with a cumulative distribution function (*cdf* or *CDF*) plot. _ we build this plot using the `stat_ecdf()` function in `ggplot2`.

```r
require(ggplot2)
ggplot(xmprice.r.df, aes(Rate)) + stat_ecdf(colour = "blue")
```


1.00
0.75
0.50
0.25
0.00
y
−0.08
−0.04
0.00
0.04
Rate

Try this

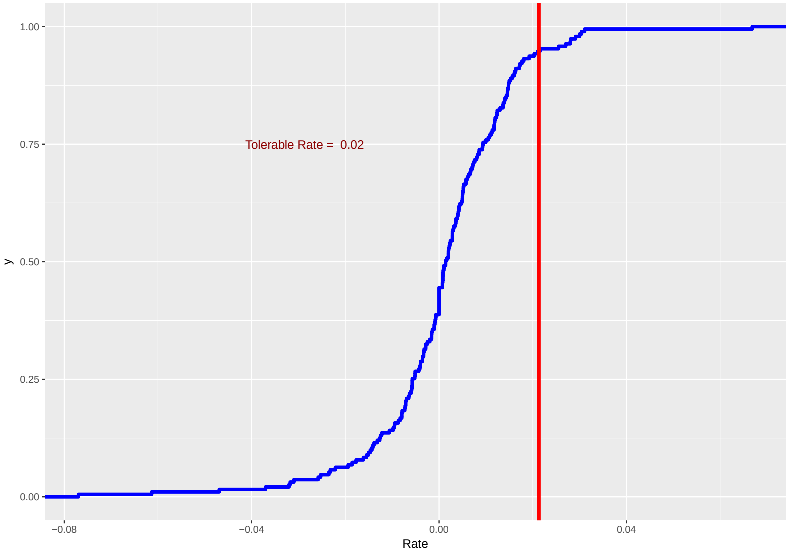
1. Suppose the procurement team's delegation of authority remit states: "Procurement may approve input invoices when there is only a 5% chance that prices will rise any higher than the price rate associated with that tolerance. If input prices do rise higher than the tolerable rate, you must get divisional approval."
2. Plot a vertical line to indicate the maximum tolerable rate for procurement using the BLS EIUR data from 2000 to the present.

- Use `r.tol <- quantile(xmprice.r.df$Rate, 0.95)` to find the tolerable rate.
- Use `+ geom_vline(xintercept = r.tol)` in the CDF plot.

Thinking...

Result

```r
require(ggplot2)
r.tol <- quantile(xmprice.r.df$Rate,
    0.95)
r.tol.label <- paste("Tolerable Rate = ",
    round(r.tol, 2))
ggplot(xmprice.r.df, aes(Rate)) + stat_ecdf(colour = "blue",
    size = 1.5) + geom_vline(xintercept = r.tol,
    colour = "red", size = 1.5) + annotate("text",
    x = r.tol - 0.05, y = 0.75, label = r.tol.label,
    colour = "darkred")
```


Tolerable Rate = 0.02
1.00
0.75
0.50
0.25
0.00
y
−0.08
−0.04
0.00
0.04
Rate

A little more than you bargained for?

- We used the `paste` and `round` (to two, 2, decimal places) functions to make a label.
- We made much thicker lines (`size = 1.5`).
- 2% is where the line is drawn.

That was intense!

Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Next on the agenda

Now that we have *made* some distributions out of live data, let's estimate the parameters of specific distributions that might be fit to that data.

Whitman
SCHOOL *of* MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE

Last...but not least

- Optimization, that is
- Otherwise known as finding the distribution that best fits the data
- So we can simulate that data to help us make decisions *prospectively*
- Use `fitdistr` to help us out

Many distributions in R: `?distributions` will tell you all

1. If `name` is the name of a distribution (e.g., `norm` for "normal"), then

   - dname = the probability *d*ensity (if continuous) or probability mass function of `name` (pdf or pmf), think "histogram"
   - pname = the cumulative *p*robability function (CDF), think "s-curve"
   - qname = the *q*uantile function (inverse to CDF), "think tolerance line"
   - rname = draw *r*andom numbers from `name` (first argument always the number of draws), think whatever you want...it's kind of random

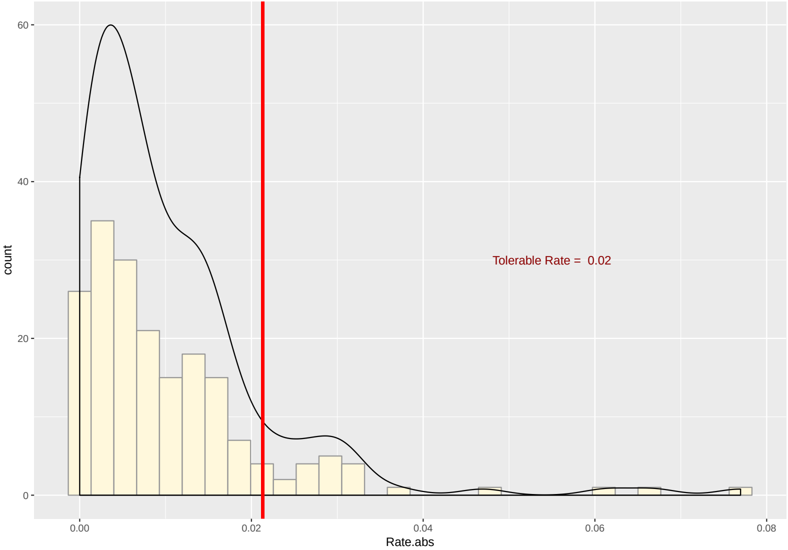
2. And ways to write your own (like the `pareto` distribution we use in finance)

- Suppose the EIUR price series is the *benchmark* in several import contracts you write as the procurement officer of your organization.
- Your concern is with volatility. Thus you think that you need to simulate the size of the price rates, whatever direction they go in.
- Draw the histogram of the absolute value of price rates.

```
require(ggplot2)
r.tol <- quantile(xmprice.r.df$Rate,
    0.95)
r.tol.label <- paste("Tolerable Rate = ",
    round(r.tol, 2))
ggplot(xmprice.r.df, aes(Rate.abs)) +
    geom_histogram(fill = "cornsilk",
        colour = "grey60") + geom_density() +
    geom_vline(xintercept = r.tol, colour = "red",
        size = 1.5) + annotate("text",
    x = 0.055, y = 30, label = r.tol.label,
    colour = "darkred")
```

Thinking...

Result



60
40
20
0
count
Tolerable Rate = 0.02
0.00
0.02
0.04
0.06
0.08
Rate.abs

- A right-skewed, thick-tailed beast for sure...
- Use this function to pull all of the calculations together

```
# r_moments function INPUTS: r vector
# OUTPUTS: list of scalars (mean, sd,
# median, skewness, kurtosis)
data_moments <- function(data) {
    require(moments)
    mean.r <- mean(data)
    sd.r <- sd(data)
    median.r <- median(data)
    skewness.r <- skewness(data)
    kurtosis.r <- kurtosis(data)
    result <- data.frame(mean = mean.r,
        std_dev = sd.r, median = median.r,
        skewness = skewness.r, kurtosis = kurtosis.r)
    # result <- data.frame(result, table
    # = t(result))
    return(result)
}
```


Run this

```
ans <- data_moments(xmprice.r.df$Rate.abs)
ans <- round(ans, 4)
knitr::kable(ans)
```

mean	std_dev	median	skewness	kurtosis
0.0104	0.0113	0.0071	2.681	13.3815

- Right skewed
- Very thick tailed
- We will try the `gamma` and `pareto` functions
- We will make liberal use of the `fitdistr` function
- We will come back to this moments function

Estimate until morale improves...

We will try one method that works often enough in practice...

Method of Moments ("MM" or, more affectionately, "MOM"): Find the distribution parameters such that the moments of the data match the moments of the distribution.

Other Methods

- `fitdistr`: Let the opaque box do the job for you; look at the package `MASS` which uses the "maximum likelihood" approach in the `fitdistr` estimating function (like `lm` for regression).
- `fitdistrplus`: For the more adventurous analyst, this package contains several methods, including MM, to get the job done.

Getting right into it all... suppose we believe that absolute price rates somehow follow a `gamma` distribution. You can look up this distribution easily enough in Wikipedia's good article on the subject.

Behind managerial scenes, we can model the loss with

- A `gamma` severity function – Allows skew and "heavy" tails – Specified by shape, α, and scale, β, parameters
- Especially useful for time-sensitive losses

We can specify these parameters using the mean, μ, and standard deviation, σ of the random severities, X. The scale parameter is

$$\beta = sigma^2/\mu,$$

and shape parameter,

$$\alpha = \mu^2/\sigma^2.$$

The distribution itself is defined as

$$f(x; alpha, \beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)},$$

where,

$$\Gamma(x) = \int_0^{\infty} x^{t-1} e^{-x} dx.$$

Enough of the math,... let's finally implement into `R`.

Load a cost sample and calculate moments and gamma parameters:

```
cost <- read.csv("data/cost.csv")
cost <- cost$x
cost.moments <- data_moments(cost)
cost.mean <- cost.moments$mean
cost.sd <- cost.moments$std_dev
(cost.shape <- cost.mean^2/cost.sd^2)
```

```
## [1] 19.06531
```

```
(cost.scale <- cost.sd^2/cost.mean)
```

```
## [1] 0.5575862
```

```
gamma.start <- c(cost.shape, cost.scale)
```

Using `fitdistr` from the `Mass` package we find:

```
require(MASS)
fit.gamma.cost <- fitdistr(cost, "gamma")
fit.gamma.cost
```

```
##      shape        rate
##   20.2998092    1.9095724
##  ( 2.3729250) ( 0.2259942)
```


How good a job did we do?

- Now construct the ratio of estimates to the standard error of estimates.
- This registers the number of standard deviations away from zero the estimates are.
- If they are "far" enough away from zero, we have reason to reject the null hypothesis that the estimates are no different from zero.

```
(cost.t <- fit.gamma.cost$estimate/fit.gamma.cost$sd)
```

```
##    shape     rate
## 8.554762 8.449652
```

Nice. . . but the scale parameter is `fit.gamma.cost$estimate[2] / gamma.start[2]` times the moment estimates above.

Try this

Use the export-input price series rates and the `t` distribution instead of the `gamma`.

Thinking. . .

Result

Calculate the moments

```
rate <- xmprice.r.df$Rate
rate.moments <- data_moments(rate)
(rate.mean <- rate.moments$mean)
```

```
## [1] 0.001140379
```

```
(rate.sd <- rate.moments$std_dev)
```

```
## [1] 0.01530012
```

Using `fitdistr` from the `Mass` package we find:

```
fit.t.rate <- fitdistr(rate, "t", hessian = TRUE)
```

```
## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in dt((x - m)/s, df, log = TRUE): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced
```

How good a job did we do?

```
##          m          s         df 
##  2.417862 10.502239  4.161013
```

- Nice. . . but that location parameter is a bit low relative to moment estimate.
- What else can we do? Simulate the estimated results and see if, at least, skewness and kurtosis lines up with the moments.

Whitman

SCHOOL of MANAGEMENT

SYRACUSE UNIVERSITY

MBA@SYRACUSE

What have we done?

- . . . Used our newly found ability to write functions
- . . . and built insightful pictures of distributions
- . . . and ran nonlinear (gamma and t-distributions are indeed very nonlinear) regressions
- All to answer critical business questions

The wrap

- Lots more R practice
- Excel look alike processes: Pivot tables and VLOOKUP
- Excel look alike functions
- Graphics to get insights into distributions
- Estimating parameters of distribution
- How good a fit?
- Public data fetches
- ...and why it might all matter: answering critical business questions

To prepare for the live session:

List these:

1. What are the top 3 key learnings for you from this segment?
2. What pieces of this segment are still a mystery?
3. What parts would you like more practice on?
4. Review the assignment. What questions do you have about the assignment for the live session?

Thanks! Till next week...

Whitman
SCHOOL of MANAGEMENT
SYRACUSE UNIVERSITY

MBA@SYRACUSE