UCLAAnderson
SCHOOL of MANAGEMENT

MGMTMFE 431:

Data Analytics and Machine Learning

Course Introduction and Topic 1

Spring 2020

Professor Lars A. Lochstoer

"What information consumes is rather obvious: it consumes the attention of its recipients. Hence, a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it."

- Herbert Simon,
Nobel Laureate in Economics

UCLAAnderson
SCHOOL of MANAGEMENT

A new world

- Unprecedented amount of information available to investors, firm managements, risk officers
  - Tick-by-tick data from lots of different financial markets around the world
  - Easy digital access to financial reports, finance media
  - New data: satellite data, improved and more detailed weather data and forecasts, internet search data, gps and other data from cell phones, Twitter, Facebook, blogs, improved financial transaction data at household level, etc.
- Problem: ***Drowning in information, starving for knowledge***
- People with the skill to analyze and ***extract useful knowledge*** from vast amount of available data in increasing demand

UCLAAnderson
SCHOOL of MANAGEMENT

JP Morgan Data Analytics Report

- See https://news.efinancialcareers.com/us-en/285249/machine-learning-and-big-data-j-p-morgan/

- "Financial services jobs go in and out of fashion. In 2001 equity research for internet companies was all the rage. In 2006, structuring collateralized debt obligations (CDOs) was the thing. In 2010, credit traders were popular. In 2014, compliance professionals were it. In 2017, it's all about machine learning and big data. If you can get in here, your future in finance will be assured." *–efinancialcareers.com*

- JPMorgan issued a comprehensive report on machine learning and data analytics in finance.
  - "Big Data and AI Strategies: Machine Learning and Alternative Data Approach to Investing"

UCLAAnderson
SCHOOL of MANAGEMENT

JP Morgan Report: Main Points

1. Banks will need to hire excellent data scientists who also understand how markets work
2. Machines are best equipped to make trading decisions in the short and medium term
3. An army of people will be need to acquire, clean, and assess the data
4. There are different kinds of machine learning and they are used for different purposes
5. Supervised learning will be used to make trend-based predictions using sample data
6. Unsupervised learning will be used to identify relationships between a large number of variables
7. Deep learning systems will undertake tasks that are hard for people to define but easy to perform
8. Reinforcement learning will be used to choose a successive course of actions to maximize the final reward
9. ***You won't need to be a machine learning expert, you will need to be an excellent quant and an excellent programmer***

- **From report**: If you're only planning to learn one coding language related to machine learning, J.P. Morgan suggests you choose ***R***

UCLAAnderson
SCHOOL of MANAGEMENT

Barclays' report on Machine Learning in Hedge Funds

Summary of main points:

- See: https://news.efinancialcareers.com/us-en/287300/barclays-machine-learning-ai-and-big-data-jobs-in-hedge-funds

1. Not all hedge funds apply these tools just yet (54% are investing in big data, 62% in machine learning)
2. Finding and cleaning data is a major part of the job
3. Data mining and 'black box' nature of machine learning is a big concern
4. Move towards "quantamental" hedge funds. ("Quantamental" is a mix of quant strategies and classic fundamental analysis). Need data specialists that can communicate efficiently with more classically trained analysts and portfolio managers

UCLAAnderson
SCHOOL of MANAGEMENT

Big data in finance

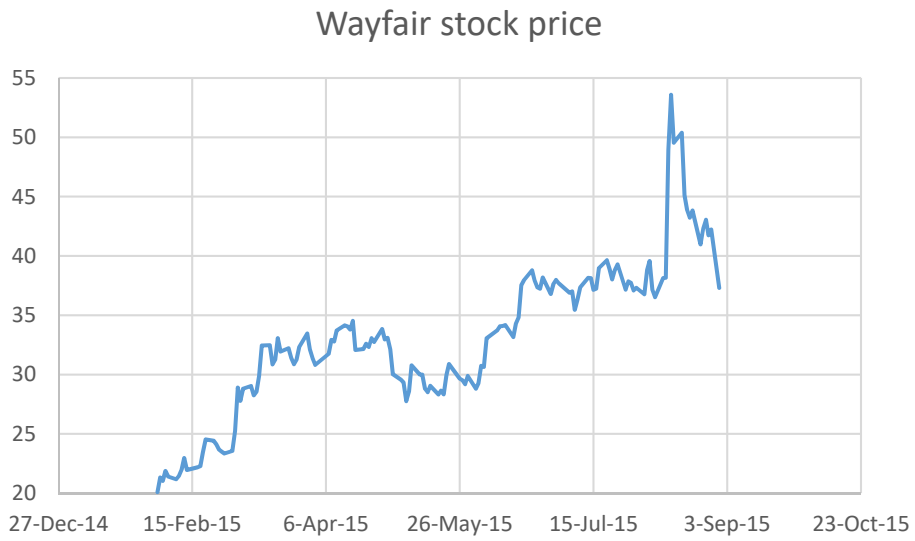
Actors:

- Data providers
  - Specialize in collecting and analyzing specific data (text, satellite, social media, etc.) with applications for investments
- Asset managers (hedge funds)
  - Use alternative data as inputs to improve performance (alpha)
  - E.g., predict returns or covariances, defaults in structured products, etc.
- Banks, credit
  - Assess creditworthiness, typically for household finance
  - E.g., credit card default probability and the design of credit card products
  - Credit card purchase data can also be used to predict consumer trends, potentially even at the macro level

Some examples on next slides…

UCLAAnderson
SCHOOL of MANAGEMENT

Wayfair Inc. and Thinknum



- Thinknum (www.thinknum.com) small non-traditional data provider
  - Social media buzz, satellite images, etc.
  - Surge in downloads from Apple App Store of the Wayfair app with jump in (positive) customer reviews ***preceded*** the spike in stock price by several weeks
  - 20%+ spike in stock price in Aug 2015 was due to release of quarterly report showing high earnings for Wayfair Inc (~$4bn online retailer)

UCLAAnderson
SCHOOL of MANAGEMENT

Orbital Insight

- https://techcrunch.com/2017/05/02/orbital-insight-closes-50m-series-c-led-by-sequoia/
- Uses satellite imagery to analyze economic (and other) activity
- https://orbitalinsight.com/blog/
- E.g., retail demand by tracking retail parking lots:

"Orbital Insight Correctly Predicts Retail Sales Miss. Hit Rate Grows to 78%"

UCLAAnderson
SCHOOL of MANAGEMENT

CargoMetrics

- Quantitative hedge fund that mines global shipping data to track and trade international commodity flows
- About 90 per cent of global trade moves by sea. CargoMetrics uses VHF radio transmissions to track the movements of more than 120,000 ships across the world, monitoring where they dock to gauge their type and size of cargo.
- The data are used to trade commodity prices and, occasionally, currencies and stocks via automated algorithms.

UCLAAnderson
SCHOOL of MANAGEMENT

Eagle Alpha

- The data can also be granular. For example, Eagle Alpha (http://eaglealpha.com/), an alternative information platform, has partnered with a big transport and logistics company that uses individual invoices to construct a contemporaneous gauge of global trade, which allows clients to see export and import patterns for 12 major countries long before the official monthly data are released.

Innovative Asset Managers are integrating Alternative Data into their investment processes.

"Asset managers have got to get much smarter and work out how we can use the vast amount of data out there more effectively. Using data effectively will give you the vital, winning edge."

- Head of Investment of asset manager with $375bn AUM.

"Big data is underappreciated in the sense that its rise represents a watershed moment in the history of investment management. As the traditional line between passive and active has begun to blur with the rise of smart beta strategies, true alpha has proven increasingly elusive. In response, managers across the spectrum from equity to fixed income to impact investing are looking to big data for an edge."

- Asset manager >$1tn AUM.

"Sourcing data to find investment opportunities, such as trawling through credit card and social network data to forecast sales, analysing rail car networks to measure the pulse of the US economy or using weather data to forecast crop yields."

- CIO of asset manager with $30bn AUM.

"How will any fund manager be able to function in the future without adopting at least some aspects of a quant approach? My firm is exploring ways that we might discern investor and insider sentiment from big data, including news, blogs, tweets and product reviews."

- CIO of asset manager with $60bn AUM.

"We believe the Data Revolution is here to stay, and that investors should recognize its potential to reshape the economic landscape. We believe the changes wrought by the Data Revolution will continue to ripple across industries–separating winners from losers, based on those who can best use data as an advantage–including in the world of investment management."

- Asset manager $850bn AUM.

"We can capture every time somebody is saying they bought a new car. We could add those up and can compare that to the stats and be really on the pulse of what's going on with something like auto sales or, similarly, with home sales."

- Co-CEO of asset manager with $150bn AUM.

UCLAAnderson
SCHOOL of MANAGEMENT

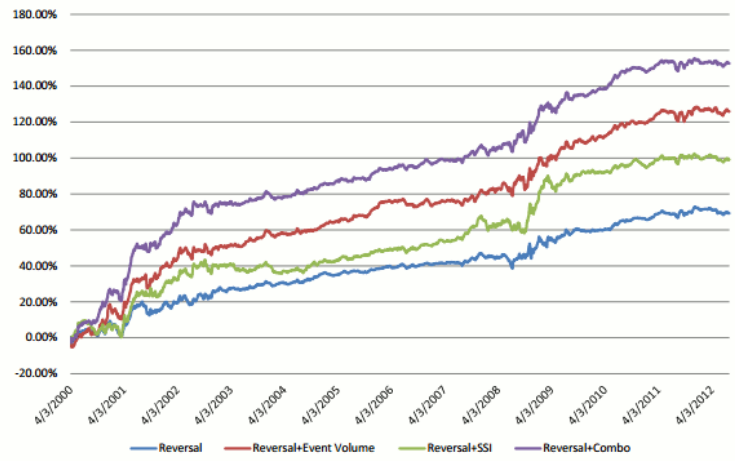
RavenPack

RavenPack

- Textual analysis. Construct 'Sentiment' indexes, quantifying how people currently feel about a company, the economy, a currency, etc.

- https://www.youtube.com/watch?v=zj5fETKF-p0

Fig 5: Cumulative Return of Short-Term Reversal Strategy Enhanced by RavenPack News Analytics-Zero Transaction Cost



This figure shows the cumulative return of the short-term reversal strategy enhanced by RavenPack news analytics without considering the transaction cost. The stock universe includes all S&P 500 index constituents.

SOURCE: RavenPack, 2013

UCLAAnderson
SCHOOL of MANAGEMENT

Quantamental Funds

- Combine classic ***fundamental analysis*** with ***quant methods***

"DYNAMIC DIVERSIFICATION DESIGNED TO PROTECT WHEN IT MATTERS

We are a team of quantamental specialists driven to grow client wealth by utilizing rich data and sophisticated algorithms with the goal of generating superior returns while proactively preserving capital."

DE Shaw moved beyond its roots as a pure quant fund years ago and combines computer-driven analysis with fundamental strategies frrm human stock-pickers, earning it the title 'quantamental.' Eric Schmidt, the former Google chairman who owns a 20% stake in DE Shaw is a fan and reckons this approach will be the future.

UCLAAnderson
SCHOOL of MANAGEMENT

What is this class about?

- Recall problem: ***Drowning in information, starving for knowledge***
- How do you make data ***actionable***?

UCLAAnderson
SCHOOL of MANAGEMENT

Key questions (1)

1. What is the ***quantitative relation*** between data?
   - Say, the relation between textual data from "management discussion" in quarterly reports and investment "alpha," risk management, or corporate strategy (e.g., real investment)

We are trying (typically) to predict an outcome Y.

a. What are the predictive variables X we should use?
b. What is the functional form f(·) relating X to Y? [*Y = f(X) + noise*]

<u>Nomenclature:</u>

X = *predictive variables, attributes, independent variables*

Y = *outcome, observation, response, dependent variable*

UCLAAnderson
SCHOOL of MANAGEMENT

Key questions (2)

2. Is the data ***valuable***? I.e., is it ***incrementally useful*** relative to existing information? For instance, prices contain a lot of information already.

There may be existing, well-known predictors of the outcome you are trying to predict

- To be *economically useful*, a signal must be useful above and beyond existing predictors
- *Incrementally useful*: remains significant when other predictors included in analysis

So, we need to assess marginal contribution

- In linear regression with lots of data, this is easy
- Let *z* be known predictors, *x* be new predictor, test if β is significant in multiple regression:

$$y = \alpha + \beta x + \gamma z + \varepsilon$$

UCLAAnderson
SCHOOL of MANAGEMENT

Key questions (3)

3. ***Model selection***. How do you build workable models that incorporate the enormous amount of new information and are useful for economic decision making?

As number of signals go to infinity, which do you choose? How do they interact (what is the functional form)?

Does it make sense to transform the outcome we are trying to predict? E.g., discretize outcome space, truncate, Winsorize?

The idea underlying a successful model is ***the art and the scarce resource***. Lots of pre-packaged software to help implement once idea behind model specification is determined.

- ***Using economic intuition or economic models can be extremely useful***

UCLAAnderson
SCHOOL of MANAGEMENT

Briefly about me

- Professor in the Finance Area here at UCLA Anderson School of Management
- Previously: Columbia Business School, London Business School, PhD in Finance from UC Berkeley
- Research interests:
  - Investments, financial markets and the macro economy, commodity markets, model uncertainty and learning
- Other:
  - Work experience in Asset Management (institutional investments, tactical allocation, asset-liability management)
  - Currently on Asset Allocation Expert Panel for $1T sovereign wealth fund

UCLAAnderson
SCHOOL of MANAGEMENT

The Course Outline

1) Introductory Concepts

- Examples of data analytics in finance
- Prediction and the value of models
- Data analytics and R, useful tools for data management
- Visualization as a model specification and communication tool
- ggplot2: a useful graphics package in R
- Forecasting signals based on data visualization

2) Big data and panel regressions

- Introduction to panel regressions
- Clustering, fixed effects
- Multiple regressions and marginal effects, omitted variables
- Fama-MacBeth (review): from signal to trading strategy

UCLAAnderson
SCHOOL of MANAGEMENT

The Course Outline

3) Logistic Regressions, Credit Data and Sample Selection

- Models for discrete outcomes
- Default prediction and interest rates using loan/credit cared databases
- Sample selection and propensity scores

4) Machine Learning I: Model selection

- Bayesian shrinkage
- Lasso and ridge regressions
- Sparsity principle
- Cross-validation, best subset
- The data-mining problem in finance

UCLAAnderson
SCHOOL of MANAGEMENT

The Course Outline

5) Unstructured data and textual analysis

- Introduction to text analysis in R
- Mapping unstructured data to numeric signals
- Similarity metrics, investor inattention
- Text as Big Data: 21st century reading of financial reports (web-scraping, EDGAR database)

6) Machine Learning II: Nonlinear methods

- High-dimensional data; model selection and regularization revisited
- Classification: k-means clustering, support vector machines, asymptotic PCA
- Flexible prediction models: Decision trees, bagging (Random Forests), boosting (XGBoost)
- Not covered: Deep learning and neural networks

Student presentations

Some Class Admin

UCLAAnderson
SCHOOL of MANAGEMENT

Classroom Policies

- Please log in to Zoom meeting before class begins
- Please participate, ask questions! Use raise hand feature. I will stop at regular intervals during class and ask if there are questions or comments.
- It is hard to monitor class chat stream. Let's try not to use it. If you have a question or want to let me know about a technical issue that can't wait until I stop and ask for questions, use the 'raise hand' feature or simply turn on your mic and say "Professor!"

UCLAAnderson
SCHOOL of MANAGEMENT

Readings

- Lecture Notes

- Supplementary readings
  - This includes all code used in the slide. The code-snippets will be posted of CCLE.

- Recommended textbook for statistical learning resource:
  - "Introduction to Statistical Learning"
  - http://faculty.marshall.usc.edu/gareth-james/ISL

- Advanced statistical learning resource (not directly relevant for class)
  - "Elements of Statistical Learning"
  - https://web.stanford.edu/~hastie/ElemStatLearn/

- A useful reference for deep learning is: http://www.deeplearningbook.org/

- For most of the R used in this course, I recommend "*R for Everyone*," by Jared P. Lander.
  - Also: "*Machine Learning with R,*" by Brett Lantz

UCLAAnderson
SCHOOL of MANAGEMENT

Class material

- All class material will be posted on class website
  - https://ccle.ucla.edu/course/view/20S-MGMTMFE431-2
  - I will post lecture notes on the day before class and homeworks one week before they are due

UCLAAnderson
SCHOOL of MANAGEMENT

Course Grading

- Problem Sets (6 of 6 graded) 24%
  - Use the pre-assigned MFE groups of four. If a group ends up with 2 students or less since this is an elective, let me know.
- Group project and presentation 25%
  - Presentations will take place in two last classes (Friday May 29th, Monday June 1st.)
- Final exam 51%
  - Date and location: Monday June 8, Take-home, open book

UCLAAnderson
SCHOOL of MANAGEMENT

Teaching Assistant

- Denis Mokanov (denis.mokanov.phd@Anderson.ucla.edu)

- Office hours and location:
  - Zoom meetings as posted on ccle

UCLAAnderson
SCHOOL of MANAGEMENT

Rescheduled classes

- Monday May 25 (Memorial Day) is rescheduled for Friday May 29

UCLAAnderson
SCHOOL of MANAGEMENT

The group projects

- Find an idea for a predicting returns, earnings (sales), yields or similar. Can also try to predict return variances, return covariances, or other metrics relevant to portfolio construction (e.g., skewness, kurtosis).
- Obtain and clean relevant data (whatever you want: 10-K's, news, other)
- Show econometric techniques used (regularizations, decision trees and boosting) and explain why well-suited for problem. Show forecasting results and, if relevant, any trading profits (e.g., Sharpe ratio)
- Present in-class assuming presentation is for the fund management as a pitch to use this new model for a new fund or as an overlay on an existing strategy
- Presentations are to be about 15 mins per group.

UCLAAnderson
SCHOOL of MANAGEMENT

The group projects: example from last year

- Great example of innovative use of big data, machine learning techniques to prediction relevant for investors
  - At the center: a great idea! (nothing substitutes for this….)
  - Group: H. Liu, S. Tan, Y. Chen, and Y. Wang

"Using AirBnB Data to Predict Tourism Industry Return"

airbnb

UCLAAnderson
SCHOOL of MANAGEMENT

Using AirBnB to predict tourism industry return

- When does the tourism industry make money?
  - When people travel, go on holidays
- What do people need to do before traveling?
  - Figure out where they want to go (click to check prices, what-to-do, etc)
  - Make accommodation reservations for the chosen trip
- Airbnb has data on where people search, what areas they look in, how many people are searching + prices offered at different locations
  - Thus: one can use Airbnb search and reservation data to predict tourism services demand!!
  - If we predict demand better than the market, we can make money by buying the tourism index when demand is projected to be high in the future, sell when demand is projected to be low in the future.
  - ***Very nice idea***. Basic, intuitive economics. Good, relevant data.

UCLAAnderson
SCHOOL of MANAGEMENT

Briefly on AirBnB data

- Use Airbnb dataset extracted by AirAPI, a NodeJS wrapper for Airbnb's API endpoints. The dataset contains a random sample of listings from three major markets in US: San Francisco, New York, and Los Angeles, from Jan 1st, 2015 to December 31st, 2015.

- Each record in the dataset is a combination of a listing and a calendar night.

Attributes:

- *Price*: price per night.
- *Listing attributes*: each home on Airbnb is unique on multiple dimensions, such as exact location, size, reviews, decor, etc. Users' willingness to pay may vary with these attributes.
- *Occupancy and availability of the listing*: historical availability with different time ranges.
- *Demand for a listing*: some listings are more popular from search results and generally attracts more views. This set of features represent interest from guests for a listing.
- *Demand and supply within the market*: aggregated features like number of searches and the number of contacts can depict the general demand for a market. The number of available listings represents supply.
- *Demand and supply within a KDT-Room type cluster*: Airbnb clusters the listings that are close geographically, as an automated way to identify neighborhoods.

UCLAAnderson
SCHOOL of MANAGEMENT

1. Observable Features

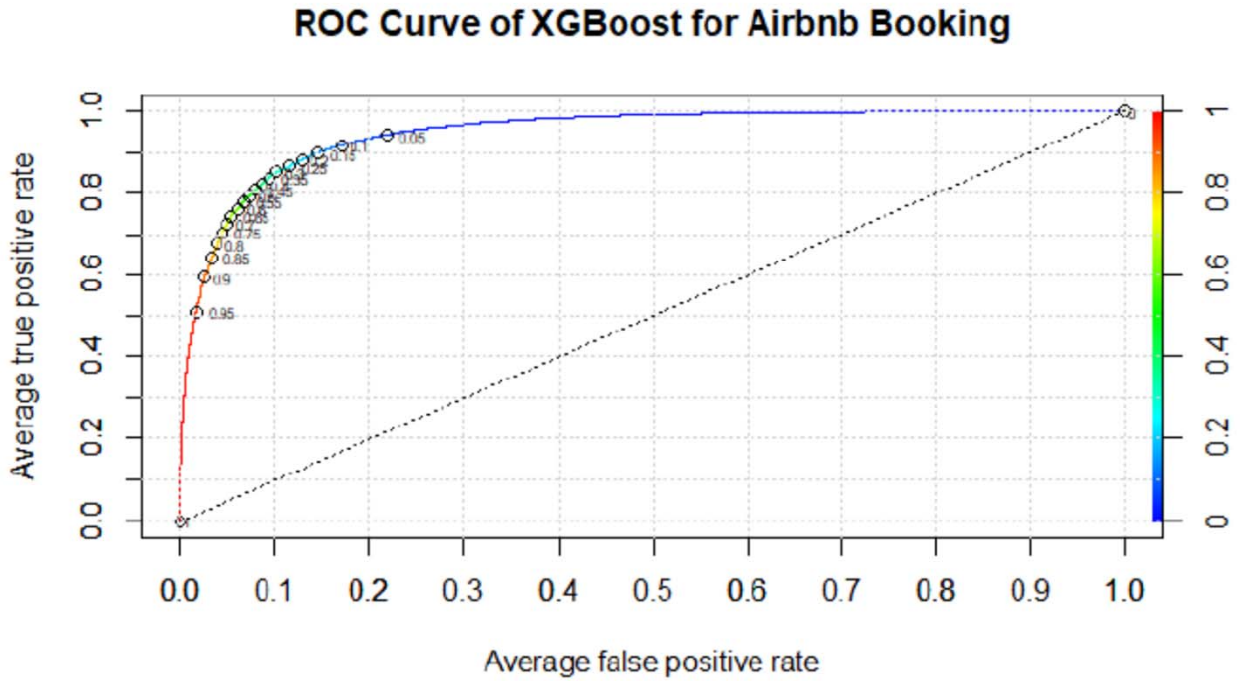
Feature Name	Description
dim_is_requested	True if ds_night receives a booking request eventually; false otherwise.
ds_night	The night on the calendar
ds	The date stamp on which data is collected
id_listing_anon	Anonymized listing ID
id_user_anon	Anonymized user ID of the host of the listing
m_effective_daily_price	Effective daily price on listing calendar in USD
m_pricing_cleaning_fee	Cleaning fee in USD.
dim_market	Market of the listing.
dim_lat	Latitude of the listing.
dim_lng	Longitude of the listing.
dim_room_type	The type of room (Shared room, Private room or Entire home/apt).
dim_person_capacity	Number of people the listings can accomadate
dim_is_instant_bookable	1 if the listing is instant bookable, 0 otherwise.
m_checkouts	Total number of checkouts
m_reviews	Total number of reviews
days_since_last_booking	Number of days since last booking
cancel_policy	Cancellation policy for the listing, coded in integers from 3 to 9
image_quality_score	A score for the image quality
m_total_overall_rating	Number of overall ratings left by guests
m_professional_pictures	Number of professional pictures taken
dim_has_wireless_internet	1 if the listing has wifi, 0 otherwise.
ds_night_day_of_week	0 is Sunday
ds_night_day_of_year	1 is January 1
ds_checkin_gap	Number of days available prior to ds_night, can be 0, 1, ... to 6
ds_checkout_gap	Number of days available post to ds_night, can be 0, 1, ... to 6
occ_occupancy_plus_minus_7_ds_night	Occupancy rate* around the ds_night for +/-7 days. *: Occupancy = Booked/(Booked + Available)
occ_occupancy_plus_minus_14_ds_night	Occupancy rate around the ds_night for +/-14 days.
occ_occupancy_trailing_90_ds	Occupancy rate in the past 90 days prior to ds (included)
m_minimum_nights	Minimum nights required for requesting to book that ds_night
m_maximum_nights	Maximum nights to request book that ds_night
price_booked_most_recent	Daily price in USD in the most recent booking
p2_p3_click_through_score	Historical frequency of clicking on a particular listing from search results.
p3_inquiry_score	Represents historical frequency of someone contacting the listing from the listing page.
listing_m_listing_views_2_6_ds_night_decay	Average listing views in the past 2 days to 6 before ds with decay weights for the ds_night
general_market_m_unique_searchers_0_6_ds_night	Average number of unique searchers in the past 6 days before ds for ds_night
general_market_m_contacts_0_6_ds_night	Average number of unique contacts in the past 6 days before ds for ds_night
general_market_m_reservation_requests_0_6_ds_night	Average number of unique requests in the past 6 days before ds for ds_night
general_market_m_is_booked_0_6_ds_night	Avg number of booked listings in this market for the ds_night 0-6 days before ds
m_available_listings_ds_night	Number of available listings in this market for the ds_night
kdt_score	A score specific for each kdt node
r_kdt_listing_views_0_6_avg_n100	Average of listing views within the kdt node (100 listings) and same room_type listings
r_kdt_n_active_n100	Number of active listings for the same room_type listings within the kdt node
r_kdt_n_available_n100	Number of available listings for the same room_type listings within the kdt node
r_kdt_m_effective_daily_price_n100_p50	The median of effective daily price for the same room_type listings within the kdt node
r_kdt_m_effective_daily_price_available_n100_p50	The median of effective daily price for the same room_type listings that are available within the kdt node
r_kdt_m_effective_daily_price_booked_n100_p50	The median of effective daily price for the same room_type listings that are booked within the kdt node
month_of_year	month(ds_night)
weekday	dummy indicating weekend
m_effective_daily_price_sum	sum of cleaning and room fee
cleaning_fee_rate	m_pricing_cleaning_fee/(m_pricing_cleaning_fee+m_effective_daily_price)
review_rate	m_reviews/(m_reviews+m_checkouts)
click_c	categorical for different p2_p3_click_through_score
inquiry_c	categorical for different p3_inquiry_score
ds_gap	ds_checkin_gap+ds_checkout_gap
ds_gap_c	ifelse(ds_gap>=14,3,ifelse(ds_gap<7,1,2))
price_change_rate	(m_effective_daily_price-price_booked_most_recent)/price_booked_most_recent)
occupancy_marke	general_market_m_reservation_requests_0_6_ds_night/m_available_listings_ds_night
r_kdt_n_available_rate	r_kdt_n_available_n100/r_kdt_n_active_n100
r_kdt_m_effective_daily_price_n100_p50_ratio	m_effective_daily_price/r_kdt_m_effective_daily_price_n100_p50
r_kdt_m_effective_daily_price_available_n100_p50_ratio	m_effective_daily_price/r_kdt_m_effective_daily_price_available_n100_p50
r_kdt_m_effective_daily_price_booked_n100_p50_ratio	m_effective_daily_price/r_kdt_m_effective_daily_price_booked_n100_p50

UCLAAnderson
SCHOOL of MANAGEMENT

Predicting bookings

- Use decision tree framework and XGBoost to predict future bookings

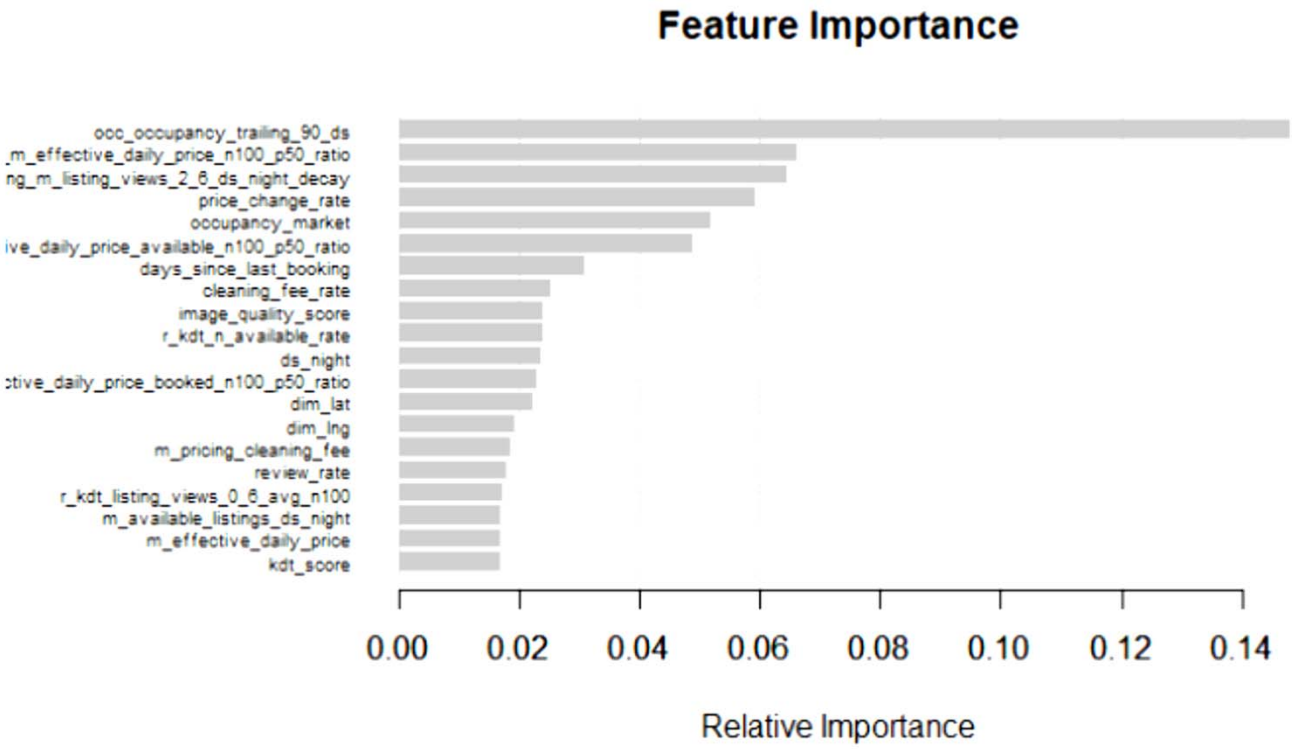
Figure 1. ROC Curve of Airbnb Booking Model



UCLAAnderson
SCHOOL of MANAGEMENT

Further model results

Figure 2. Feature Importance Rank



UCLAAnderson
SCHOOL of MANAGEMENT

Also get industry index returns

2. Components in Dow Jones U.S. Travel & Tourism Index

Company Name	Symbol
Ambassadors Group Inc	EPAX
Avis Budget Group Inc	CAR
Central Japan Ry Co	CJPRY
CTrip.com International Ltd	CTRP
Dollar Thrifty Auto Group Inc	DTG
Dynamic Leisure Corp	DYLI
East Japan Railway Co	EJPRY
Elong Inc	LONG
Expedia Inc	EXPE
Green Globe International In	GGII
Guangshen Railway Co Ltd	GSH
Hertz Global Holdings Inc	HTZ
Interval Leisure Group Inc	IILG
Invicta Group Inc	IVIT
Mtr Corp Ltd	MTRJY
Onelink Corporation	OLNK
Orbitz Worldwide Inc	OWW
Priceline.com Inc	PCLN
Shun Tak Hldgs Ltd	SHTGY
Transnational Automotv Grp I	TAMG
TravelCenters of America LLC	TA
Travelstar Inc	TVLS
Travelzoo Inc	TZOO
Universal Travel Group	UTA
Weikang Bio Tech Group Co In	WKBT
Ytb Intl Inc	YTBLA

UCLAAnderson
SCHOOL of MANAGEMENT

Trading strategy performance

Table 2. Performance of Fama-French Factors and Occupancy Signals

	Mkt.RF	SMB	HML	RMW	CMA	Occupancy
Mean	0.01%	-0.02%	-0.04%	0.00%	-0.04%	0.06%
Standard Deviation	0.0097	0.0048	0.0052	0.0029	0.0030	0.0146
Annualized Sharpe Ratio	0.1436	-0.7351	-1.2482	-0.218	-1.9275	0.6964

- Aggregate results to get aggregate predicted occupancy rate
- Buy if high, sell if low, roll daily

- While the sample is short so the results should be interpreted with caution, the idea and procedure are good!
  - Nice example of type of thinking you should strive for
  - Generally, one could predict demand in other settings, or perhaps how other people will trade, default probabilities, uncertainty (volatility), etc.

UCLAAnderson
SCHOOL of MANAGEMENT

Topic 1

Visualization:
Model Selection and Communication

UCLAAnderson
SCHOOL of MANAGEMENT

Overview

a. Introduction, the value of models
b. Graphics in R
c. ggplot2 introduction
d. Intro to data.table package for large data sets
e. Model specification and visualization
f. Fama-MacBeth regressions and portfolio choice

UCLAAnderson
SCHOOL of MANAGEMENT

a. Introduction

What is Data Analytics?

Tools for constructing models for the purpose of making business decisions

Key Elements:

1. The ability to ***manipulate and transform data*** for model building.
2. ***Visualization*** methods to ***help suggest hypotheses***, clean data, and validate models.
3. A toolkit of models which can be used to predict behavior. We will build on Econometrics and Empirical Methods in Finance which emphasized regression-style models. In particular, we will consider more advanced regression topics and models for discrete data. We will also provide an intro to Bayesian methods, and nonlinear machine learning techniques.
4. Methods to ***evaluate, compare and select models***.
5. The ability to ***communicate to others*** the results of your model-building exercise.

UCLAAnderson
SCHOOL of MANAGEMENT

b. Graphics in R

Visualization of research results and ideas is ***critical for successful communication of your ideas*** and, in fact, also for your own understanding and idea generation.

- *That is: graphics also help **you** choose model specification/technique*

One of the strengths of R is the numerous methods for producing graphics.

Most graphics packages in R take the idea of building up a ***plot as the model***.

That is, we start by defining the plot area, add axes, add points, lines, annotate the plot with labels, titles and legends.

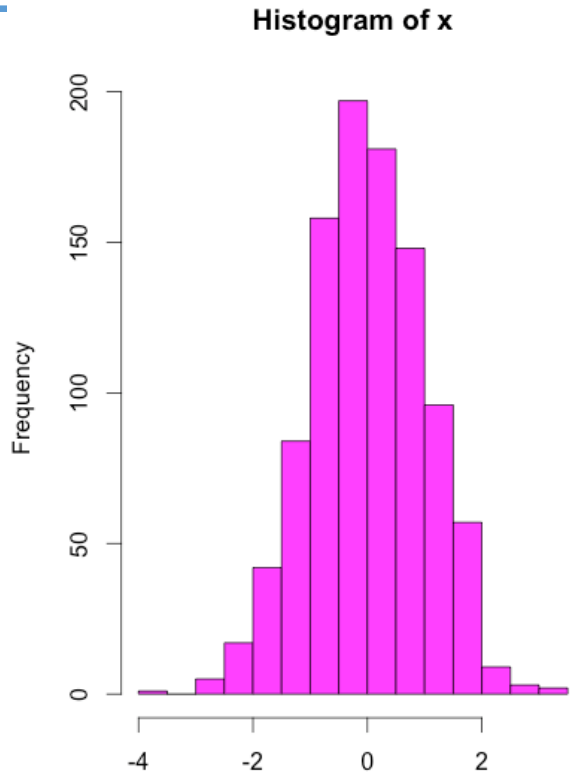
This is certainly the approach of the Base R graphics.

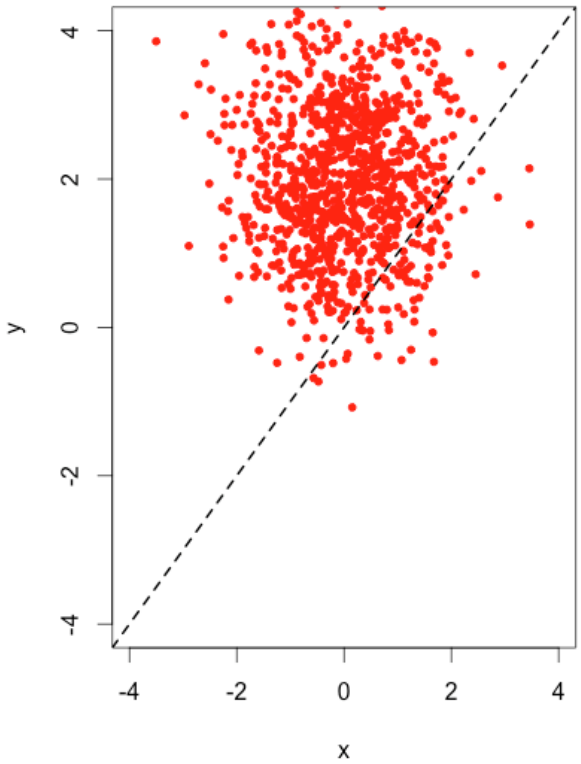
UCLAAnderson
SCHOOL of MANAGEMENT

b. Graphics in R

See Quick-R site for excellent summary of basic graphics. Try

`demo(graphics)`






```
> par(mfrow=c(1,2))    # create a 1x2 array of plots, paint row by row from upper left
> x=rnorm(1000)
> y=rnorm(1000)+2
> hist(x,main="Histogram of x",breaks=20,col="magenta",xlab="")
> plot(x,y,xlab="x", ylab="y",pch=20,col="red",xlim=c(-4,4),ylim=c(-4,4))
> abline(c(0,1),lwd=2,lty=2)
```

UCLAAnderson
SCHOOL of MANAGEMENT

b. Non-Base Graphics

`ggplot2` is the leading graphics package. Elegant plots can be done quickly. Complicated plots can be done efficiently.

Other notable packages:

- `rgl` for 3-D visualization
- `maps` for mapping (U.S centric) but can use map databases.
- `ggmap` for using Google maps with ggplot2
- `OpenStreetMap` has interfaces to many map databases

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

ggplot2 can be used on many levels:

as a simple but very powerful graphics engine via the `qplot()` function

for sophisticated and "picky" users who want to have a high degree of control over the appearance of standard graphical displays of data

for those who want to invent new ways of visualizing data.

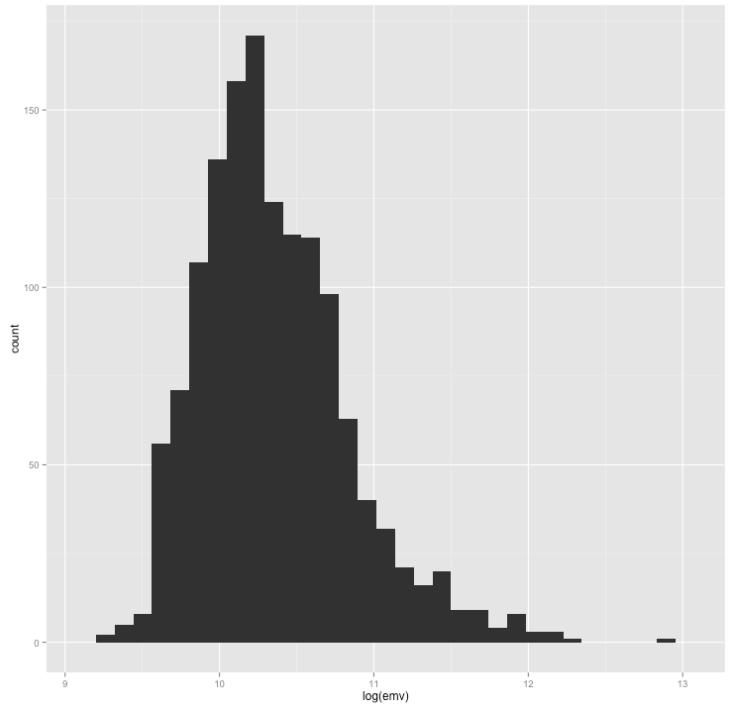
limitations: may be somewhat slow and only 2-D.

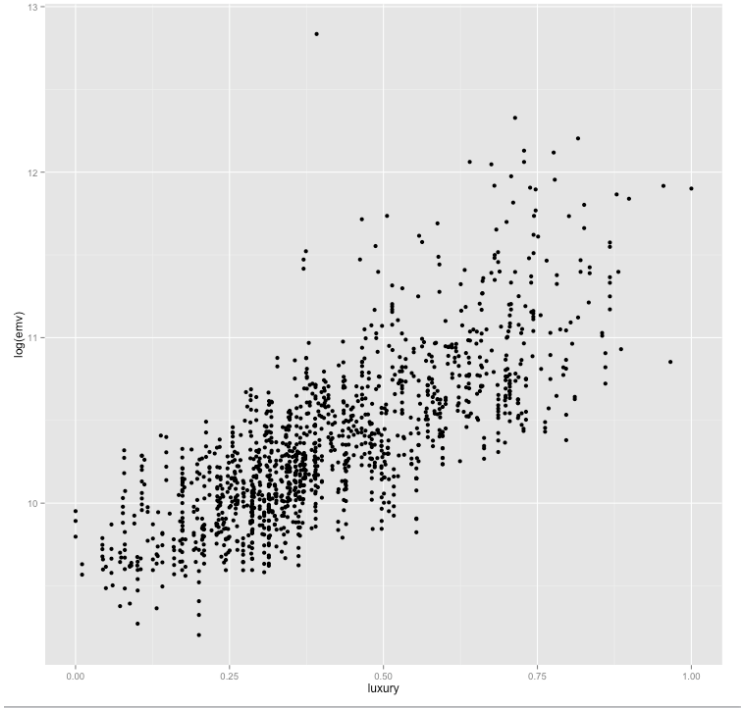
UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

Let's do some simple standard plots:

```
> qplot(log(emv),data=cars)
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
> qplot(luxury,log(emv),data=cars)
```

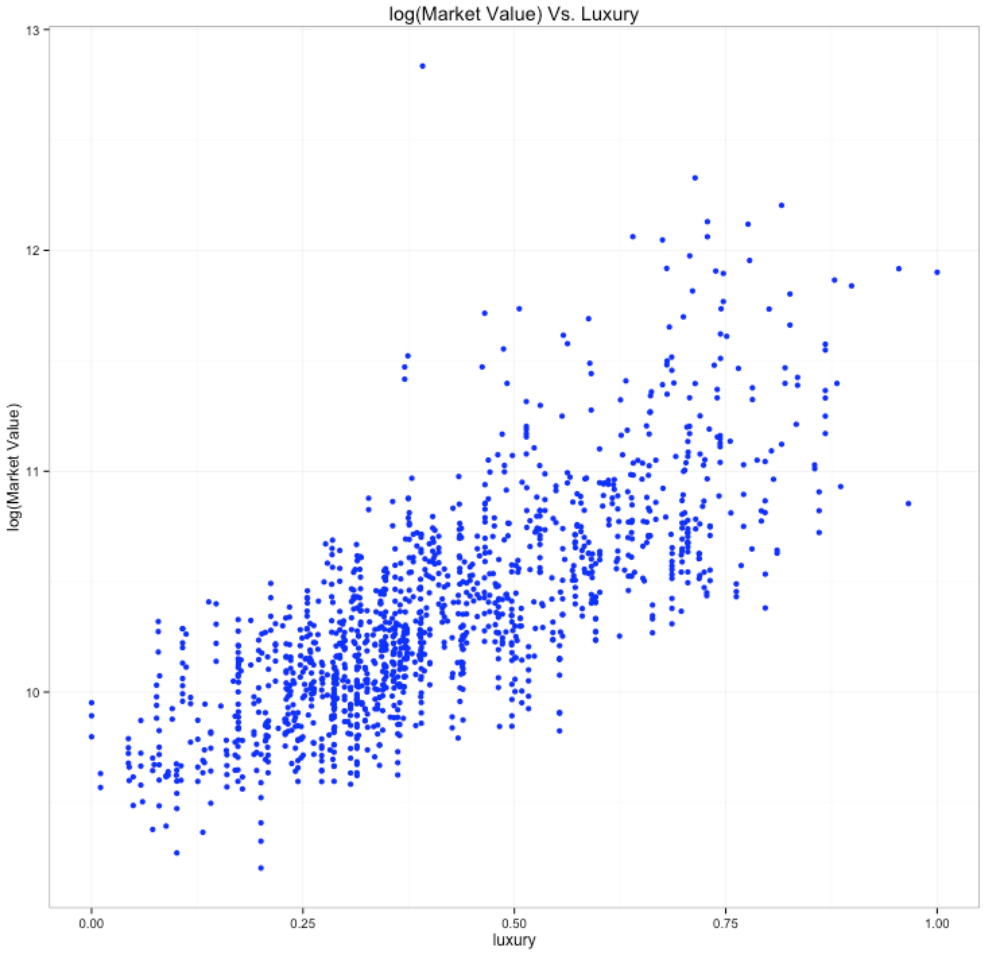




UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

I'm fussy.
Let's put on some labels and control the color of points.
I like a very "clean" look – white background.

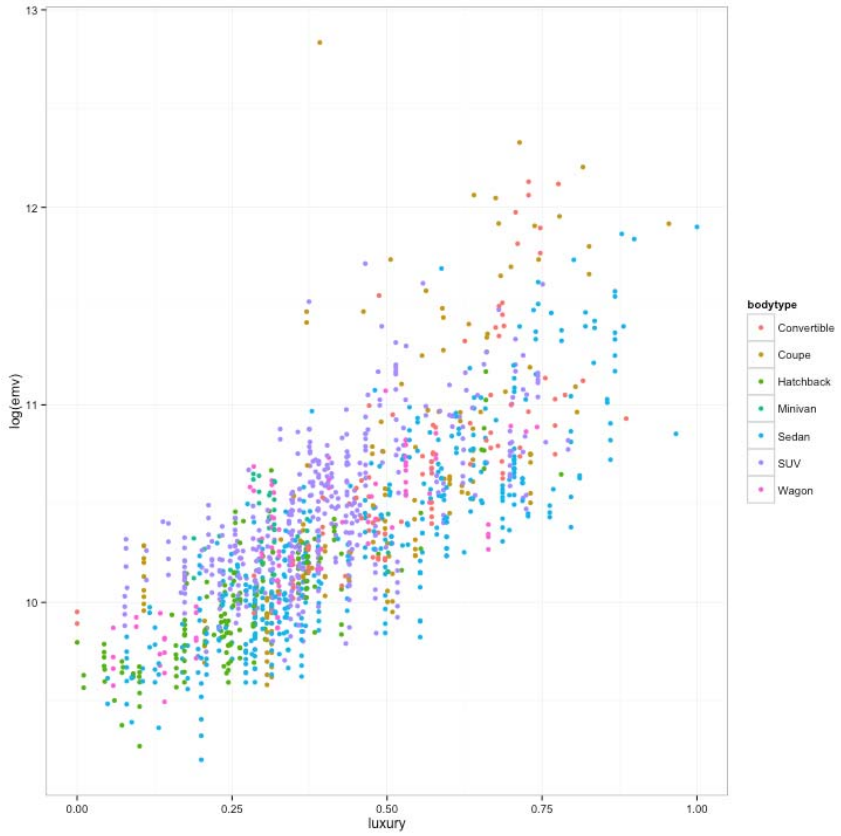



```
> qplot(luxury,log(emv),data=cars,ylab="log(Market Value)",col=I("blue"),
+     main="log(Market Value) Vs. Luxury") + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

Let's try to visualize another variable such as `bodytype` in the same plot.

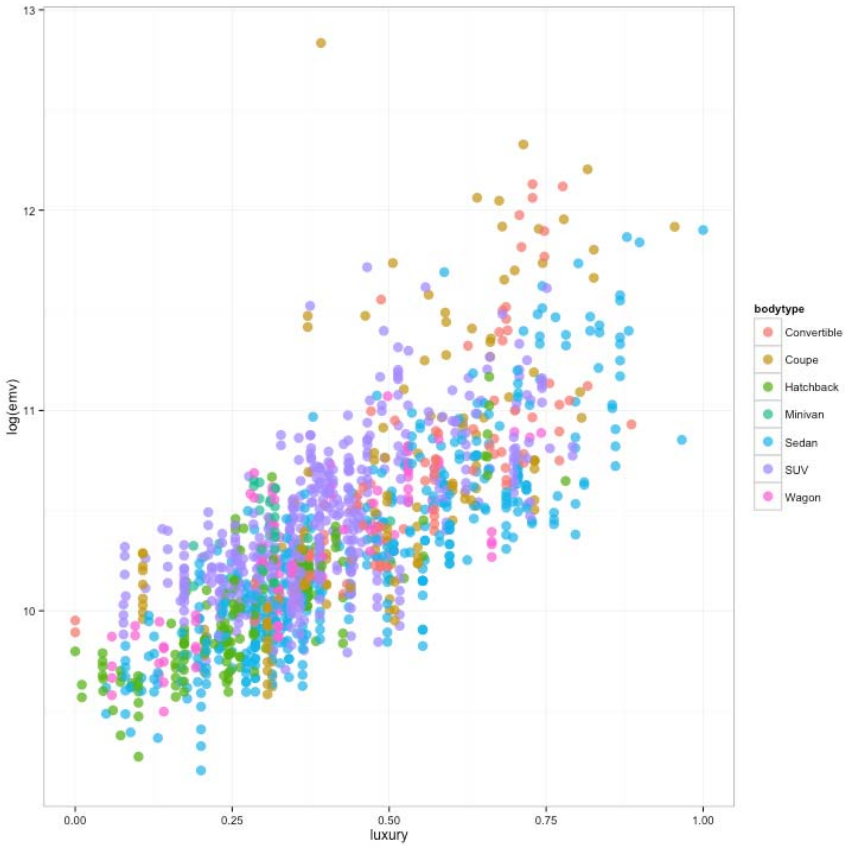



```
> qplot(luxury,log(emv),data=cars,col=bodytype) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

Make dots bigger and a bit transparent.

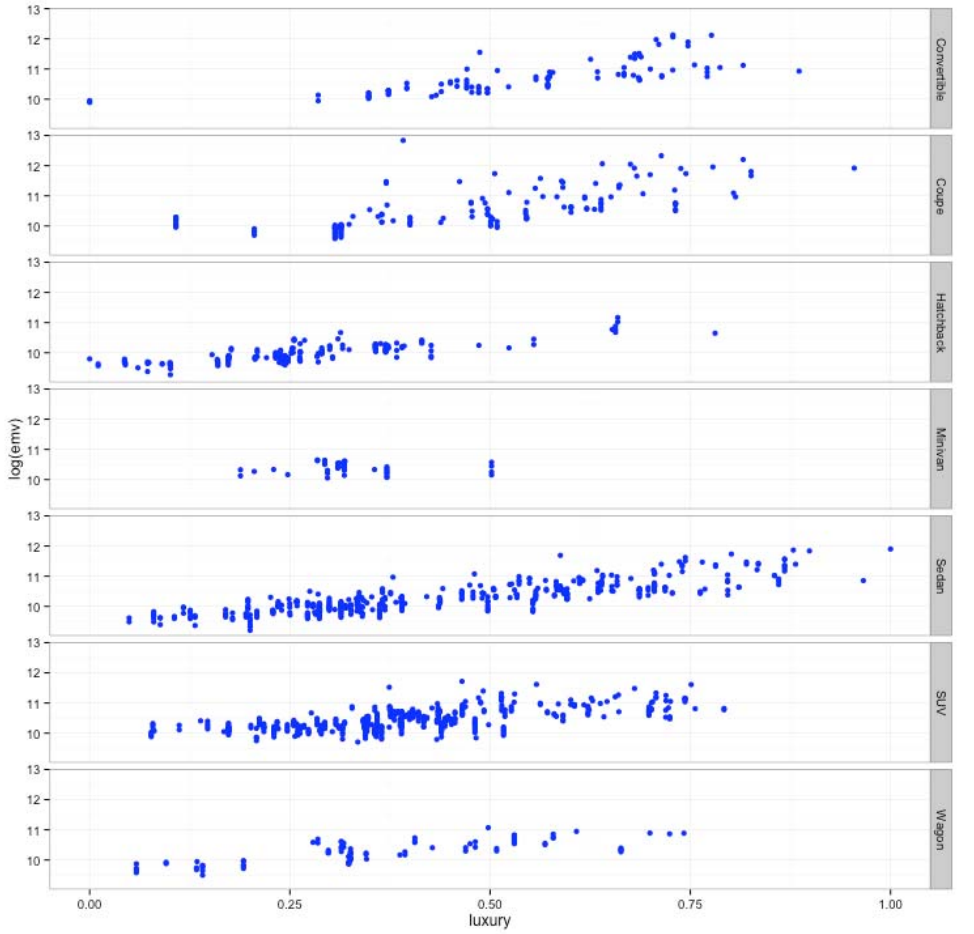



```
> qplot(luxury,log(emv),data=cars,col=bodytype,size=I(4),alpha=I(3/4)) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

Separate plots for each value of `bodytype`.



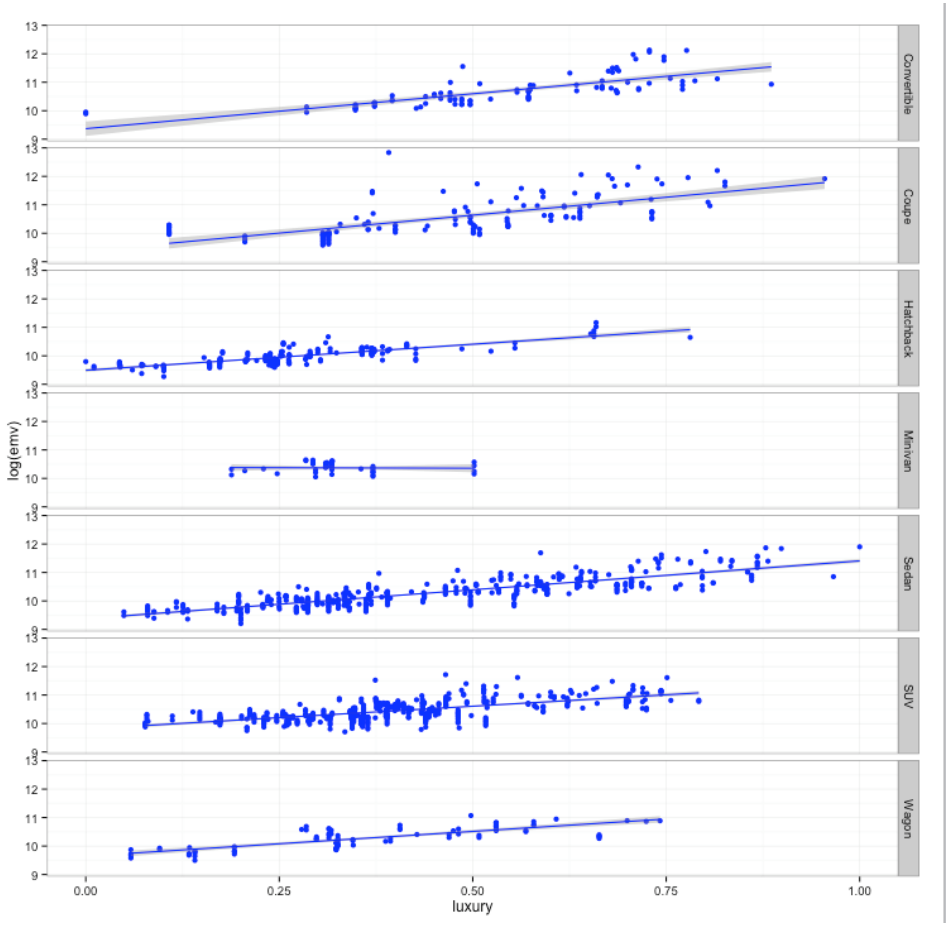

```
> qplot(luxury,log(emv),data=cars,facets=bodytype~.,col=I("blue")) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

Let's put a fitted line on each plot!

If you can do this for all plots, you can certainly do it for one!



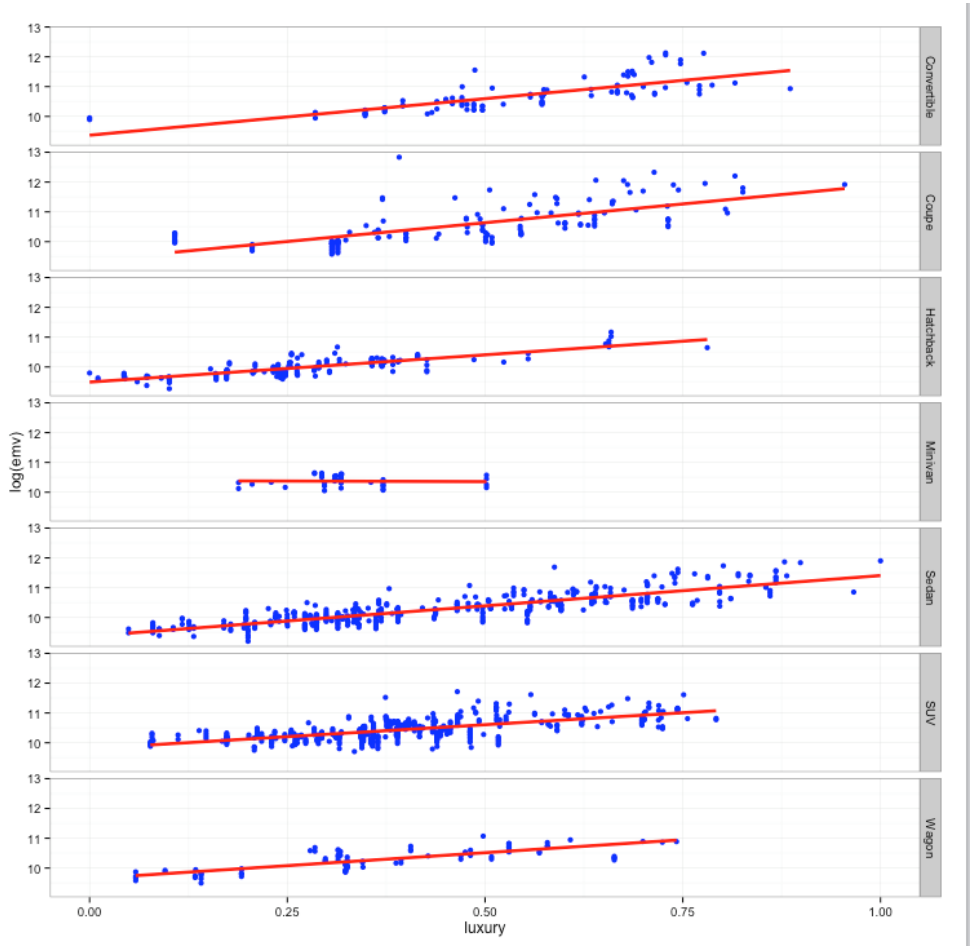

```
> qplot(luxury,log(emv),data=cars,facets=bodytype~.,col=I("blue"),
+       geom=c("point","smooth"),method=lm) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

I'm fussy.
I want to have thicker red lines and get rid of the gray error bands.

`geom_smooth( )` adds lines to the plots. You have control over pretty much everything about those lines!



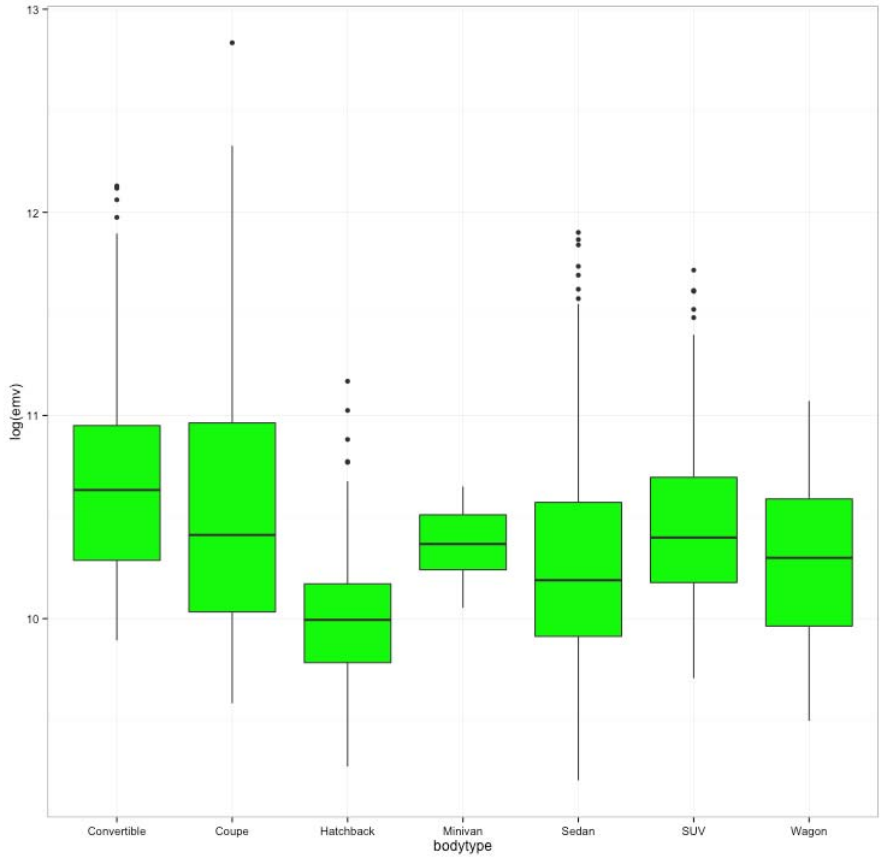

```
> qplot(luxury,log(emv),data=cars,facets=bodytype~.,col=I("blue")) +
+    geom_smooth(method="lm",col=I("red"),size=I(1.2),se=FALSE) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

What about the relationship between continuous variables like market value and categorical or qualitative variables like bodytype?

Plot boxplots of market value for each value of body type. I like mine “green”




```
> qplot(bodytype,log(emv),data=cars,geom="boxplot",fill=I("green")) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to ggplot2

See also ggplot2-CheatSheet.pdf on ccle for an overview of capabilities

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

`data.table` is a package designed to bring advanced capabilities and high speed access for very large datasets. Every data.table object is also a data.frame and can be used with any R function (like lm) which requires data.frames.

`data.table` is used primarily for summary and merging capabilities as well as speed of access. I will illustrate this with a few "toy" dataset examples and then move to a slightly larger example.

`data.table` uses what are called "keys" to access and summarize information. "keys" are variables used to subset and index a file. For example, when merging Vanguard fund data with market returns "date" is a key (see Appendix to these slides)

The reason `data.table` is so fast is that it uses keys to sort the data sets and access by values of the key is then much faster. See: http://datatable.r-forge.r-project.org/datatable-intro.pdf

d. Introduction to data.table

Why data table?

Reduce programming effort

fewer function calls, less repetition

Reduce compute time

faster access to elements

update by reference with no copying

fast “aggregation” or apply functions to subsets of observations

Reduce read-in time

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

Reducing programming time

```
trades[
  filledShares < orderedShares,
  sum( (orderedShares-filledShares)
       * orderPrice / fx ),
  by = "date,region,algo"
]
```

R :	i	j	by
SQL :	WHERE	SELECT	GROUP BY

d. Introduction to data.table

Reducing compute time

e.g. 10 million rows x 3 columns x,y,v 230MB

```
DF[DF$x=="R" & DF$y==123,]   # 8     s
DT[.("R",123)]               # 0.008s

tapply(DF$v,DF$x,sum)        # 22    s
DT[,sum(v),by=x]             #  0.83s
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

Let's start by creating a small data table that resembles many marketing data sets. We have data on sales of products and each product is identified according to which "category" or group of products there are.

For packaged goods products, Nielsen is the primary source of data.

Each specific brand, size, and packaging is specified by a UPC code (a 12 or 13 digit number). UPCs are organized into product modules.

For example, a specific brand and size and flavor of toothpaste (say Crest, mint flavored, 4 oz) has a specific UPC number. All toothpastes are in the oral care "module" or category.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

Create table and set the “keys” to be module and upc. This means we can access the table very efficiently using these keys.

```
> setwd("~/workspace/class/237M1/lecture notes")
> sales.table = data.table(module=c("a","a","c","c","c","b"),upc=c(2,3,4,5,5,1),
+                          sales=round(runif(6)*10000,0))
> sales.table
   module upc sales
1:      a   2  6995
2:      a   3  1275
3:      c   4  1326
4:      c   5  6009
5:      c   5  5927
6:      b   1  5179
> setkey(sales.table,module,upc)  # makes module and upc keys and "sorts" in place
> # show all data.tables in the workspace
> #
> tables()
     NAME         NROW MB COLS              KEY
[1,] sales.table    6 1  module,upc,sales module,upc
Total: 1MB
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

data.table syntax: `dt[i,j,by]`

"take data table **dt**, subset rows using **i**, then calculate j grouped by **by**"

Accessing rows in a data table is pretty much the same as a data.frame except that you can use values of the key variables to access rows.

```
> sales.table[1:3,]
   module upc sales
1:      a   2  6995
2:      a   3  1275
3:      b   1  5179
> sales.table[upc > 3,]
   module upc sales
1:      c   4  1326
2:      c   5  6009
3:      c   5  5927
> # note you don't have to say "sales.table$upc" in the logical expression
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

We can also use the key variables to select observations in data table. Here we can refer directly to the keys without adding the ",".

```
> sales.table["a"]
   module upc sales
1:      a   2  6995
2:      a   3  1275
> sales.table[.(c("a","c"))]
   module upc sales
1:      a   2  6995
2:      a   3  1275
3:      c   4  1326
4:      c   5  6009
5:      c   5  5927
```

You can also use `!.(c("a","c"))` to select what is not in the list.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

To refer to columns or “variables” in data.table, we can use the same “$” syntax as before but we can’t use numerical values

```
> sales.table$module
[1] "a" "a" "c" "c" "c" "b"
> #
> # or use the name of the variable in the "j" position
> sales.table[,list(module,upc)]
   module upc
1:      a   2
2:      a   3
3:      c   4
4:      c   5
5:      c   5
6:      b   1
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

Lets create a new variable. Note anything that creates a column or variable goes on the right hand side of the ",". The "creation" operator is ":="

```
> sales.table[,lnsales:=log(sales)]
> sales.table
   module upc sales  lnsales
1:      a   2  6057 8.708970
2:      a   3  3949 8.281218
3:      b   1  1706 7.441907
4:      c   4  5512 8.614683
5:      c   5  4629 8.440096
6:      c   5  6550 8.787220
```

If we say `sales.table[,sales:=NULL]`, we will erase a variable.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

But most people want to do some sort of operation on each subset such as to add-up or aggregate data for each module. This shows the strength of data.table for a simple and powerful syntax.

```
> sales.module=sales.table[,sum(sales),by=module]
> setnames(sales.module,"V1","agg_sales")
> sales.module
   module agg_sales
1:      a     10006
2:      b      1706
3:      c     16691
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

Now we have total sales for each module but we want to refer to the modules not by the key value but we want to add a module description.

We create a module description and do a "outer" join using data.table

```
> module.description = data.table(module=c("a","b","c"),
+                                  description=c("meats","milk","cereal"))
> setkey(module.description,module)
> sales.module[module.description]
   module agg_sales description
1:      a     10006       meats
2:      b      1706        milk
3:      c     16691      cereal
```

Here the `sales.module[module.description]` is doing the join!

Note both data.tables (the "X" or left table (`sales.module`)) and the "Y" or right table(`module.description`)) must have common keys.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to data.table

We can also use “merge.”

```
> # or
> merge(sales.module,module.description,by="module")
   module agg_sales description
1:      a     10006        meats
2:      b      1706         milk
3:      c     16691      cereal
>
> # or
>
> merge(sales.module,module.description)
   module agg_sales description
1:      a     10006        meats
2:      b      1706         milk
3:      c     16691      cereal
```

UCLAAnderson
SCHOOL of MANAGEMENT

Visualization and Model Building

e. Model specification and visualization

f. Fama-MacBeth regressions and portfolio choice

UCLAAnderson
SCHOOL of MANAGEMENT

e. Model specification and visualization

Graphics can help guide model specification

- Recall: Y = **f(X)** + least possible noise

We will now consider the problem of estimating expected returns on various trading strategies using a large, unbalanced panel dataset

Stata-file StockRetAcct.dta

- Contains data on annual stock returns (Y) for all listed U.S. firms from 1981 to 2015 (except smallest 20% of firms (micro-firms)), in addition to accounting variables (X) thought to be relevant for predicting returns.
- We will load this dataset into a data.table and use visualization techniques to suggest a model for expected returns

UCLAAnderson
SCHOOL of MANAGEMENT

e. About StockRetAcct-dataset

70756 by 16 = 1,132,096 data entries

- FirmID (key)
- year (key)
- lnAnnRet (log annual firm stock return from June of 'year' to June of 'year'+1, delisting events included)
- lnRf (log nominal 1-yr T-bill rate from June of 'year' to June of 'year'+1)
- MEwt (a variable proportional to market value of firm as of June of 'year')
- lnIssue (last 36 month stock issuance as of June of 'year')
- lnMom (12-2 month Momentum as of June of 'year')
- lnME (log market size as of June of 'year')
- lnProf (profitability (log of 1+revenue minus cost of goods sold divided by total book value))
- lnEP (log of earnings to price)
- lnInv (log of 1+total asset growth)
- lnLever (log of leverage)
- lnROE (log of 1+return on equity)
- rv (last year's realized variance using daily data)
- lnBM (log book-to-market)
- ff_ind (Fama-French 12 industries, classification variable)

- All accounting variables are taken from the last annual report, conditional on it being released at least 6 month earlier
  - This lag is to ensure it is public information
  - All accounting variables as well as lnME are winsorized at the 1%/99% level cross-sectionally each year

UCLAAnderson
SCHOOL of MANAGEMENT

e. Data Analysis using data.table

```
# set your working directory, where you have downloaded the Stata data file
> # change the below to match your folder
> setwd("D:/llochsto/Dropbox/Data Analytics/Data")
>
> # we need the foreign package to import data in different format
> require(foreign)
> require(data.table)
> require(ggplot2)
>
> # Download data and set as data.table
> StockRetAcct_DT <- as.data.table(read.dta("StockRetAcct_insample.dta"))
>
> # take a look at content, first 6 rows. Data is annual.
> head(StockRetAcct_DT)
```

	FirmID	year	lnAnnRet	lnRf	MEwt	lnIssue	lnMom	lnME	lnProf	lnEP	lnInv
1:	6	1980	0.3636313	0.07894428	2.814308e-04	0.03134417	0.07535515	12.58147	0.2017671	0.14641121	0.09362611
2:	6	1981	-0.2904088	0.13019902	3.214631e-04	0.04421350	0.51265192	12.90800	0.2156609	0.10255504	0.08724214
3:	6	1982	0.1866300	0.13070259	2.663127e-04	-0.06819496	-0.22050542	12.55777	0.1840875	0.11954752	0.11166344
4:	6	1983	0.4898190	0.08983046	1.699149e-04	-0.07177968	0.04621762	12.56195	0.1655312	0.11592383	-0.03311720
5:	10	1991	-0.5080047	0.06121579	3.269729e-05	0.11520413	1.34105313	11.56583	0.2397878	0.02314729	0.30005118
6:	12	2000	-1.3568472	0.06197736	1.219181e-05	0.16523764	0.25174579	12.27575	-0.3823268	-0.02378274	-0.17460629

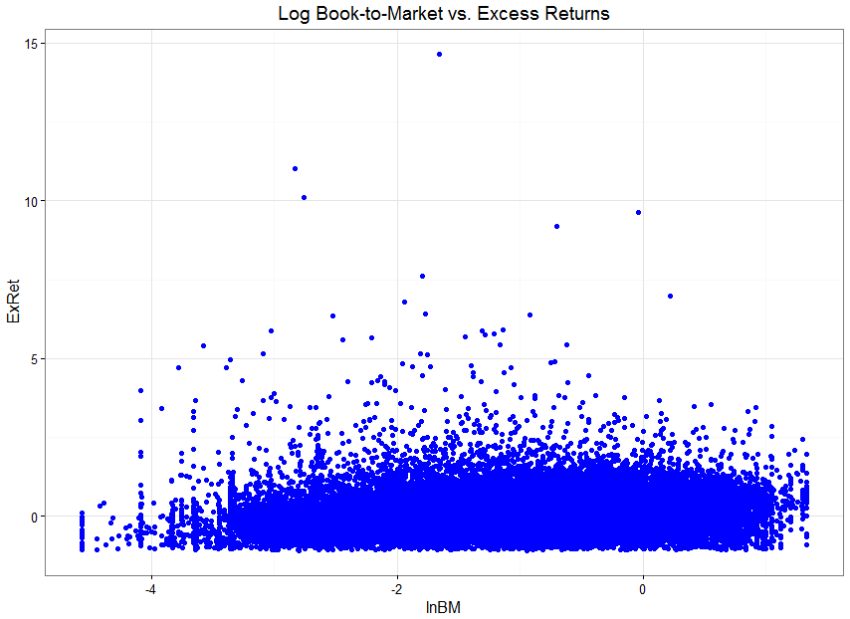
	lnLever	lnROE	rv	lnBM	ff_ind
1:	0.6960014	0.09529421	0.08413413	0.6333913	3
2:	0.7098430	0.08217967	0.05638131	0.3567226	3
3:	0.7309716	0.07951558	0.06207170	0.7794052	3
4:	0.7108847	0.05537406	0.07695480	0.7021134	3
5:	0.4187644	0.14682838	0.37436786	-2.1609421	10
6:	0.8244712	-0.59177256	1.06719565	-3.8155227	6

UCLAAnderson
SCHOOL of MANAGEMENT

e. Consider the "Value vs. Growth" signal: lnBM

Setting keys and basic plots

```
> # set keys for StockRetAcct_DT In particular, it will be useful to sort on FirmID and year
> setkey(StockRetAcct_DT, FirmID, year)
>
> # create excess returns (what we really care about)
> StockRetAcct_DT[,ExRet:=exp(lnAnnRet) - exp(lnRf)]
>
> # scatter plot excess returns vs. lagged bm
> # (notice that all firm characteristics are already lagged so we don't need to deal with this)
> qplot(lnBM, ExRet,data = StockRetAcct_DT, col=I("blue"), main = "Log Book-to-Market vs. Excess Returns") + theme_bw()
Warning message:
Removed 3011 rows containing missing values (geom_point).
```



The plot is a bit of a mess

- Lots of noise, typical for return data
- Lots of outliers, too
- Do you see a relation between lagged b/m and subsequent years' return??

How can we make it more informative?

UCLAAnderson
SCHOOL of MANAGEMENT

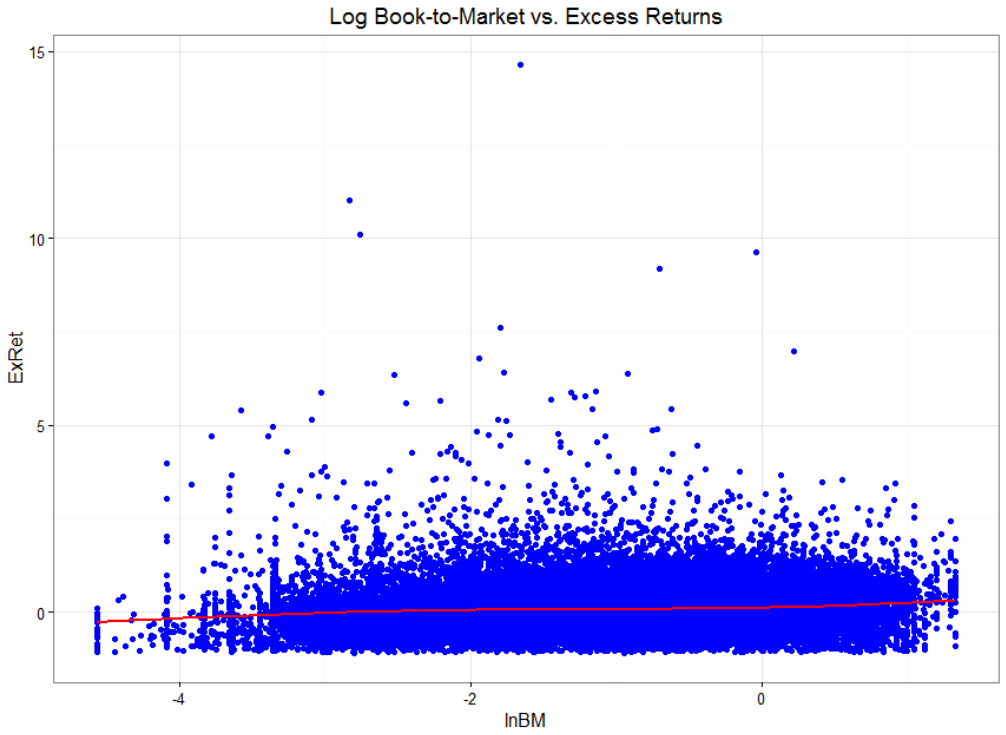
e. Consider the "Value vs. Growth" signal: lnBM

Let's fit a red smooth (possibly nonlinear) line using geom_smooth

```
# Let's fit a curve to this. Exclude missing observations for this to avoid error message: \
> qplot(lnBM, ExRet,data = StockRetAcct_DT[!is.na(StockRetAcct_DT$lnBM)], col=I("blue"),
+ main = "Log Book-to-Market vs. Excess Returns") + geom_smooth(col=I("red"),se=FALSE,na.rm=TRUE) + theme_bw()
```

Can see slight positive relation

- Let's zoom in
- Let's allow more nonlinear curve

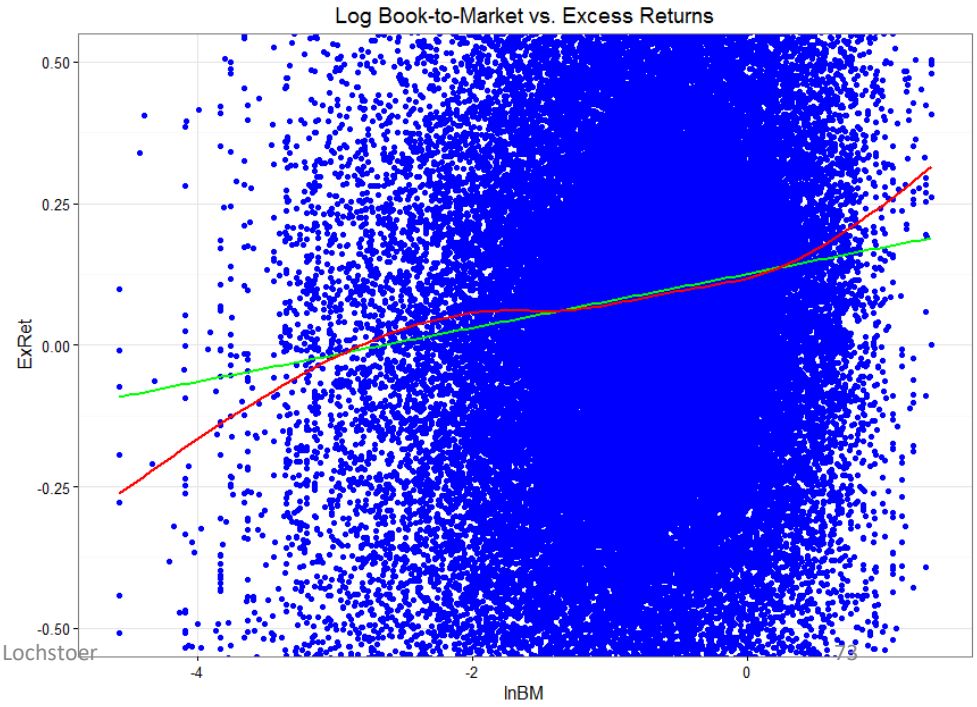


UCLAAnderson
SCHOOL of MANAGEMENT

e. Expected Return and lnBM, zooming in

- Using coord_cartesian() allow us to zoom without changing data used for curve fitting
  - xlim() and ylim() limits both axis and the data
- The geom_smooth() option span allows us to choose tighter kernel when fitting line, thus allowing more nonlinearity, span sets the fraction of points used for each local fit.
- Let's also add a linear fit in green

```
qplot(lnBM, ExRet,data = StockRetAcct_DT[!is.na(StockRetAcct_DT$lnBM)],
+ col=I("blue"), main = "Log Book-to-Market vs. Excess Returns")
+ geom_smooth(col=I("green"),method="lm",se=FALSE,na.rm=TRUE)
+ geom_smooth(col=I("red"),se=FALSE,na.rm=TRUE, span = 0.3)
+ theme_bw() + coord_cartesian(ylim=c(-0.5, 0.5))
```



- *Somewhat nonlinear pattern*
- *Large quantities, expected excess return from -25% to 30%!*
- *Overall, huge valuation effect on "risk premiums"*
- *Still, lots of noise/risk; not a very convincing-looking relation*

UCLAAnderson
SCHOOL of MANAGEMENT

e. Stop and think…

- Let's take a step back. This was fun, but what have we really learned?
  - The results seem to indicate one should be a value investor (at least in terms of expected returns), i.e. one should buy value stocks and short growth stocks

- However, the pattern in the graph could be due to…
  1. … movements in market risk premium resulting in all B/M ratios moving up or down together and thus not a cross-sectional phenomenon
  2. … outliers, only a few stocks and not a pervasive pattern, spurious nonlinearities
  3. … transitory price movements in small, illiquid stocks and thus not tradeable (if you tried, the price would move against you)

- Let's do something graphically that is much more convincing and, in fact, the basis for many empirical asset pricing papers
  - Create ***implementable trading strategies*** with large, diversified portfolios
    - Reduce noise by sorting into portfolios, a good idea when lots of noise is idiosyncratic
  - Think of these as ***mutual funds***
  - Show average return to such mutual funds
  - data.table is able to do this very quickly

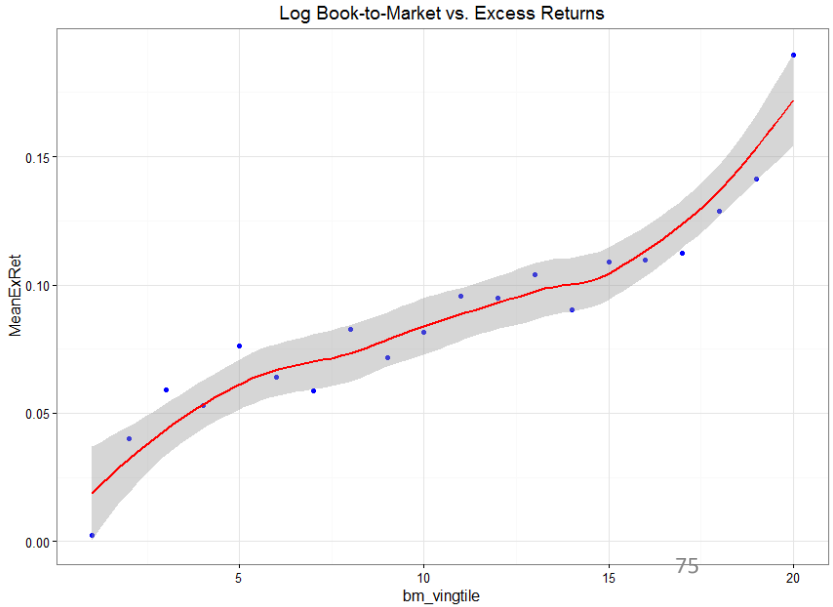
UCLAAnderson
SCHOOL of MANAGEMENT

e. Book-to-market sorted portfolios

- Let's start by considering a flexible step function for predicting returns by categorizing book-to-market ratios in discrete bins.
  - A portfolio is then all the stocks with b/m value in a particular bin
  - We'll do vingtile sort (20 bins based on the 5, 10, …, 90, and 95 percentile breakpoints).

```
> # First, define the categorical variable from 1 - 20 of which B/M vingtile a stock belongs to
> StockRetAcct_DT[,bm_vingtile:=cut(StockRetAcct_DT$lnBM,breaks=quantile(StockRetAcct_DT$lnBM,probs=c(0:20)/20,na.rm=TRUE), labels=FALSE)]
>
> # get the mean excess return by vingtile
> EW_BM_MutualFunds <- StockRetAcct_DT[,list(MeanExRet = mean(ExRet)), by = bm_vingtile]
>
> # Now, plot based on the vingtile variable. This is the average excess return in the same data period.
> qplot(bm_vingtile, MeanExRet,data = EW_BM_MutualFunds, col=I("blue"), na.rm = TRUE, main = "Log Book-to-Market vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
```

- Much better, but still this isn't a tradeable strategy
- The bins are defined on the whole sample, i.e. using forward-looking information.
- Thus, there may be years where no stocks are in the extreme bins.
- Finally, note that we are implicitly weighting later years more since mean is taken across all firms and years and since there are many more firms in the late 1990s than early 1980s.



UCLAAnderson
SCHOOL of MANAGEMENT

e. Tradeable book-to-market portfolios

- Create bins based on the current year when predicting next year's returns
  - This is what you have to do in real-time. No use of information from the future!

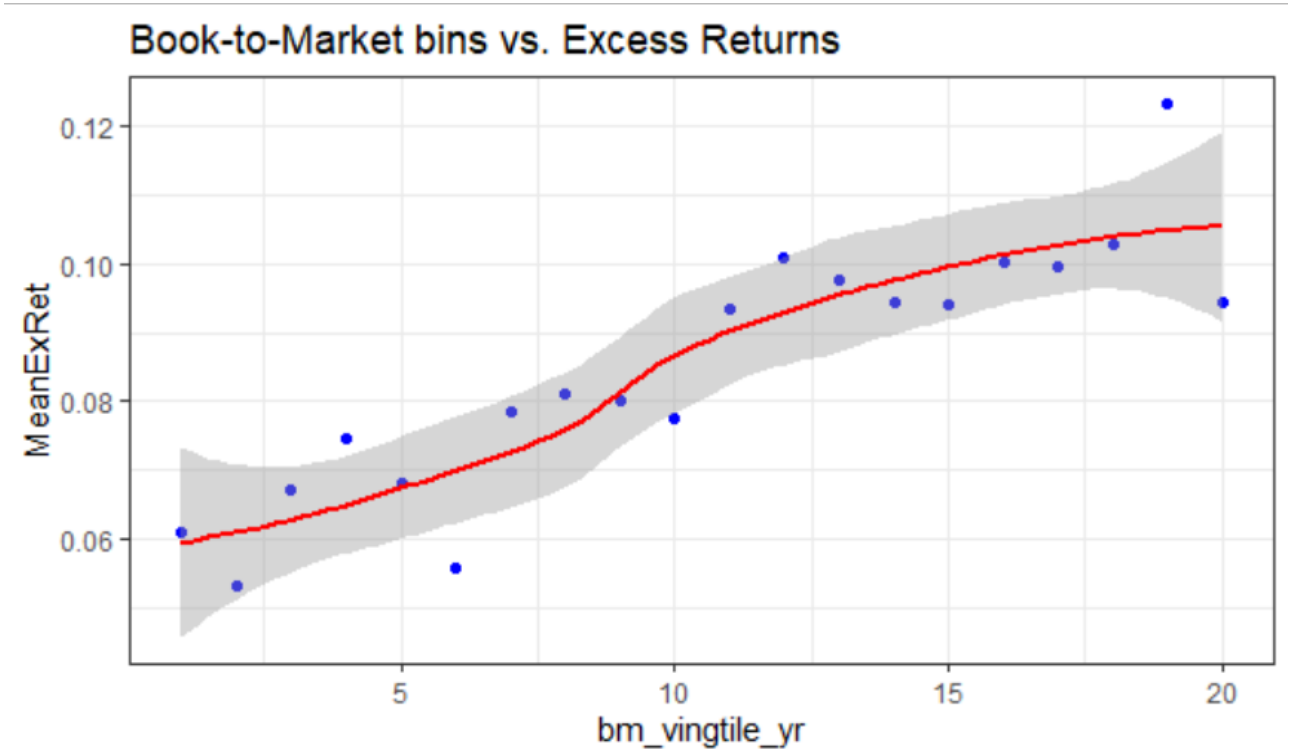
```
> # due to winsorizing of original data, we add a tiny amount of noise (jitter) to lnBM before creating annual vingtiles
> # this is to avoid ties in the quantile sorts
> StockRetAcct_DT[,lnBM:=jitter(lnBM, amount = 0)]
>
> # loop through the years in the data base
> for (i in 1981:2014)
 + {
   + StockRetAcct_DT[year == i,bm_vingtile_yr:=cut(StockRetAcct_DT[year == i,]$lnBM,
    breaks=quantile(StockRetAcct_DT[year == i,]$lnBM,probs=c(0:20)/20,na.rm=TRUE), include.lowest=TRUE, labels=FALSE)]
 + }
>
> # now, we need to take the mean in a clever way to account for the changing number of firms
> # first take mean by year and vingtile
> EW_BM_MutualFunds_yr <- StockRetAcct_DT[,list(MeanExRetYr = mean(ExRet)), by = list(bm_vingtile_yr, year)]
> # then average across years
> EW_BM_MutualFunds_yr <- EW_BM_MutualFunds_yr[,list(MeanExRet = mean(MeanExRetYr)), by = bm_vingtile_yr]
> qplot(bm_vingtile_yr, MeanExRet,data = EW_BM_MutualFunds_yr, col=I("blue"), na.rm = TRUE,
   + main = "Book-to-Market bins vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

e. Tradeable book-to-market portfolios

- These are actual fund returns from 1981-2015.
  - Caveat: we are ignoring transaction costs and price impact
  - This may be important, especially for smaller stocks

- This is now a cross-sectional sort, verifying that the value vs growth phenomenon (also) holds in the cross-section of stocks
  - Notice that the relation is pretty linear



UCLAAnderson
SCHOOL of MANAGEMENT

e. Nonlinearity: Small vs. big

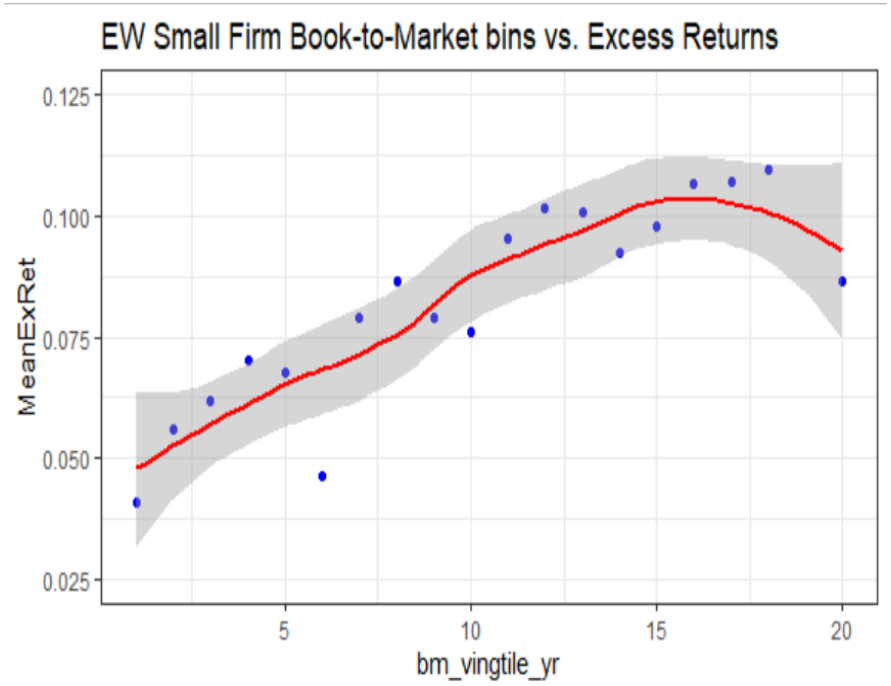
- While the relation looked linear, we have done a lot of averaging across stocks
  - Such averaging can hide important patterns.
  - Let's create conditional sorts, conditioning on whether the firm is one of the 500 biggest firms (large) or not (small)

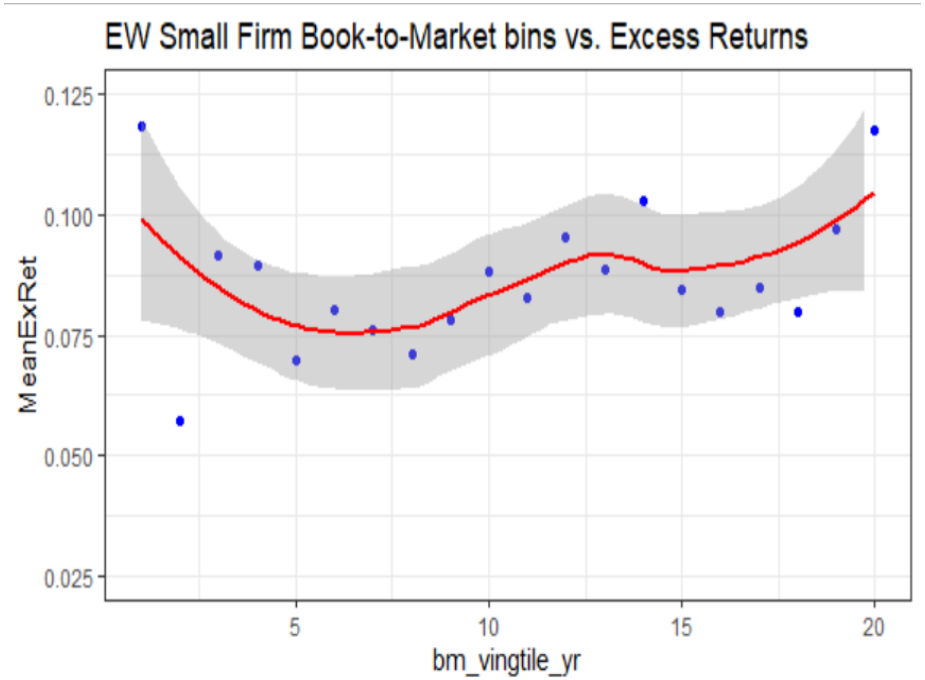
```
> # first, create a rank variable based on market equity of the firms
➢ for (i in 1981:2014)
➢ {
➢ StockRetAcct_DT[year == i,size_rank := rank(-lnME)]}
>
> # generate True/False dummy variable for whether in top 500 largest stocks or not
> StockRetAcct_DT[,LargeStock := (size_rank<501)]
> EW_BM_MutualFunds_yr_Large <- StockRetAcct_DT[LargeStock==TRUE,list(MeanExRetYr = mean(ExRet)),
+ by = list(bm_vingtile_yr, year)]
> EW_BM_MutualFunds_yr_Small <- StockRetAcct_DT[LargeStock==FALSE,list(MeanExRetYr = mean(ExRet)),
+ by = list(bm_vingtile_yr, year)]
>
> # then average across years
> EW_BM_MutualFunds_yr_Large <- EW_BM_MutualFunds_yr_Large[,list(MeanExRet = mean(MeanExRetYr)),
+ by = bm_vingtile_yr]
> EW_BM_MutualFunds_yr_Small <- EW_BM_MutualFunds_yr_Small[,list(MeanExRet = mean(MeanExRetYr)),
+ by = bm_vingtile_yr]
> qplot(bm_vingtile_yr, MeanExRet,data = EW_BM_MutualFunds_yr_Large, col=I("blue"), na.rm = TRUE,
+ main = "EW Large Firm Book-to-Market bins vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
> qplot(bm_vingtile_yr, MeanExRet,data = EW_BM_MutualFunds_yr_Small, col=I("blue"), na.rm = TRUE,
+ main = "EW Small Firm Book-to-Market bins vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
```

UCLAAnderson
SCHOOL of MANAGEMENT

e. Nonlinearity: Small vs. big

- Clear interaction between size (market equity) and value effect
  - Little evidence of value effect in S&P500
  - Note: "EW" in the plot titles stands for Equal-Weighted





- Equal-weighting in the portfolios induces more trading and transitory price moves, that may not be tradeable due to low liquidity, show up as high return.
  - Common practice is to ***value-weight*** stocks within each portfolio

UCLAAnderson
SCHOOL of MANAGEMENT

e. Tradeability v2: Value-weighting

- Note that we value-weight within each portfolio each year, relying on beginning-of-year market values of the stocks
  - We then equal-weight across the years

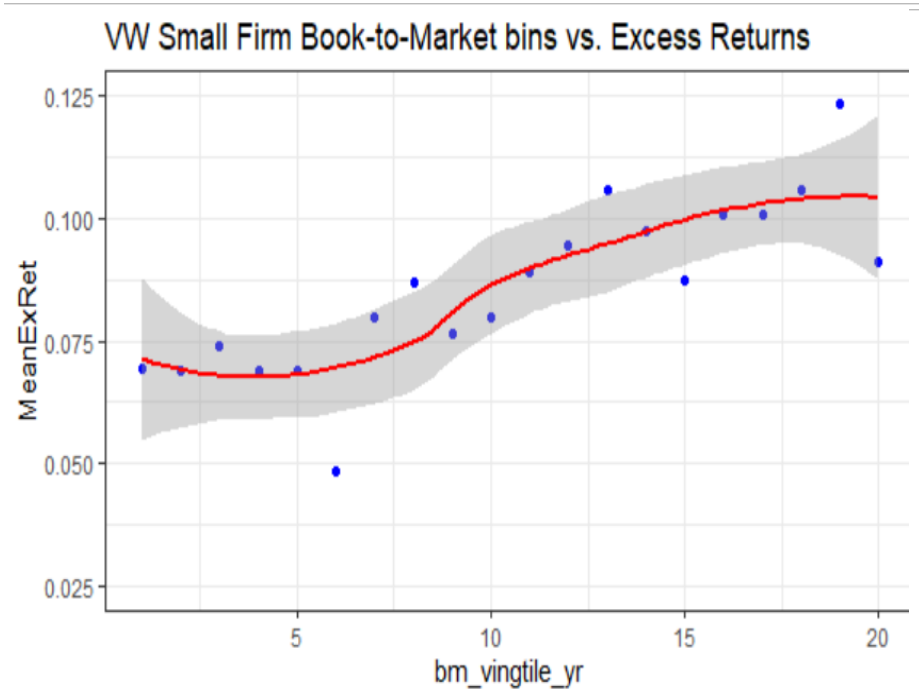
```
> EW_BM_MutualFunds_yr_Large <- StockRetAcct_DT[LargeStock==TRUE,list(MeanExRetYr = weighted.mean(ExRet, MEwt)),
+ by = list(bm_vingtile_yr, year)]
> EW_BM_MutualFunds_yr_Small <- StockRetAcct_DT[LargeStock==FALSE,list(MeanExRetYr = weighted.mean(ExRet, MEwt)),
+ by = list(bm_vingtile_yr, year)]
>
> # then average across years
> EW_BM_MutualFunds_yr_Large <- EW_BM_MutualFunds_yr_Large[,list(MeanExRet = mean(MeanExRetYr)),
+ by = bm_vingtile_yr]
> EW_BM_MutualFunds_yr_Small <- EW_BM_MutualFunds_yr_Small[,list(MeanExRet = mean(MeanExRetYr)),
+ by = bm_vingtile_yr]
>
> qplot(bm_vingtile_yr, MeanExRet,data = EW_BM_MutualFunds_yr_Large, col=I("blue"), na.rm = TRUE,
+ main = "VW Large Firm Book-to-Market bins vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
>
> qplot(bm_vingtile_yr, MeanExRet,data = EW_BM_MutualFunds_yr_Small, col=I("blue"), na.rm = TRUE,
+ main = "VW Small Firm Book-to-Market bins vs. Excess Returns") + geom_smooth(col=I("red")) + theme_bw()
```

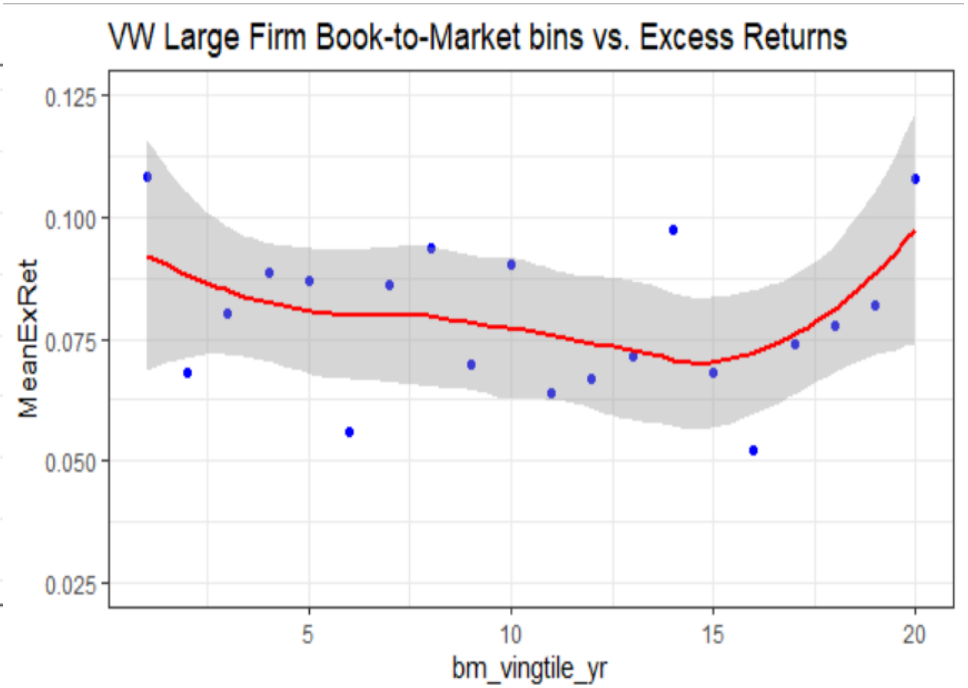
- Technical note: the *weighted.mean* function does not require the input weights to sum to one, it transforms the weights within the function so they do sum to one when taking the mean

UCLAAnderson
SCHOOL of MANAGEMENT

e. Tradeability v2: Value-weighting

- Notice that value spread is further reduced when you value-weight
  - Of course, still nothing in the biggest stocks





- So, is value-investing more or less dead?
  - More on this when we get to omitted variables later in the class

UCLAAnderson
SCHOOL of MANAGEMENT

f. …and the model specification

We now know that:

- The value vs growth spread is there for small firms (smaller than 500 biggest)
  - Less so for the big firms
- It's a cross-sectional phenomenon (potentially also time-series component)
- There are potential outlier issues that means creating 'bins' for b/m values may be more robust than using lnBM directly
  - Winsorizing also gets at this

- A reasonable, but not uniquely so, ***model specification*** is then:

$$E_t\left[R^e_{i,t+1}\right] = \lambda_{0,t} + \left(\lambda_1 + \lambda_2 lnME_{i,t}\right) lnBM_{i,t}$$

- Here we expect that $\lambda_1 > 0$ and $\lambda_2 < 0$.
  - I included a ***t*** on $\lambda_{0,t}$ to allow for the value vs. growth spread being purely cross-sectional.
  - I.e., if you go long and short equal amounts, you don't care about $\lambda_{0,t}$

UCLAAnderson
SCHOOL of MANAGEMENT

f. Fama-MacBeth regressions

A good way to test this model is by running Fama-MacBeth regressions

- Allows for time-varying number of firms
- Allows for time-varying intercept

To keep things simple, let's for now consider the model:

$$E_t\left[R^e_{i,t+1}\right] = \lambda_{0,t} + \lambda_1 lnBM_{i,t}$$

Here, we are interested in λ_1

f. Fama-MacBeth procedure

Recall procedure

- Run *T* cross-sectional regressions

$$R_{i,t}^{e} = \lambda_{0,t} + \lambda_{1,t} lnBM_{i,t-1} + e_{i,t}$$

- Save all the *T* estimated $\lambda_{1,t}$ and obtain

$$\widehat{\lambda_1} = \frac{1}{T}\sum_{t=1}^{T} \lambda_{1,t}$$

And

$$var\left(\widehat{\lambda_1}\right) = var_T\left(\lambda_{1,t}\right)/T$$

UCLAAnderson
SCHOOL of MANAGEMENT

f. Fama-MacBeth produces trading strategies

Note that the Fama-MacBeth coefficients are ***implementable trading strategies*** (ignoring transaction costs)

- In particular:

$$\lambda_{1,t} = \frac{1}{N_t} \frac{lnBM'_{t-1} - E_i\left(lnBM_{i,t-1}\right)}{var_i\left(lnBM_{i,t-1}\right)} R_t^e$$

where $lnBM_{t-1}$ and R_t^e are N_t x 1 vectors corresponding to the number of stocks in the regression at each time. The *i* subscript in E_i and var_i denotes a the expectation and variance taken ***across stocks*** at a point in time.

- Thus, $\boldsymbol{\lambda_{1,t}}$ ***is a long-short portfolio*** (weights sum to zero) based on the book-to-market signal where the weight in stock *i* from time *t-1* to time *t* is:

$$w_{i,t-1} = \frac{1}{N_t} \frac{lnBM_{i,t-1} - E_i\left(lnBM_{i,t-1}\right)}{var_i\left(lnBM_{i,t-1}\right)}$$

UCLAAnderson
SCHOOL of MANAGEMENT

f. Fama-MacBeth t-stats are scaled Sharpe Ratios

Further, the t-statistic for testing whether the characteristic lnBM is priced or not is:

$$t = \frac{\widehat{\lambda_1}}{\sqrt{var_T(\lambda_{1,t})}}\sqrt{T}$$

where $\widehat{\lambda_1}$ is the average excess return to the traded portfolio and $\sqrt{var_T(\lambda_{1,t})}$ is its standard deviation.

- Thus, the Fama-MacBeth regression test is a test of whether a traded long-short portfolio based on the characteristic has a high Sharpe ratio.
- Makes sense!

f. Fama-MacBeth trading strategies

- Absent any other information (other correlated factors, time-varying volatility), the Fama-MacBeth regression is in fact constructing the maximum Sharpe ratio portfolio related to book-to-market ratios given the null hypothesis:

$$E_t\left[R^e_{i,t+1}\right] = \lambda_{0,t} + \lambda_1 lnBM_{i,t}$$

using the implicit OLS assumptions that error terms are homoscedastic and uncorrelated and that the portfolio variance is constant across t

- Is the former assumption reasonable?
- Where was the latter assumption implicitly made?

In sum, given your model of expected returns, the trading strategy is handed to you through the portfolio weights from two slides ago!

- Admittedly, things do get more complicated when there are more factors in the above regression
- We discuss this next

Appendix

Review and background

UCLAAnderson
SCHOOL of MANAGEMENT

Appendix:
Introduction to Data Analytics and R

a. Big Data and Data Analytics
b. R/Rstudio
c. R data types
d. R Objects
e. Sub-setting Variables
f. Sub-setting Observations
g. Reshaping Data
h. Merging/Joins
i. Aggregation
j. R functions

UCLAAnderson
SCHOOL of MANAGEMENT

a. Big Data and Data Analytics

There are two types of Big Data:

Too Large:

Data that is so large that manipulating the data for simple summaries like means or tables is a challenge even in a relatively powerful computing environment

Too Complicated:

Data that is very high dimensional. The limits of visualization and standard statistical summaries is easily reached by dimension 4 or 5!

UCLAAnderson
SCHOOL of MANAGEMENT

a. Big Data and Data Analytics

Too Large:

Google has seen 30 trillion URLs, crawls over 20 billion of those a day, and answers 100 billion search queries a month.

Operations data provided by sensors in a production or operating process (Google cars produce 2GB/sec)

Too Complicated:

Retailer data: 1.5 million UPCs x 2000 stores x 300 weeks x 10 Causal Variables (sales, price, promotion)

Product Recommendation systems: too many consumers and too many products

UCLAAnderson
SCHOOL of MANAGEMENT

a. Big Data and Data Analytics

Too Large:

This is a computer science problem not an analytics problem.

Too Complicated:

This is a problem which requires structure and model building based on that structure.

This will be our focus in this course.

UCLAAnderson
SCHOOL of MANAGEMENT

b. R as the New Standard

R is ***the*** language of data science.

R is a powerful and extensible language. More than 5,000 extension packages have been developed. Runs on all platforms. Freeware is a virtue not a negative!

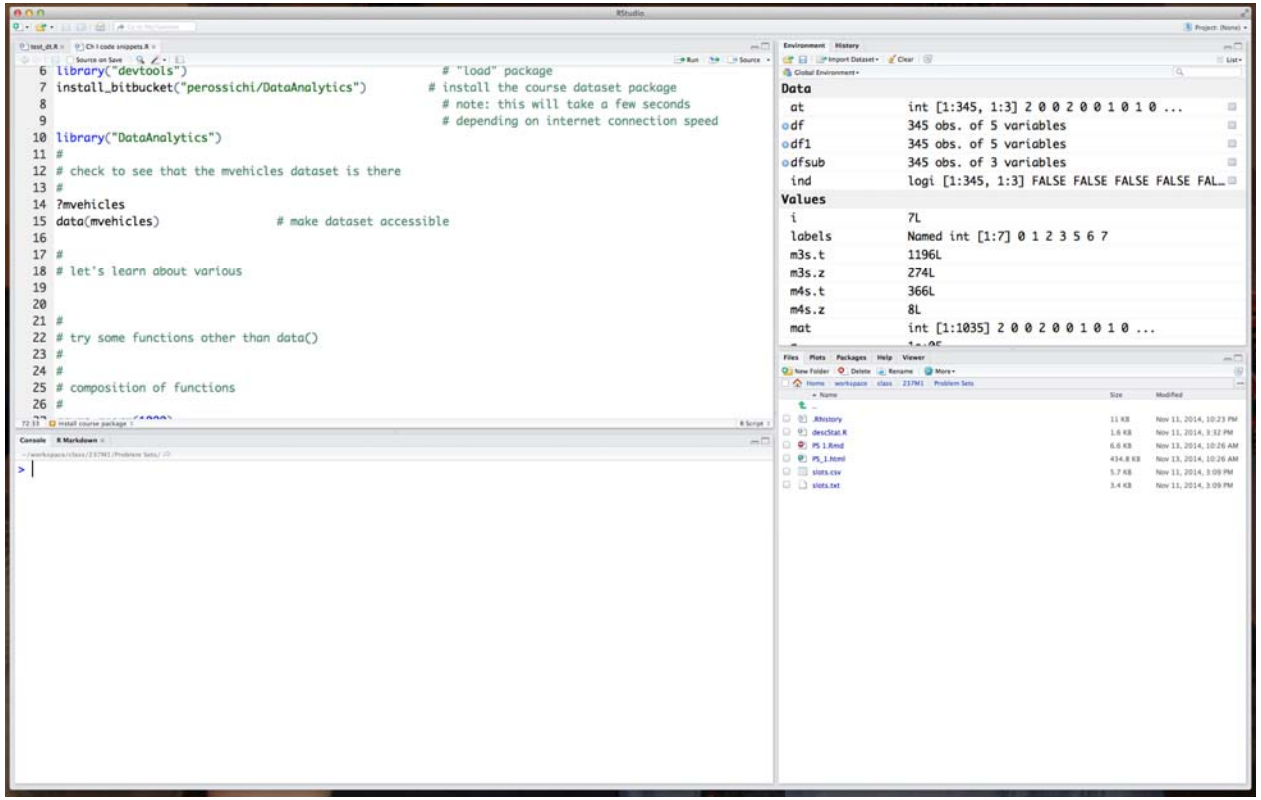
All new developments in statistics or machine learning will *first* be available as R implementations.

All of the leading firms use R – this includes the tiny start-up here in Silicon Beach or the mighty Google, Amazon, Ebays etc.

R is international. If anything, R is more popular in Europe and the Pacific Rim than in North America.

UCLAAnderson
SCHOOL of MANAGEMENT

b. The R Studio Environment



4 panes in Rstudio:

Upper Left: R script

Bottom Left: Console

Upper Right: Workspace,History

Lower Right: Files, Plots, Packages, Help

UCLAAnderson
SCHOOL of MANAGEMENT

b. The R Studio Environment

R commands can be long.
R is picky about syntax.

Therefore, it is best to type commands into the script window and execute so that you can correct and iterate!

Save the script file in some logical place on your computer and you can return to your commands.

Advantages: 1. less frustrating 2. you can replicate your analysis (no billion $ errors like the Hedge fund that used Excel or the economists who made the wrong policy recommendation)

Debt

FAQ: Reinhart, Rogoff, and the Excel Error That Changed History

By Peter Coy | April 18, 2013

Photograph by Gregor Schuster

Harvard University economists Carmen Reinhart and Kenneth Rogoff have acknowledged making a spreadsheet calculation mistake in a 2010 research paper, "Growth in a Time of Debt" (PDF), which has been widely cited to justify budget-cutting. But the authors stand by their conclusion that higher government debt is associated with slower economic growth. Here's what you need to know:

UCLAAnderson
SCHOOL of MANAGEMENT

b. R packages

R is designed to be extendible. When you install R, you are installing a very large set of commands and datasets that are called "base R."

To add capabilities and data, you need to install what are called "packages." Packages are bundles of code (R and other languages), datasets, and documentation.

Most packages can be found on the world-wide network of mirror sites called Comprehensive R Archive Network (CRAN). But others can be found on BitBucket, R-forge, and Github.

We will install several different packages during the course. To install a package, you need to do two things:

1. Install (download and put in place) packages
2. "load" it or make it available in the current R session.

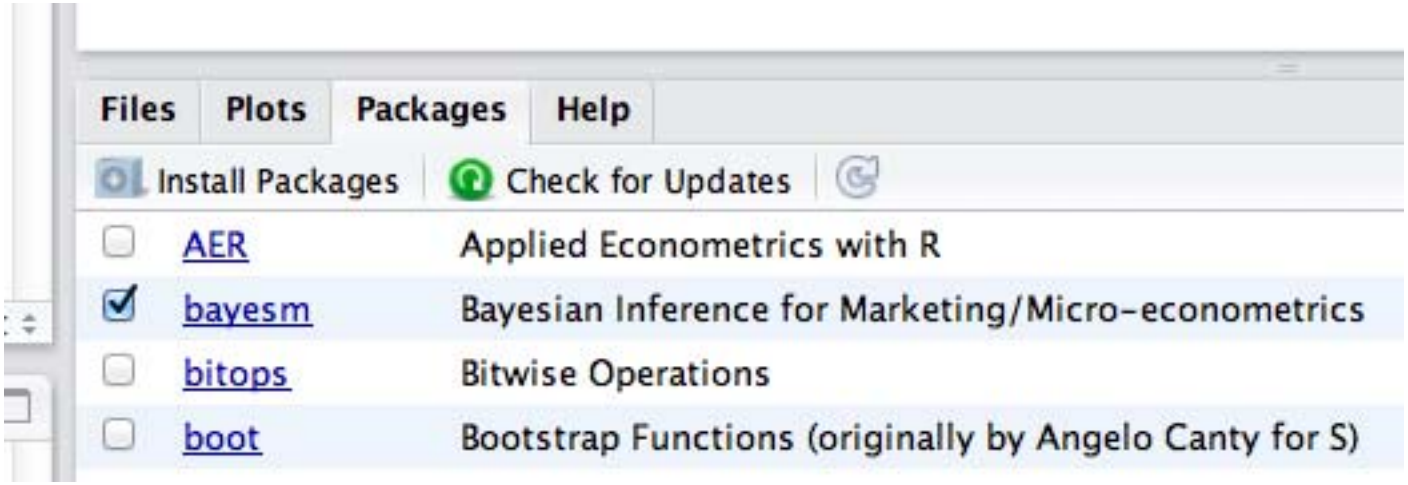
UCLAAnderson
SCHOOL of MANAGEMENT

b. R packages

Rstudio makes this simple. The packages “tab” allows you to install and load packages from CRAN with a few clicks.

Click “Install Packages” to install a new package. Once installed, click on the “checkbox” to “load”.

When you do this, you will see R command generated and executed in the console window. This is “command” vs. GUI way!



UCLAAnderson
SCHOOL of MANAGEMENT

b. DataAnalytics Package

Peter Rossi has created a repository for a DataAnalytics package on the BitBucket repository site.

To install the package, you will first need to install the package, *devtools*, from CRAN and then use a special command to load the package.

The easiest way to do this is to simply highlight and execute the relevant code in `Ch I code snippets.R`. (see class web page)

UCLAAnderson
SCHOOL of MANAGEMENT

b. DataAnalytics Package

```
> library("devtools", lib.loc="/Library/Frameworks/R.framework/Versions/3.0/Resources/library")
> install_bitbucket("perossichi/DataAnalytics")        # install the course dataset package
Downloading bitbucket repo perossichi/DataAnalytics@master
Installing DataAnalytics
```

When you do this, you will see the commands being executed in the console window. You may get a warning but don't worry.

This may take 5-30 seconds depending on your internet connection speed.

To verify that it worked, check that you can access the "`mvehicles`" dataset, by typing `?mvehicles` into the console (or highlight the command in the code snippets file). The help file should appear in the help window. The help file tells you that this is data on about 2000 2011 model cars and trucks.

To "load" the dataset, use `data(mvehicles)`. All datasets must be "loaded" using the `data()` command to access their contents.

b. The R workspace

R has the philosophy that all data and code should be loaded into memory (the "workspace") and then executed. This means that all datasets need to fit into available memory.

Is this a limitation? Not really, with today's cheap memory. With as little as 4-8GB of memory we can analyze virtually any dataset with less than 10,000,000 observations.

On a Linux server, we can go up to 500,000,000 easily. Beyond this point, you need to stop using base R and shift to `Revolution R`.

RStudio displays what is "in" the workspace in the upper right corner in the "environment" tab.

UCLAAnderson
SCHOOL of MANAGEMENT

c. R data types

R uses five basic types of data:

1. numeric
2. character
3. logical or Boolean
4. factor << avoid using them consider using
   `options(stringsAsFactors = F)`
5. date

You can distinguish between a double precision floating point number and an integer but it's not worth the effort.

The `str( )` function is very useful to determine what type of data is in an object. Other useful functions are `is.na()`, `is.numeric()`, `is.character()`, `is.nan()`.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Objects and functions

You may have noticed that all of the R commands use functional notation, e.g `data(mvehicles)` like `f(x)`. "data" is the function and "mvehicles" is the input.

In general a function takes some inputs and makes one output from each set of inputs.

Executing a function is termed "calling the function." In programming, some functions are called for their effects and not their output . `data()` is one such function. All `data(mvehicles)` does is load the dataset `mvehicles` into the workspace, making its contents available.

Let's try two more standard functions in R.

UCLAAnderson
SCHOOL of MANAGEMENT

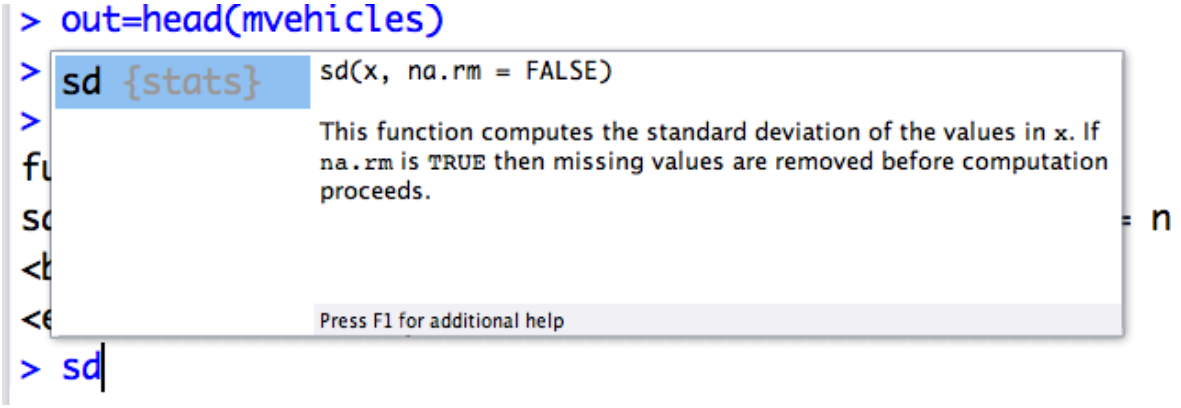
d. Composition of functions

R functions can be composed. That is, you can take the output from one function and call another function with this output.

Let's do this to calculate the standard dev of 1000 Normal R.Vs.

```
> rnums=rnorm(1000)
> sd(rnums)
[1] 0.9698541
> sd(rnorm(1000))                    # the fast way
[1] 1.034233
```

Rstudio tip:
If you type the first few letters of a command or dataset and hit <tab> then you will see what this is!



UCLAAnderson
SCHOOL of MANAGEMENT

d. R objects

R is what is called an object-oriented programming language.

What does this mean?

Everything is an object (data, code …)

All “commands” are functions which take objects as input and create objects as output.

At first, this can be confusing but then it is liberating and simplifying as everything is always in the same form!

object = function(object)

Basically, learning R is learning about what functions there are and what types of objects they can manipulate.

UCLAAnderson
SCHOOL of MANAGEMENT

d. Useful Types of R objects

Let's start simple:

There are four types of objects we need to know about:

1. vectors – an ordered list of same type of object

    vectors can be made of numeric objects (numbers) or character strings or so-called logical (Boolean) elements (i.e. composed of only TRUE or FALSE).

2. matrices/arrays – to hold multi-dimensional data

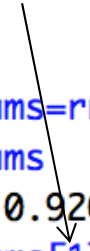
3. lists – very general "containers" which can be indexed and comprised of any sort of R objects.

4. data frames – used to hold datasets, has properties of a list and a matrix or array

UCLAAnderson
SCHOOL of MANAGEMENT

d. vectors

Let's create a vector, display it, and access various parts of it. Note that all elements of a vector must be of the same data type! The "[ ]" operator is used to access parts of a vector.



```
> rnums=rnorm(5)
> rnums
[1]  0.9200508  0.7164868 -0.8120425  1.0415078  0.8396766
> rnums[1]
[1] 0.9200508
> # let's make a vector by "concantenating" a set of integer index numbers
> ind.vec=c(2,1,4)
> rnums[ind.vec]
[1] 0.7164868 0.9200508 1.0415078
> rnums[3:5]
[1] -0.8120425  1.0415078  0.8396766
> rnums[c(TRUE,FALSE,FALSE,FALSE,TRUE)]
[1] 0.9200508 0.8396766
> sort(rnums)
[1] -0.8120425  0.7164868  0.8396766  0.9200508  1.0415078
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. matrices

A matrix is really a vector but with a "dim" attribute set. You can create a matrix from a vector in a column by column fashion. Again, this means that matrices must be of one data type.

```
> mat=matrix(c(1:9),nrow=3)
> dim(mat)
[1] 3 3
> nrow(mat)
[1] 3
> ncol(mat)
[1] 3
> colnames(mat)=c("A","B","C")
> mat
     A B C
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
```

```
> mat[1,1]
A
1
> mat[2:3,]
     A B C
[1,] 2 5 8
[2,] 3 6 9
> mat[,1]
[1] 1 2 3
> mat[,-1]          # everything but the first col
     B C
[1,] 4 7
[2,] 5 8
[3,] 6 9
> mat[,c("A","C")]
     A C
[1,] 1 7
[2,] 2 8
[3,] 3 9
```

d. lists

A major limitation of vectors/matrices and arrays is that they can only contain one data type. Data sets are often a combination of date stamps, numeric data, character data such a description texts or addresses, and qualitative variables such as ethnicity which are stored as “factors.”

In addition, output from statistical models is of a variety of types including lists of coefficients and variance-covariance arrays.

To store these kinds of objects, R has the object type “list.” lists can be comprised of any combination of data types but can be accessed sequentially as a vector. In addition, elements of a list can be accessed by names given to each element.

UCLAAnderson
SCHOOL of MANAGEMENT

d. lists

```
> my_family = list(pop=list(name="peter",dob=as.Date("1955/11/25")),
+                  son=list(name="ben",dob=as.Date("1988/08/08")),
+                  daughter=list(name="emily",dob=as.Date("1984/08/29")))
> my_family$pop
$name
[1] "peter"

$dob
[1] "1955-11-25"

> my_family[[1]]        # note the use of the double "[]"
$name
[1] "peter"

$dob
[1] "1955-11-25"

> my_family$son$dob
[1] "1988-08-08"
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. data frames

It seems, then, the data sets should be stored in an object which as both the properties of a list and a matrix.

List: access variables by name

Matrix: maintain the idea of “rows” as observations and “cols” as variables and use matrix access

A data frame is basically a list which has an access operator like “[ ]” added to it.

It is very convenient for storing datasets but access to elements can be slow for VERY large datasets (> 10,000,000 rows). Later, we will introduce the “data.table” extension which overcomes speed of access problems.

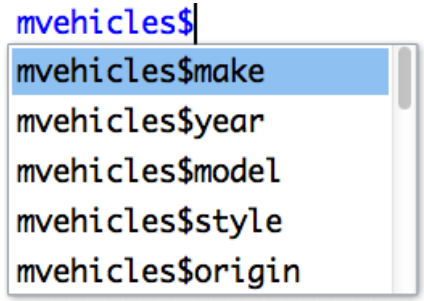
UCLAAnderson
SCHOOL of MANAGEMENT

d. data frames

Let's snoop around the `mvehicles` data frame.

```
> head(mvehicles,n=3)              # display the first few observations
  make year    model                                        style origin bodytype      emv seats
1  BMW 2011 5 Series                    528i 4dr Sedan (3.0L 6cyl 8A) Europe    Sedan 45362.90     5
2  BMW 2011 5 Series 535i xDrive 4dr Sedan AWD (3.0L 6cyl Turbo 8A) Europe    Sedan 54518.63     5
3  BMW 2011 5 Series               535i 4dr Sedan (3.0L 6cyl Turbo 6M) Europe    Sedan 50066.94     5
  roominess mpg  warranty appearance    luxury    sporty     speed technology sales
1 0.3452545  25 0.6833333  0.7922577 0.8603964 0.6717107 0.6391753    0.71875 19076
2 0.3452545  23 0.6833333  0.7922577 0.8603964 0.6717107 0.6626617    0.71875   813
3 0.3452545  22 0.6833333  0.7922577 0.8603964 0.6717107 0.7319588    0.71875 14990
> str(mvehicles)                  # "structure" command which tells you about the object
'data.frame':   1999 obs. of  17 variables:
 $ make      : chr  "BMW" "BMW" "BMW" "BMW" ...
 $ year      : int  2011 2011 2011 2011 2011 2011 2011 2011 2011 2011 ...
 $ model     : chr  "5 Series" "5 Series" "5 Series" "5 Series" ...
```

Forget the names of the variables? Just use <tab> to complete.



UCLAAnderson
SCHOOL of MANAGEMENT

d. data frames

Data frames have both the capabilities of a list to access variables by names as well as a matrix to access either rows or columns by their corresponding numbers.

```
> mvehicles[1,]                    # first row of mvehicles (first obs) note the ","
  make year     model                                style origin bodytype     emv seats roomi
1  BMW 2011 5 Series 528i 4dr Sedan (3.0L 6cyl 8A) Europe    Sedan 45362.9     5 0.345
     luxury    sporty     speed technology sales
1 0.8603964 0.6717107 0.6391753    0.71875 19076
> head(mvehicles$mpg)              # show the first 6 elements of the mpg variable
[1] 25 23 22 18 17 22
> head(mvehicles[mvehicles$make=="Lexus",])  # show only those observations for Lexus
      make year     model                                                        style or
234   Lexus 2011    IS 350                                        4dr Sedan (3.5L 6cyl 6A)
249   Lexus 2010 LS 600h L        4dr Sedan AWD (5.0L 8cyl gas/electric hybrid CVT)
292   Lexus 2012       LFA                                      2dr Coupe (4.8L 10cyl 6AM)
513   Lexus 2011   CT 200h        4dr Hatchback (1.8L 4cyl gas/electric hybrid CVT)
514   Lexus 2011   CT 200h Premium 4dr Hatchback (1.8L 4cyl gas/electric hybrid CVT)
1241 Lexus 2011    GS 350                                        4dr Sedan (3.5L 6cyl 6A)
```

UCLAAnderson
SCHOOL of MANAGEMENT

e. Sub-setting Variables

Sub-setting variables is very easy when using a data frame. You simply use the matrix ways of referring to columns. It is preferred that you use the actual variables names rather than column numbers so that your code is robust to ordering changes.

```
> subset.mvehicles=mvehicles[,c("emv","make","year","mpg")]
> head(subset.mvehicles)
       emv make year mpg
1 45362.90  BMW 2011  25
2 54518.63  BMW 2011  23
3 50066.94  BMW 2011  22
4 61599.65  BMW 2011  18
5 60560.78  BMW 2011  17
6 35441.33 Ford 2011  22
```

UCLAAnderson
SCHOOL of MANAGEMENT

f. Logical Expressions and Sub-setting Observations

We have seen that a logical vector can be used to subset observations. We can use logical expressions to create such a logical vector. We have already seen a very simple case of this where we sub-set to only the "Lexus" models.

Logical expressions are made from the following logical operators:

"=="	equal to
"!="	not equal to
">", ">="	greater than or greater than or equal to
"<", "<="	
"&"	and
"|"	or
"y %in% x"	check each element of y to see if it is in x

UCLAAnderson
SCHOOL of MANAGEMENT

f. Logical Expressions and Sub-setting Observations

Let's try an example.

```
> german.highmpg = mvehicles[mvehicles$make %in% c("BMW","Audi","Mercedes-Benz","Volkswagen") &
+                            mvehicles$mpg > 20,]
> table(german.highmpg$make)

        Audi          BMW Mercedes-Benz    Volkswagen
          22           18             7           124
> summary(german.highmpg$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  21.00   22.00   25.00   25.04   27.00   34.00
```

Now suppose we wanted all the four door sedans. We now have to search thru the string "style" for the occurrence of "4 dr." We can use "`grepl`" for this.

```
> four.door = mvehicles[grepl("4dr",mvehicles$style,ignore.case=TRUE),]
> dim(four.door)
[1] 1598   17
> dim(mvehicles)
[1] 1999   17
```

UCLAAnderson
SCHOOL of MANAGEMENT

f. Creating New Variables and Recoding

We will want to create new variables in a data.frame as well as remove unused variables.

```
> mvehicles$lnemv=log(mvehicles$emv)
> mvehicles$emv = NULL                    # "NULL" is a special reserved word
```

Is this dangerous? Not really. Why?

Change a continuous variable to a qualitative or categorical var.

```
> pricecat=cut(mvehicles$emv,breaks=quantile(mvehicles$emv,probs=c(0,.25,.75,1.0)))
> # create categorical variable (R calls this a factor)
> str(pricecat)
 Factor w/ 3 levels "(9.93e+03,2.42e+04]",..: 3 3 3 3 3 2 2 2 2 2 ...
> levels(pricecat)=c("low","med","high")                    # change labels
> table(pricecat)
pricecat
 low  med high
 499  999  500
```

UCLAAnderson
SCHOOL of MANAGEMENT

g. Reshaping Data

Vanguard

In 237Q, you developed regression methods to evaluate the performance of various Vanguard mutual funds.

Your first task is to download data on monthly returns (percent change in the fund's trading price). The web interface asks you to supply the ticker symbols of the funds you want to download and you get back one file.

This file is apt not to be in a "shape" for analysis.

UCLAAnderson
SCHOOL of MANAGEMENT

g. Vanguard Data Frame

What does `Vanguard` look like?

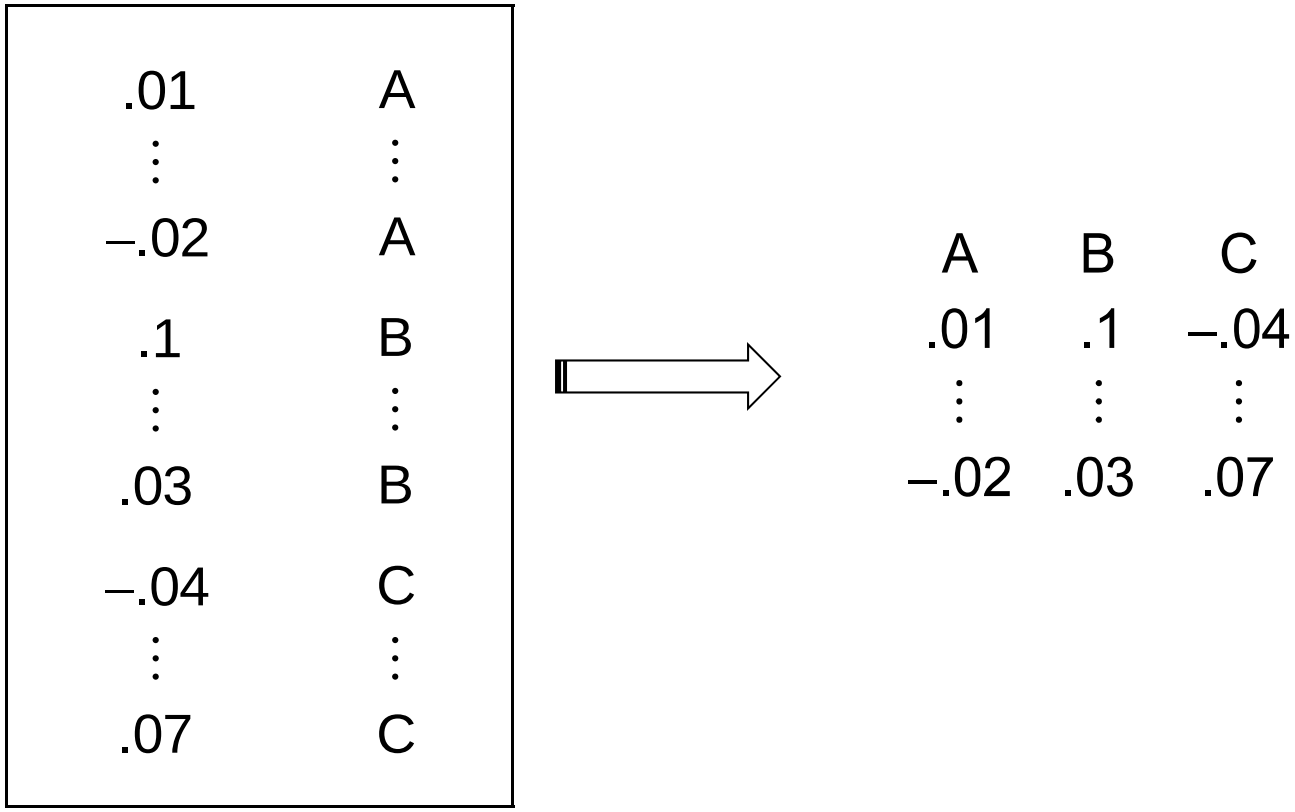
It has the returns series for each fund stacked up on top of each other! The number of observations varies because funds have been started at different points in time!

```
> data(Vanguard)
> head(Vanguard)
        date ticker crsp_fundno   mtna      mret
1 1988-04-29  VEIPX       31217     NA  0.010215
2 1988-05-31  VEIPX       31217     NA  0.027300
3 1988-06-30  VEIPX       31217 16.763  0.030535
4 1988-07-29  VEIPX       31217     NA  0.005797
5 1988-08-31  VEIPX       31217     NA -0.013449
6 1988-09-30  VEIPX       31217 28.259  0.041992
> unique(Vanguard$ticker)
[1] "VEIPX" "VFIAX" "VGENX" "VGHCX" "VMGRX" "VQNPX" "VSMAX" "VTSAX" "VWNFX"
> head(which(Vanguard$ticker=="VGHCX"))   #
[1] 804 805 806 807 808 809
```

UCLAAnderson
SCHOOL of MANAGEMENT

g. Vanguard Data Frame

It would be convenient to "unstack" the long skinny vector of fund returns into an array where each column corresponds to a different fund.



UCLAAnderson
SCHOOL of MANAGEMENT

g. Vanguard Data Frame

There are 9 mutual funds in the dataset. The variable **`ticker`** tells us which fund the data is on. The variable **`mret`** contains the monthly returns for each fund stacked up on top of each other in one column.

Let's "unstack" or reshape our data so that there is an array or spreadsheet with one column for each fund. That is, let's create a new dataset with 10 columns (date and monthly returns for each of the funds.

To do this, we will use the R contributed package, `reshape2`, available from CRAN.

UCLAAnderson
SCHOOL of MANAGEMENT

g. Using reshape2

Here is the R-code to do this:

Variable defining rows

Variable defining columns

```
library(reshape2)
data(Vanguard)
Van=Vanguard[,c(1,2,5)]  # grab relevant cols
V_reshaped=dcast(Van,date~ticker,value.var="mret")
```

```
> dim(Vanguard)
[1] 2294    5
> dim(V_reshaped)
[1] 349  10
> head(V_reshaped)
        date VEIPX VFIAX     VGENX     VGHCX VMGRX VQNPX VSMAX VTSAX VWNFX
1 1984-06-29    NA    NA -0.067921  0.002070    NA    NA    NA    NA    NA
2 1984-07-31    NA    NA -0.105381 -0.021694    NA    NA    NA    NA    NA
3 1984-08-31    NA    NA  0.186717  0.090813    NA    NA    NA    NA    NA
4 1984-09-28    NA    NA  0.016895 -0.023233    NA    NA    NA    NA    NA
5 1984-10-31    NA    NA -0.041537  0.027750    NA    NA    NA    NA    NA
6 1984-11-30    NA    NA  0.010834  0.002893    NA    NA    NA    NA    NA
```

UCLAAnderson
SCHOOL of MANAGEMENT

h. Combining Datasets

There are three types of combination situations:

1. Concatenate two data frames. Both data frames have the same number of variables but represent different observations. For example, updating information with new time periods or combining information across different markets. Use `rbind()`

2. Add variables for the same set of observations. This is a "column" type of concatentation. Use `cbind()`

3. Combine two data sets with a common set of observations. Typically to add new variables or to add information from a look-up table. This is called a merge or a "join". Use `merge()`

UCLAAnderson
SCHOOL of MANAGEMENT

h. Merging or "Joining" Datasets

To examine the relationship between returns on individual Vanguard funds and the market, we need to obtain market returns for the same period of time we observe the Vanguard funds.

A merge is also called a "join" in the database literature.

For each value of the `date` variable in the Vanguard data, we want to pull the associated values from the `marketRf` data file for the various market indices and risk-free rates.

What happens if there is a date for which we have Vanguard return data but we do not happen to have a market return?

"Inner" join: output nothing into the merged dataset

"Outer" join: output a record with a missing value for the market return.

UCLAAnderson
SCHOOL of MANAGEMENT

h. Merging or "Joining" Datasets

As long as whatever graphical or modeling function can handle missing data, we don't care. By default, the "merge" function in R does an "inner" join.

We will output a file with only one column for the common variable (the "by" variable). This is called a "natural" join.

```
merge(datasetA, datasetB,by="<name of common
     var>")
```

`datasetA` is called the "left" and `datasetB` the "right" in the DB literature. R calls them "x" and "y".

UCLAAnderson
SCHOOL of MANAGEMENT

h. Merging or "Joining" Datasets

Merge in progress, merging on date = "2001-01-31"

date	VGHCX
1984 – 06 – 29	.00207
⋮	⋮
2001 – 01 – 31	–.073
⋮	⋮
2013 – 06 – 28	–.004366

date	vwretd
1927 – 07 – 31	.022937
⋮	⋮
2001 – 01 – 31	.0323
⋮	⋮
2013 – 06 – 28	–.013665

date	VGHCX	vwretd
1984 – 06 – 29	.00207	.022937
⋮	⋮	⋮
2001 – 01 – 31	–.073	.0323
⋮	⋮	⋮
2013 – 12 – 31	empty	empty

Output of merge

UCLAAnderson
SCHOOL of MANAGEMENT

i. Aggregation or computing on subsets of data

Another way of computing simple statistics (or any function for that matter) on subsets of the data.

This can be done in a loop or by using the aggregate function or the apply function.

First, via the `aggregate()` function:

```
> means=aggregate(mret~ticker,Vanguard,mean,na.rm=TRUE)
> means
  ticker        mret
1  VEIPX 0.008746119
2  VFIAX 0.003959993
3  VGENX 0.011634089
4  VGHCX 0.013929255
5  VMGRX 0.010418323
6  VQNPX 0.008842613
7  VSMAX 0.008915656
8  VTSAX 0.004827424
9  VWNFX 0.009463979
```

Compute function of `mret` for all subsets defined by the variable `ticker`. The function is `mean()`

UCLAAnderson
SCHOOL of MANAGEMENT

i. Aggregation or computing on subsets of data

Via a loop.

```
> tickers=unique(Vanguard$ticker)
> means=double(length(tickers))
> i = 1
> for(ticker in tickers){
+   means[i] = mean(Vanguard$mret[Vanguard$ticker == ticker],na.rm=TRUE)
+   i=i+1
+ }
> names(means)=tickers
> means
      VEIPX       VFIAX       VGENX       VGHCX       VMGRX       VQNPX       VSMAX       VTSAX
0.008746119 0.003959993 0.011634089 0.013929255 0.010418323 0.008842613 0.008915656 0.004827424
      VWNFX
0.009463979
```

Loop: do everything enclosed in { } for each value in `for()`.

UCLAAnderson
SCHOOL of MANAGEMENT

i. Aggregation or computing on subsets of data

Via `apply()`

```
> means=apply(Van_mkt[,2:10],2,mean,na.rm=TRUE)
> means
      VEIPX       VFIAX       VGENX       VGHCX       VMGRX       VQNPX       VSMAX       VTSAX
0.008746119 0.003959993 0.011634089 0.013929255 0.010418323 0.008842613 0.008915656 0.004827424
      VWNFX
0.009463979
```

"`apply`" the function "`mean`" to selected cols(2) [rows(1)] of the data.frame `Van_mkt`

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

R programming consists of designing and running functions. This is the reason why R is so extendible.

R functions are designed to take the repetitive tasks and bundle them into tools. Essentially, you assemble a useful collection of tools and then put them to work in various combinations.

An R function is an object created by the function command which has the structure (note: arguments can be any valid R object)

```
myFunction = function(arg1,arg2,arg3=default){
        << R statements which manipulate
        the arguments >>

    return(value)
}
```

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

A function can be invoked or "called" simply by using the name of the function and supplying arguments. The order of arguments can be over-ridden by supplying named arguments. Named arguments are to be preferred as in.

You only have to specify named arguments by enough of the first portion of the argument name for the argument to be uniquely identified.

```
out = myFunction(arg2=object2, arg1=object1)
```

NOT

```
out= myFunction(object1,object2)
```

I prefer to name my functions using the "Java" style sequence of words with first letter capitalized for readability.

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

What happens when a function is called?

R creates a new "environment" which is a area where R can look for objects created during the execution of the function. If an object is referenced in the function which cannot be found in the environment created by the function, then R looks for the object in the calling environment.

This can be dangerous!

Avoid depending on it.

Send everything down!

```
> y=1
> foo = function(x) {return(x+y)}
> foo(3)
[1] 4
> y=2
> foo(3)
[1] 5
```

j. R functions

In many languages, there is a distinction between calling by reference or by copy. That is, if you call by reference then only the address of the object is sent to the function. This allows objects to be modified in place as well as avoids unnecessary copying.

In calling by copy, the arguments are copied.

R is a hybrid model. If the arguments are not modified by the functions, then no copying occurs. But if the argument is modified by the function then a copy is made.

This means that R is pretty efficient in function calls.

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

Let's write a non-trivial function which manipulates dates. We will implement error checking to ensure that the arguments are correct.

This function is designed to take a vector of R internal format date types and convert them into the serial month since 01/1960.

The function includes a function "serialmonth" that is only defined inside of the function and, therefore, is never available to be called by the calling environment. This is often used for utility functions that don't include error checking.

Note that this function is fully vectorized in the sense that it will work equally well whether or not you supply a vector or one value.

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

```
#
# example of a function to manipulate dates
#
convertSerialmo = function(date){
  # find serial month corresponding to a date
  # returns serial month where 01/1960 is 0
  serialmonth=function(year,month){
    serialmonth=(year-1960)*12 + month-1
    return(serialmonth)
  }
  adate=attributes(date)$class
  if(is.null(adate)) {
    stop("input is not in date format, try as.Date()")
  }
  else {
    if(adate != "Date") stop("input is not in date format, try as.Date()")
  }
  mo=as.numeric(format(date,"%m"))
  year=as.numeric(format(date,"%Y"))   # %Y is four digit year, %y is two digit
  return(serialmonth(year,mo))
}
```

UCLAAnderson
SCHOOL of MANAGEMENT

j. R functions

```
> dates.chr <- c("02/27/92", "02/27/92", "01/14/92", "02/28/92", "02/01/02")
>
> dates = as.Date(dates.chr, "%m/%d/%y")
>
> convertSerialmo(dates.chr)
Error in convertSerialmo(dates.chr) :
  input is not in date format, try as.Date()
> convertSerialmo(dates)
[1] 385 385 384 385 505
```