UCLAAnderson
SCHOOL of MANAGEMENT

MGMTMFE 431:

Data Analytics and Machine Learning

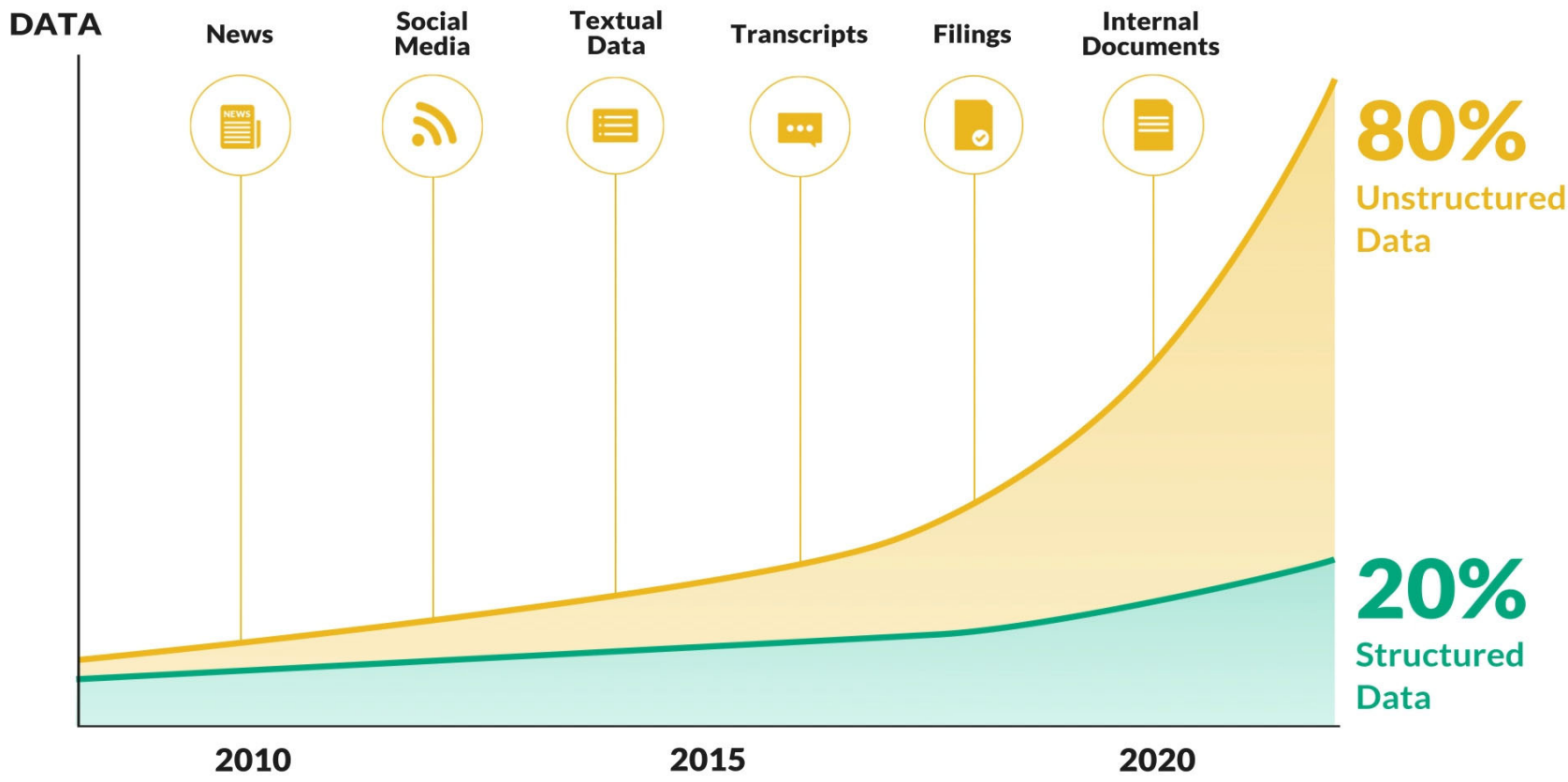
Course Introduction and Topic 1

Spring 2025

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


DATA
News
Social Media
Textual Data
Transcripts
Filings
Internal Documents
80%
Unstructured Data
20%
Structured Data
2010
2015
2020

UCLAAnderson
SCHOOL of MANAGEMENT

Big data in finance

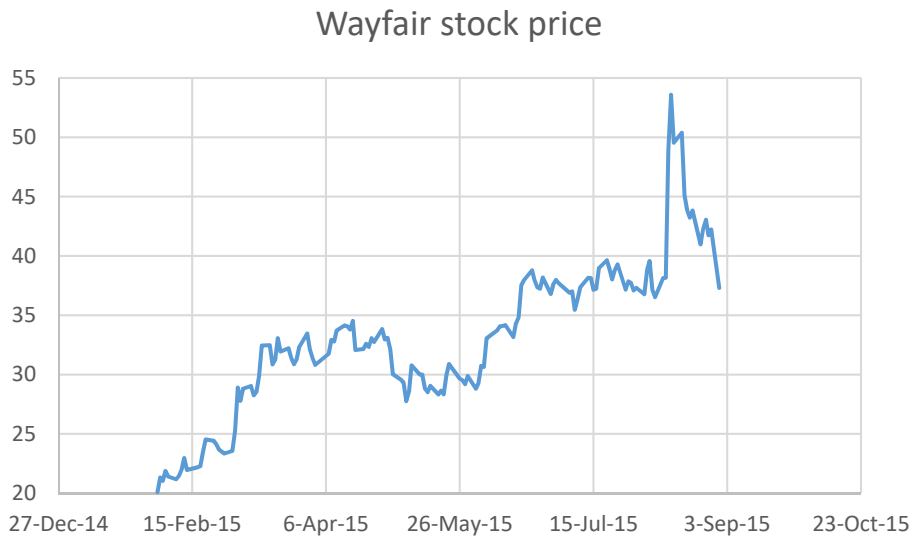
Actors:

- 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

Privateer

- Uses satellite imagery to analyze economic (and other) activity
- https://www.privateer.com
- E.g., retail demand by tracking retail parking lots:

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

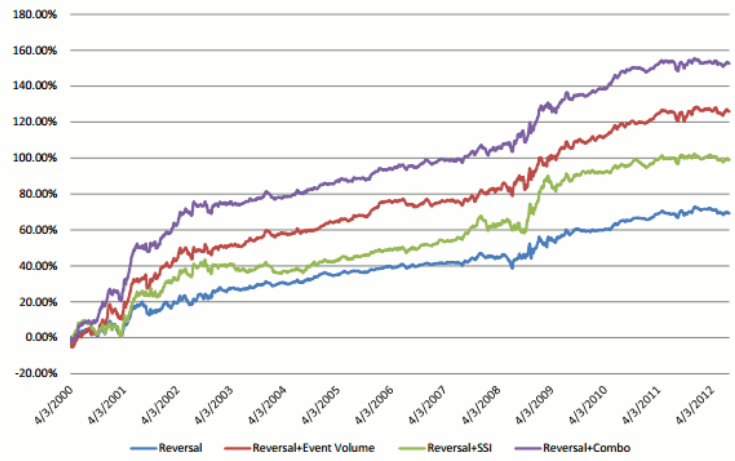
RavenPack

RavenPack

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

- https://www.ravenpack.com/insights

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?

- Real-world 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, features, 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

The Course Outline

1) Introductory Concepts

- Examples of data analytics in finance
- Data analytics and Python, useful tools for data management
- Visualization as a model specification and communication tool
- useful graphics packages in Python
- 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 card databases
- Sample selection

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) Machine Learning II: Nonlinear methods Part 1

- High-dimensional data; model selection and regularization revisited
- Flexible prediction models: Decision trees, bagging (Random Forests), boosting (XGBoost)

6) Unstructured data and textual analysis

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

7) Machine Learning III: Nonlinear methods Part 2

- Classification: k-means clustering, support vector machines
- Deep learning and neural networks

Student presentations

UCLAAnderson
SCHOOL of MANAGEMENT

Some Class Admin

UCLAAnderson
SCHOOL of MANAGEMENT

Classroom Policies

- Please arrive before class begins
- Please don't answers emails, be on chat, etc. during class. Everyone notices: me, your fellow students
  - Disruptive for the students around you
- Please participate, ask questions, more fun!

UCLAAnderson
SCHOOL of MANAGEMENT

Readings

- Lecture Notes

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

- Recommended textbook for statistical learning resource:
  - "Introduction to Statistical Learning"
  - https://www.statlearning.com/

- Advanced statistical learning resource (not directly relevant for class)
  - "Elements of Statistical Learning"
  - https://hastie.su.domains/ElemStatLearn/

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

UCLAAnderson
SCHOOL of MANAGEMENT

Class material

- All class material will be posted on BruinLearn class website
  - I will post lecture notes on the day before class and homework about a week before they are due

UCLAAnderson
SCHOOL of MANAGEMENT

Course Grading

- Problem Sets (6 of 6 graded) 24%
  - Use pre-assigned groups.
- Group project and presentation 24%
  - Presentations will take place in the last classes.
  - Form froups freely of up to 6, minimum 3, students.
- Class participation 6%
- Final exam 46%
  - In-person, cheat-sheet and calculator allowed

UCLAAnderson
SCHOOL of MANAGEMENT

Teaching Assistant

- Kibeom Lee (kibeom.lee.phd@anderson.ucla.edu)

- Office hours/TA session:
  - Fridays 1-2pm

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
- Presentations about 20 mins per group.

UCLAAnderson
SCHOOL of MANAGEMENT

The group projects: example from previous 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

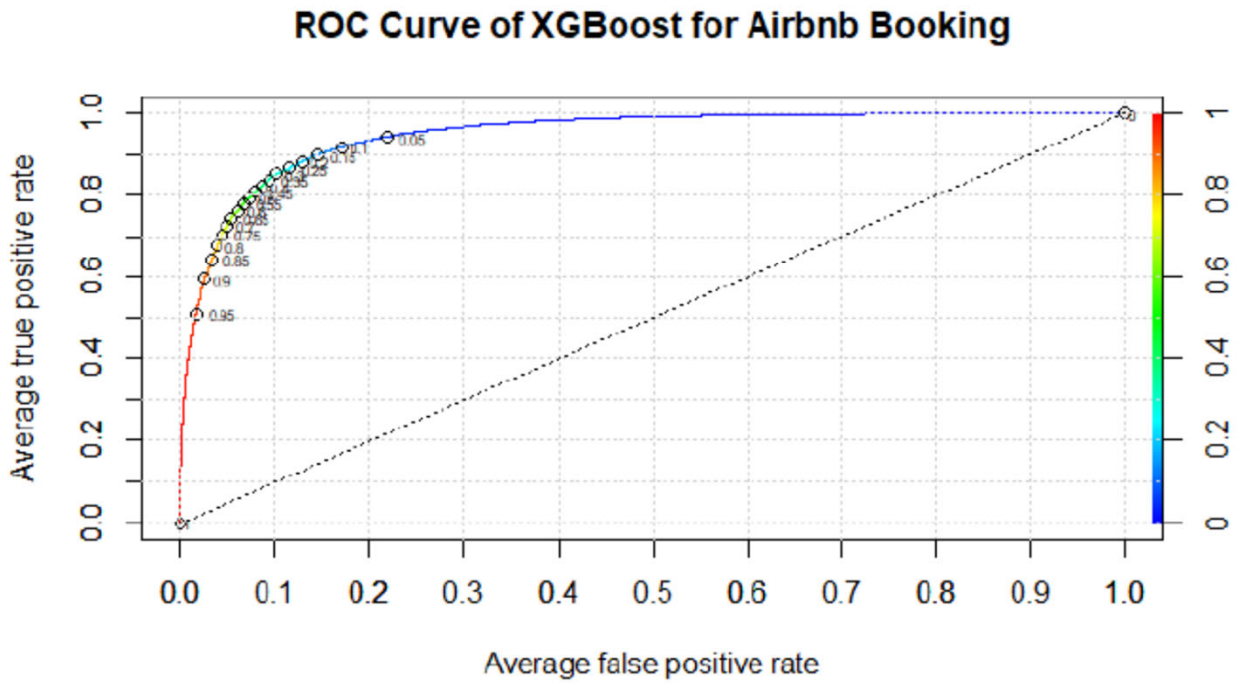
Feature Name	Description
dim_is_requested	True if ds_night receives a booking request eventually; false otherwise.
ds_night	The night on the calendar
ds	The date stamp on which data is collected
id_listing_anon	Anonymized listing ID
id_user_anon	Anonymized user ID of the host of the listing
m_effective_daily_price	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

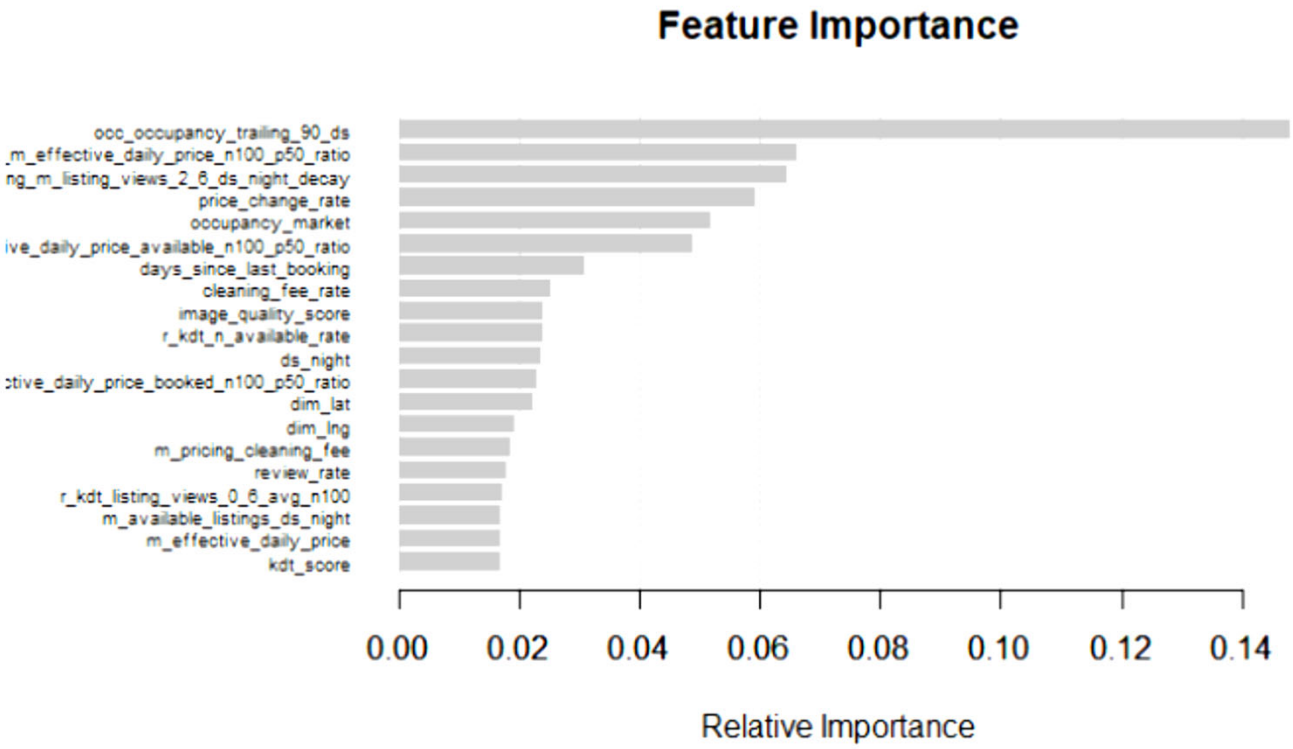
Figure 1. ROC Curve of Airbnb Booking Model



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 Python
c. *matplotlib* and *seaborn* introduction to graphics
d. Intro to *DataFrame* and data management
e. Model specification and visualization

Basic Python reference: www.learnpython.org

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. Two suggested graphics packages

`matplotlib` is a popular graphics package. Elegant plots can be done quickly. Complicated plots can be done efficiently.

- https://matplotlib.org/stable/index.html

`seaborn` builds on matplotlib and offers higher level interface, shorter syntax, and more templates

- https://seaborn.pydata.org/

Baseline packages:

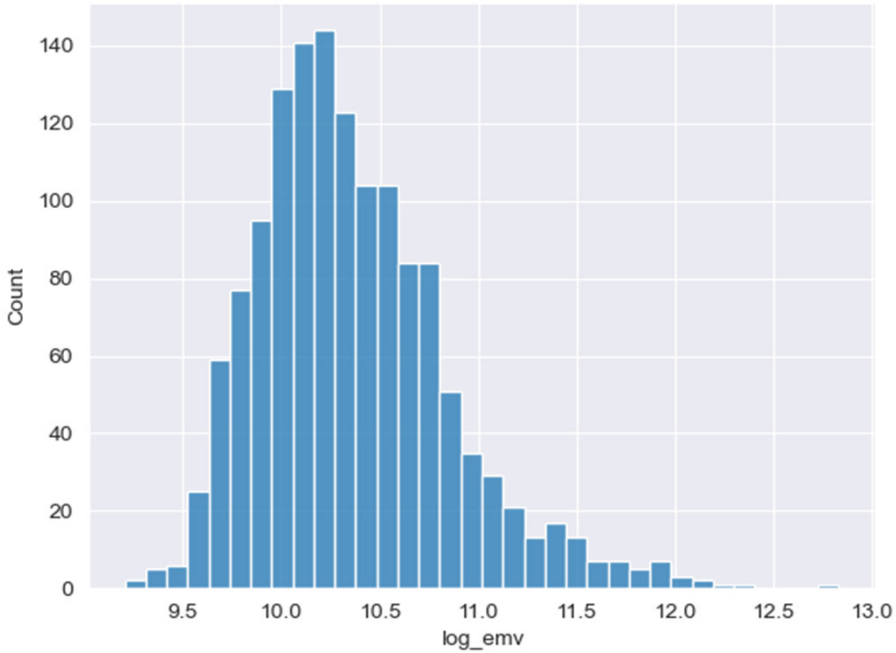
```
# standard set of imports at beginning of code
import os
import matplotlib.pyplot  as plt
import pandas             as pd
import numpy              as np
import seaborn            as sns
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to matplotlib and seaborn

Let's load some data and do some simple standard plots:

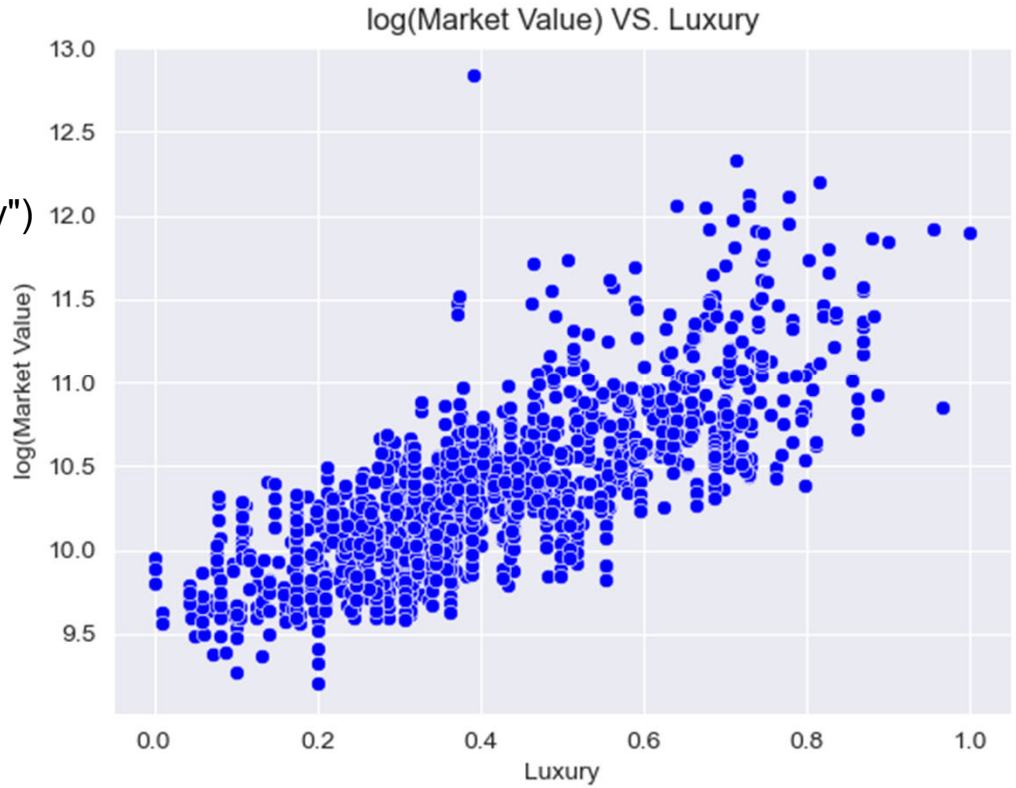
```
# Change working directory to data location
os.chdir("/Users/llochsto/Dropbox/Data Analytics/DAML_2025/Data")

# Load data, extract only cars
mvehicles          = pd.read_csv("mvehicles.csv")
cars               = mvehicles[mvehicles.bodytype != "Truck"]
cars["log_emv"]    = np.log(cars["emv"])

# Plot histogram
plt.figure()
ax = sns.histplot(cars.log_emv,kde=False)
# If you want to save the figure
ax.get_figure().savefig('plot.png')
```



UCLAAnderson
SCHOOL of MANAGEMENT

c. Introduction to matplotlib and seaborn

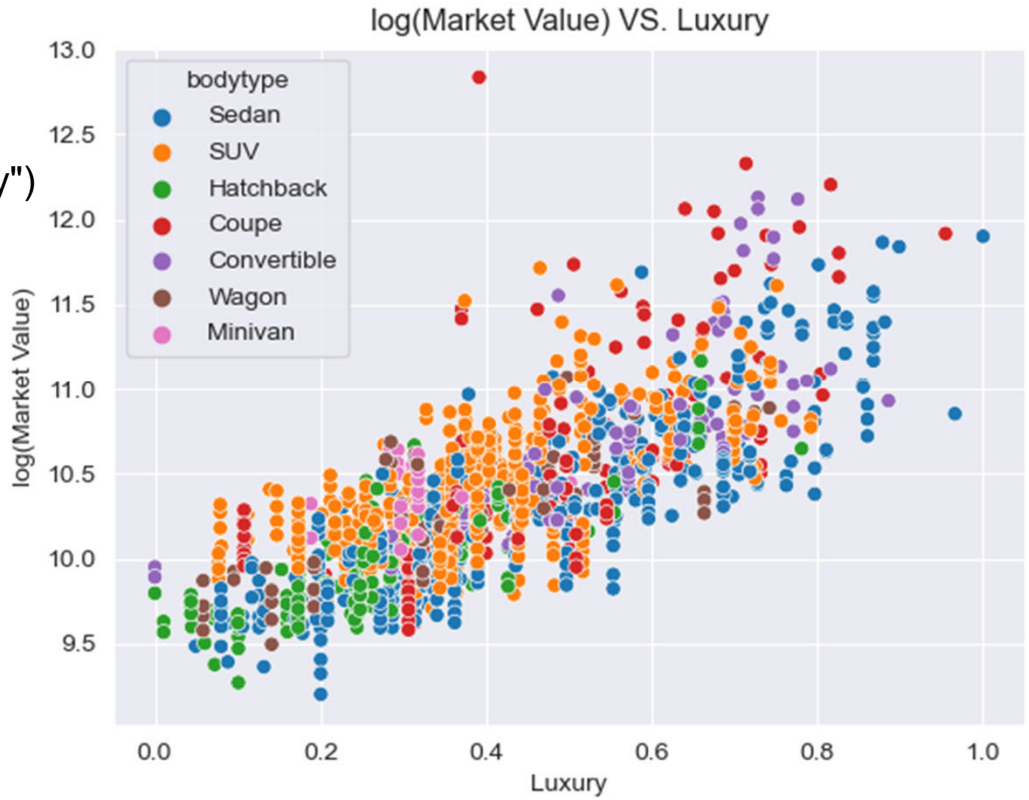
```
# Scatterplot of Market Value vs Luxury metric
plt.figure()
ax=sns.scatterplot(
                x           = "luxury",
                y           = "log_emv",
                data        = cars,
                color       = "blue"   # or "pink", "black", etc.
                )

# Add title and axis labels
ax.set(xlabel  = 'Luxury',
       ylabel  = 'log(Market Value)',
       title   = "log(Market Value) VS. Luxury")
```



UCLAAnderson
SCHOOL of MANAGEMENT

c. Plotting three dimensions

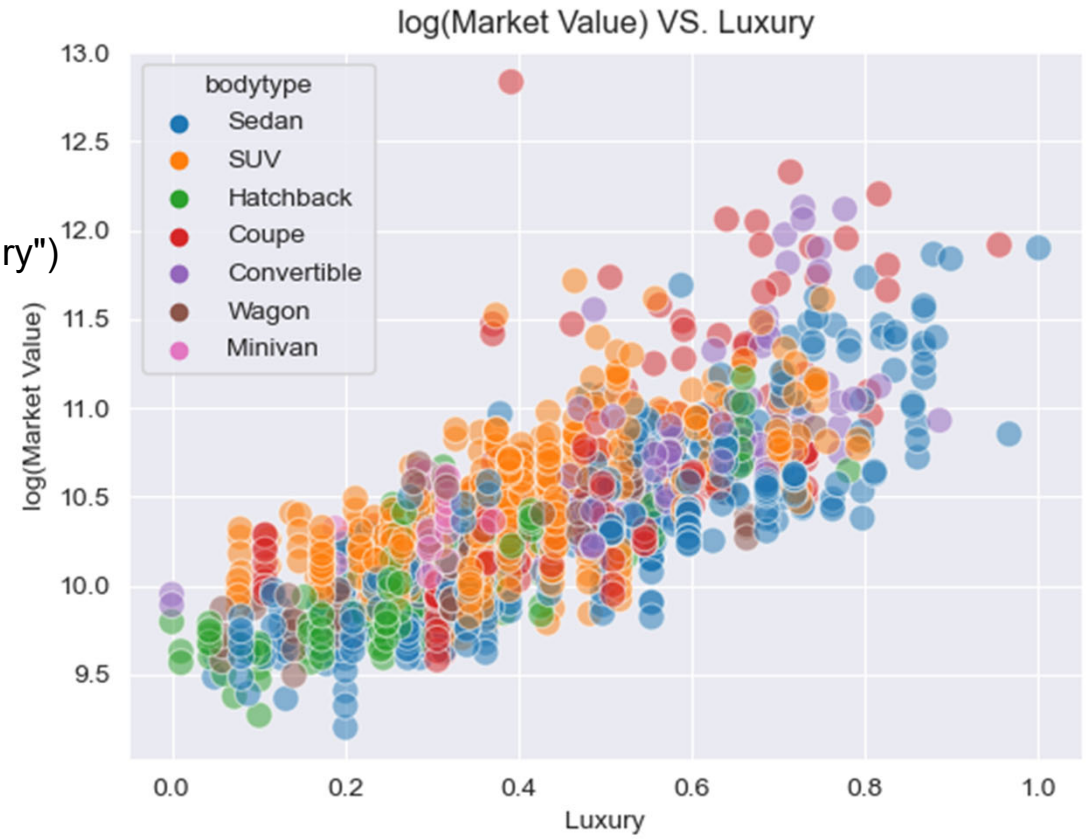
```
# Scatterplot of Market Value vs Luxury metric but with color indicating bodytype
plt.figure()
ax=sns.scatterplot(
                x     = "luxury",
                y     = "log_emv",
                data  = cars,
                color = "blue",
                hue   = "bodytype")

ax.set(xlabel  = 'Luxury',
       ylabel  = 'log(Market Value)',
       title     = "log(Market Value) VS. Luxury")
```



UCLAAnderson
SCHOOL of MANAGEMENT

c. Plotting three dimensions

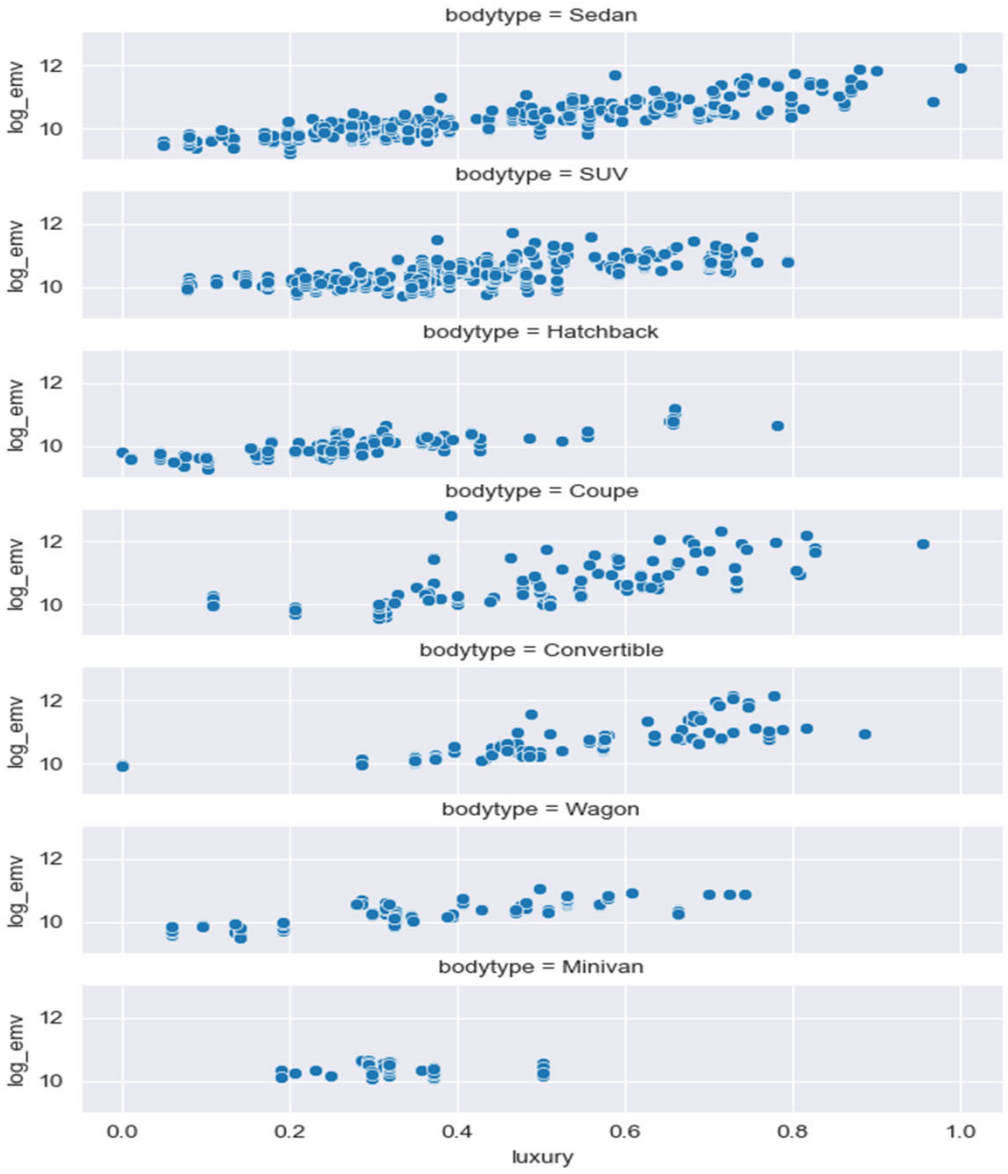
```
# Change size and transparency of points
plt.figure()
ax=sns.scatterplot(
                x          = "luxury",
                y          = "log_emv",
                data       = cars,
                color      = "blue",
                hue        = "bodytype",
                alpha      = 0.5,
                s          = 100)

ax.set(xlabel = 'Luxury',
       ylabel = 'log(Market Value)',
        title  = "log(Market Value) VS. Luxury")
```



UCLAAnderson
SCHOOL of MANAGEMENT

c. Facets: alternative representation

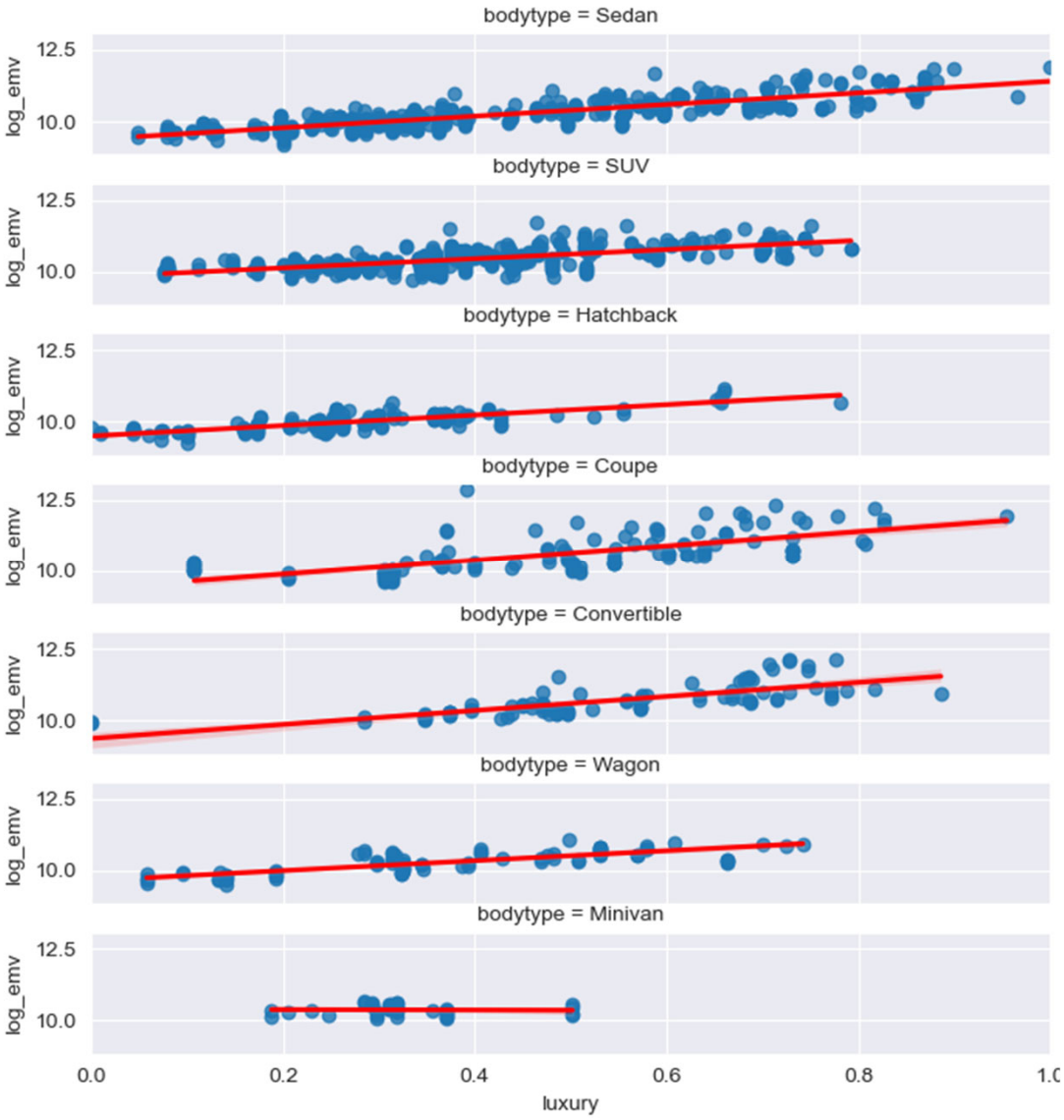
```
# Individual scatterplots for each bodytype
ax = sns.FacetGrid(cars,  row="bodytype")
ax.map(sns.scatterplot, "luxury", "log_emv")
```



UCLAAnderson
SCHOOL of MANAGEMENT

c. Adding fitted line

```
# Individual scatterplots for each bodytype
# with a red fitted line
ax = sns.FacetGrid(cars,  row="bodytype")
ax.map(sns.regplot, "luxury",
       "log_emv",line_kws={"color": "red"})
```

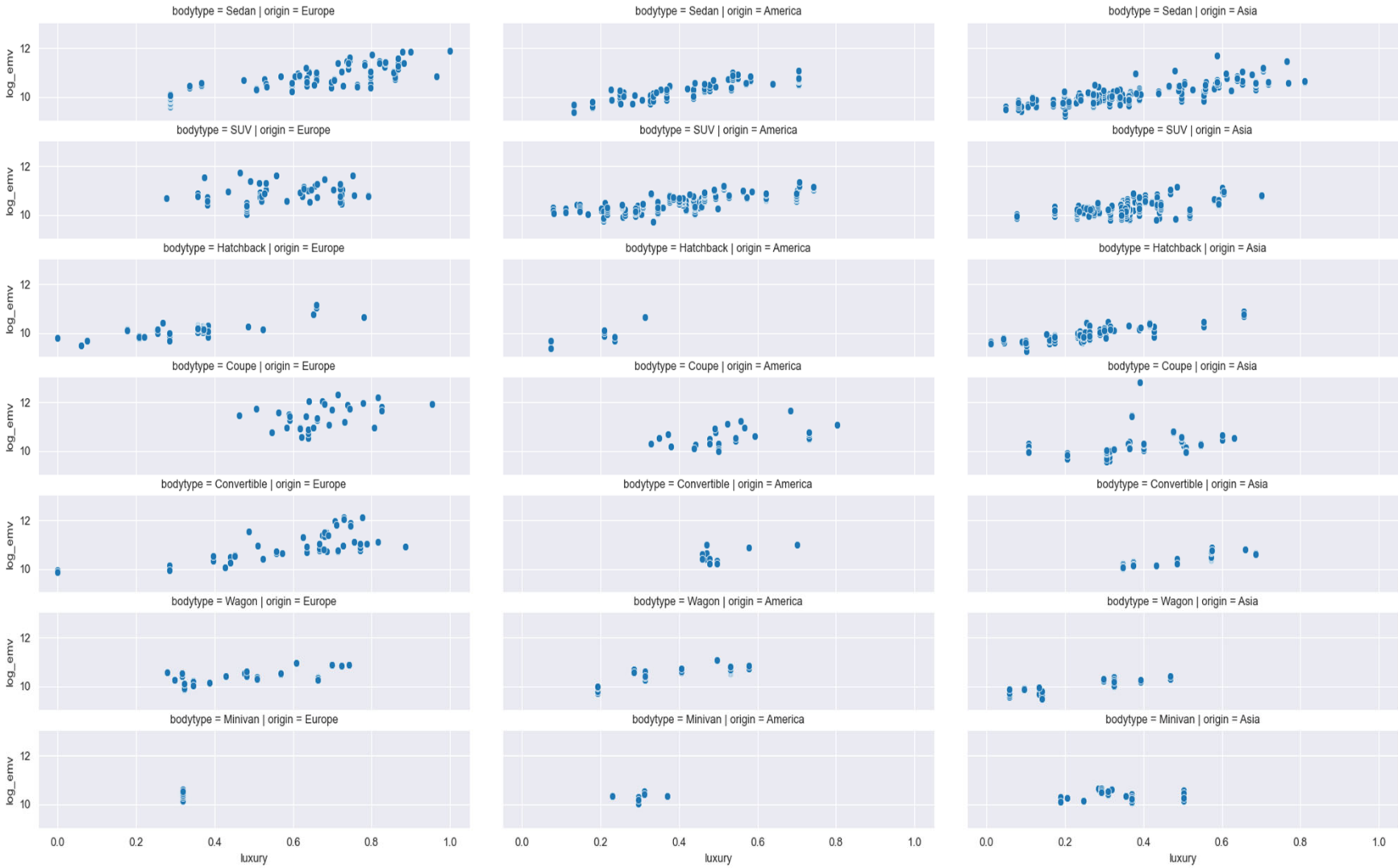


Next slide, increase dimensionality:

```
# Individual scatterplots for each
# bodytype *** and origin ***
ax = sns.FacetGrid(cars, col="origin",
                   row="bodytype")
ax.map(sns.scatterplot, "luxury", "log_emv")
```

UCLAAnderson
SCHOOL of MANAGEMENT

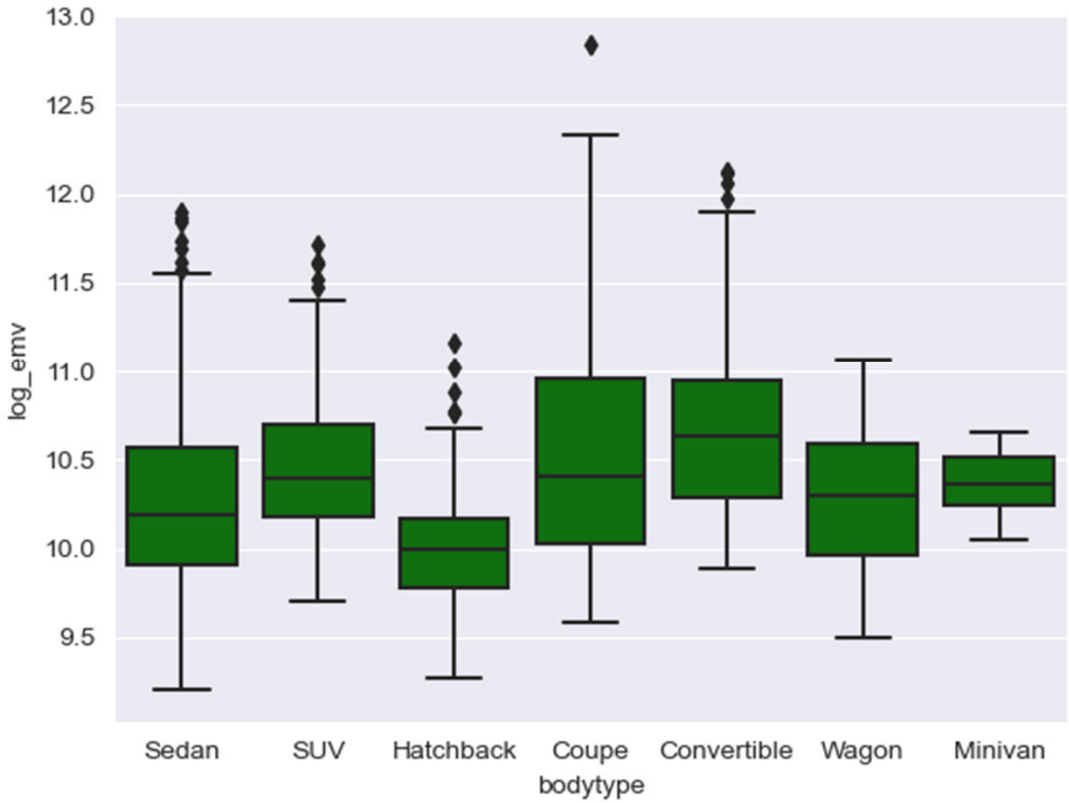
c. Increasing dimensionality again



UCLAAnderson
SCHOOL of MANAGEMENT

c. Box plots

```
# Relate a continuous variable like emv to a categorical variable like bodytype
plt.figure()
ax = sns.boxplot(x="bodytype",  y="log_emv", data=cars,color="green")

# If you want different colors
plt.figure()
ax = sns.boxplot(x="bodytype", y="log_emv", data=cars, palette="Set3")
```



UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to DataFrame

`DataFrame` is an object that allows easy manipulation of data.

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 DataFrame

```
# Create a DataFrame
sales_table = pd.DataFrame({"Module":["a","a","c","c","c","b"],
                            "upc":[2,3,4,5,5,1],
                            "sales":np.random.randint(0,10000,6)})
```

From prompt:

In [9]: sales_table

Out[9]:

	Module	upc	sales
0	a	2	9181
1	a	3	6090
2	c	4	8354
3	c	5	634
4	c	5	5539
5	b	1	4345

UCLAAnderson
SCHOOL of MANAGEMENT

d. Introduction to DataFrame

```
# show all DataFrames in the workspace
all_dataframes = [var for var in dir() if isinstance(eval(var), pd.core.frame.DataFrame)]
print('List of objects of class DataFrame in workspace: ', all_dataframes)

>> List of objects of class DataFrame in workspace:  ['cars', 'mvehicles', 'sales_table']
```

Indexing in Python is a little strange (to me!)

```
a[start:stop]  # items start through stop-1
a[start:]      # items start through the rest of the array
a[:stop]       # items from the beginning through stop-1
a[:]           # a copy of the whole array
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Indexing examples

sales_table[1:3]

Out[28]:

	Module	upc	sales
1	a	3	6090
2	c	4	8354

sales_table[sales_table.upc>3]

Out[29]:

	Module	upc	sales
2	c	4	8354
3	c	5	634
4	c	5	5539

sales_table.iloc[2:3,2]

Out[30]:

2 8354

Name: sales, dtype: int32

UCLAAnderson
SCHOOL of MANAGEMENT

d. Select rows based on column values

sales_table[sales_table.Module.isin(['a','b'])]

Out[33]:

	Module	upc	sales
0	a	2	9181
1	a	3	6090
5	b	1	4345

sales_table[~sales_table.Module.isin(['a','c'])]

Out[34]:

	Module	upc	sales
5	b	1	4345

sales_table[np.logical_and(sales_table.Module.isin(['a','c']), sales_table.upc.isin([3,4]))]

Out[35]:

	Module	upc	sales
1	a	3	6090
2	c	4	8354

UCLAAnderson
SCHOOL of MANAGEMENT

d. Select, get, or add columns

```
# Get an entire column
sales_table.Module
# Fetch multiple columns
sales_table[['Module','upc']]
# Create a new column that contains the log transformation of the sales column
sales_table['lnsales']=np.log(sales_table['sales'])
# Create a new DataFrame with aggregated sales over each category in the module column
sales_module=pd.DataFrame({'Module'  :sales_table.Module.unique(),
                    'agg_sales':sales_table.groupby('Module')['sales'].transform('sum').unique() })

>> sales_module
Out[37]:
  Module  agg_sales
0      a      15271
1      c      14527
2      b       4345
```

UCLAAnderson
SCHOOL of MANAGEMENT

d. Merge DataFrames

```
# Add a new DataFrame with description of each module category
module_description=pd.DataFrame({"Module"     : ["a","b","c"],
                                 "Description": ["meats","milk","cereal"]})

# Merge the two DataFrames by the module column
pd.merge(sales_module, module_description, on='Module')
```

Out[38]:

	Module	agg_sales	Description
0	a	15271	meats
1	c	14527	cereal
2	b	4345	milk

UCLAAnderson
SCHOOL of MANAGEMENT

Visualization and Model Building

e. Model specification and visualization

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

CSV-file StockRetAcct.csv

- 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 DataFrame 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 DataFrame

```
# additional package to deal with rank sorts
from  scipy.stats import rankdata

# Load data
StockRetAcct_DT = pd.read_csv("StockRetAcct_DT.csv")

# Look at top (head) of data set
StockRetAcct_DT.head()
Out[48]:
Unnamed:  0  FirmID  year  lnAnnRet  ...    lnROE       rv          lnBM      ff_ind
0             1      6     1980  0.363631  ...  0.095294  0.084134  0.633391     3.0
1             2      6     1981 -0.290409  ...  0.082180  0.056381  0.356723     3.0
2             3      6     1982  0.186630  ...  0.079516  0.062072  0.779405     3.0
3             4      6     1983  0.489819  ...  0.055374  0.076955  0.702113     3.0
4             5     10     1991 -0.508005  ...  0.146828  0.374368 -2.160942    10.0

[5 rows x 17 columns]

# create excess returns
StockRetAcct_DT['ExRet'] = np.exp(StockRetAcct_DT.lnAnnRet) - np.exp(StockRetAcct_DT.lnRf)
```

UCLAAnderson
SCHOOL of MANAGEMENT

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

Natural to simply do a scatter plot, but not that revealing due to noise

```
plt.figure()
ax   = sns.scatterplot(x     = "lnBM",
                       y     = "ExRet",
                       data  = StockRetAcct_DT,
                       color = "blue" )
ax.set(title = 'Log Book-to-Market vs. Excess Return')
```

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

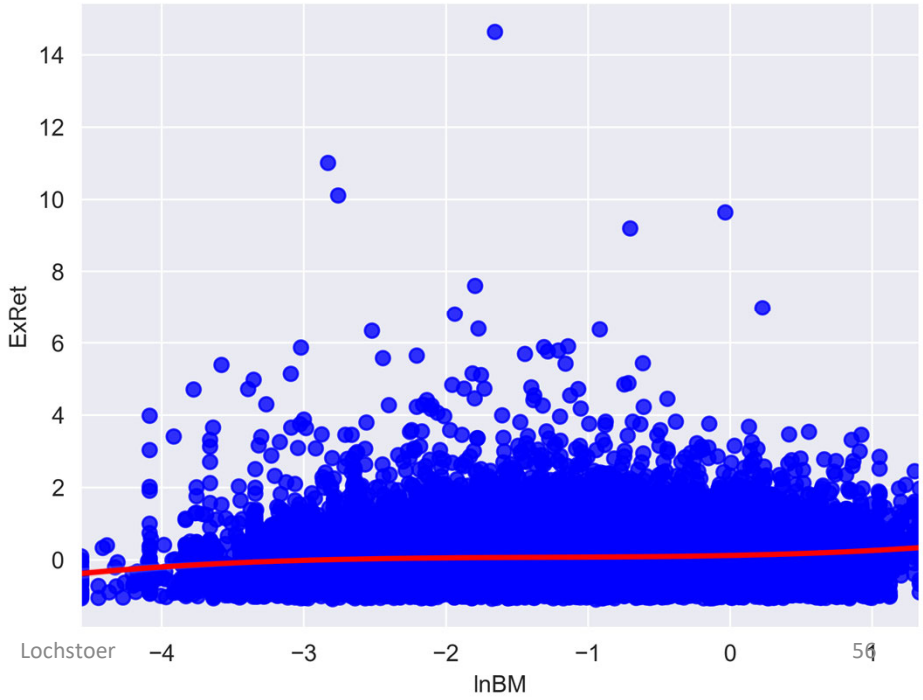
e. Consider the "Value vs. Growth" signal: lnBM

Let's fit a red smooth (possibly nonlinear) line using sns.regplot

```
# Fit a non-linear line to the points. 3rd order polynomial
plt.figure()
ax = sns.regplot(x     = "lnBM",
                 y     = "ExRet",
          data  = StockRetAcct_DT,
          scatter_kws={"color": "blue"},
          line_kws={"color": "red"},
          fit_reg=True,
          order = 3)
```

Can see slight positive relation

- Still not very informative
- Let's zoom in

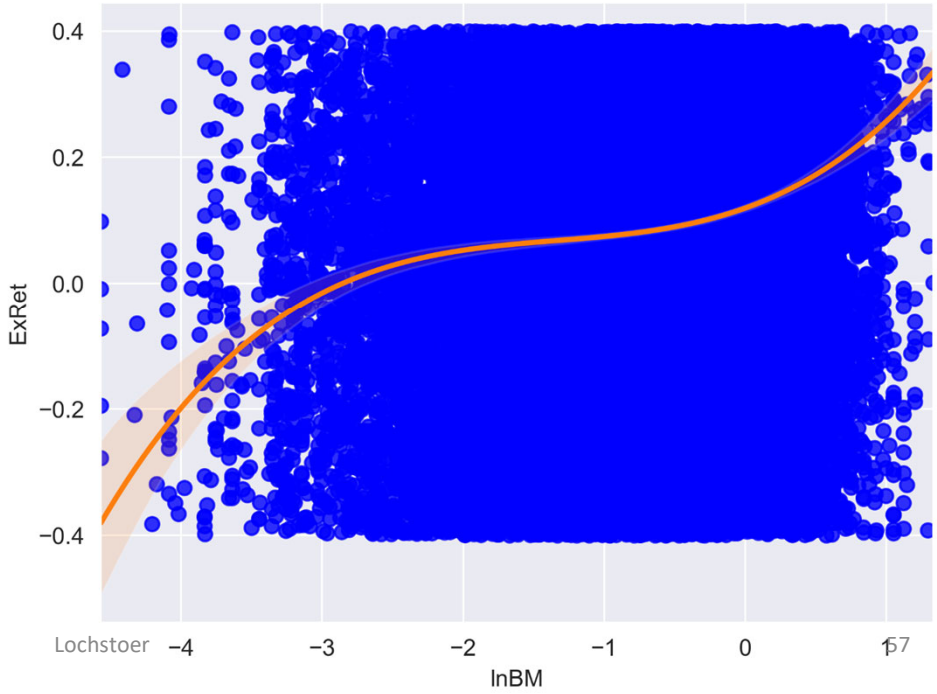


UCLAAnderson
SCHOOL of MANAGEMENT

e. Expected Return and lnBM, zooming in

```
# Zoom in for scatter plot, define truncated dataset
local_stock_data = StockRetAcct_DT[abs(StockRetAcct_DT.ExRet)<=.4 ]
# plot scatter plot with truncated dataset
plt.figure()
ax = sns.regplot (x     = "lnBM", y     = "ExRet",
                  data  = local_stock_data,
                  scatter_kws={"color": "blue"}, fit_reg=False)
# next, add regression line fitted on all data
sns.regplot(data=StockRetAcct_DT, x = "lnBM", y="ExRet", order = 3,scatter=False, ax=ax)'
```

- *Somewhat nonlinear pattern*
- *Large quantities, expected excess return from -35% 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
  - Python 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)

```
# Get quantiles
StockRetAcct_DT['bm_vingtile']=pd.qcut(StockRetAcct_DT['lnBM'], 20, labels=np.arange(1, 21, 1))
# Drop missing observations
StockRetAcct_DT = StockRetAcct_DT[StockRetAcct_DT['bm_vingtile'].notna()]
# mean excess return by vingtile
EW_BM_MutualFunds=StockRetAcct_DT.groupby('bm_vingtile')['ExRet'].mean()
# Plot the different portfolios against their excesss return
plt.figure()
ax=sns.regplot(x=np.arange(1,21),y=EW_BM_MutualFunds,
                fit_reg=True,order=3)
ax.set(title   = "Log Book-to-Market vs. Excess Returns")
```

- 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!

```
# Sort into 20 bins (vingtiles) by the book-to-market value within year
StockRetAcct_DT['bm_vingtile_yr']=np.nan
for year in range(1981,2015):
    dt   = StockRetAcct_DT[StockRetAcct_DT['year']==year]
    inds = np.where(StockRetAcct_DT['year']==year)
    StockRetAcct_DT['bm_vingtile_yr'].iloc[inds] = pd.qcut(dt['lnBM'], 20, labels=np.arange(1, 21, 1)).values.to_list()

# Remove empty observations due to missing lnBM observations
StockRetAcct_DT = StockRetAcct_DT[StockRetAcct_DT['bm_vingtile_yr'].notna()]

# Sort into portfolios for each year vingtile in each year and take the average, then average over years.
EW_BM_MutualFunds_yr=pd.DataFrame({'bm_vingtile_yr':np.tile(range(1,21), 34),
                                   'year':np.sort(np.tile(range(1981,2015),20)),
                                   'MeanExRetYr':np.empty(((2014-1981+1)*20))})
for year in range(1981,2015):
    dt   = StockRetAcct_DT[StockRetAcct_DT['year']==year]
    inds = np.where(EW_BM_MutualFunds_yr['year']==year)
    EW_BM_MutualFunds_yr['MeanExRetYr'].iloc[inds] = list(dt.groupby('bm_vingtile_yr')['ExRet'].mean())

# Finally take the average for each bin across years
EW_BM_MutualFunds_yr = EW_BM_MutualFunds_yr.groupby('bm_vingtile_yr')['MeanExRetYr'].mean()
```

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

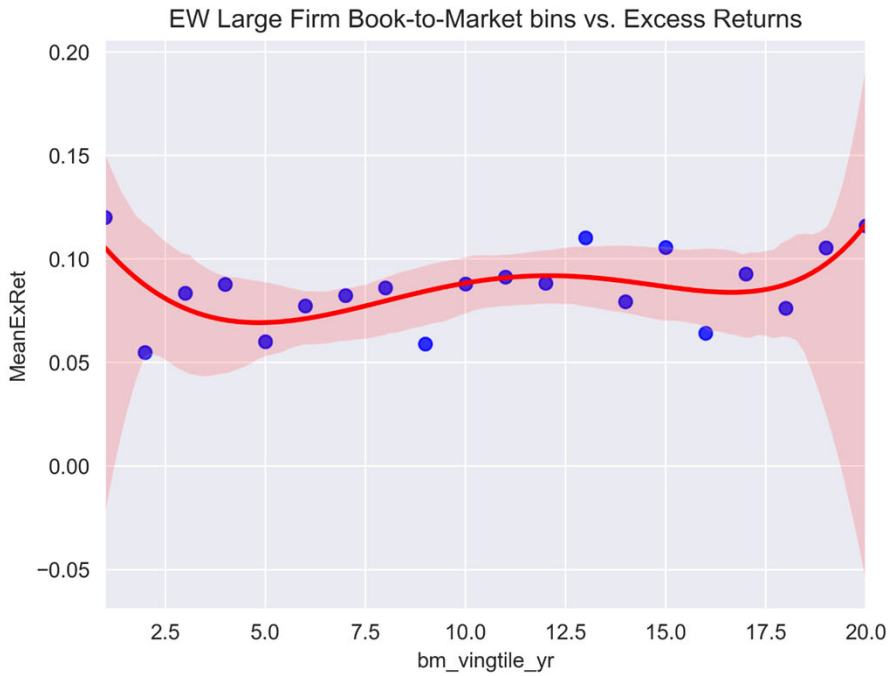
- The relation look linear, but we have done a lot of averaging across stocks
  - Such averaging can hide important patterns. Let's create conditional sorts, conditioning on whether the firm is one of the 500 biggest firms (large) or not (small)

```
StockRetAcct_DT['LargeStock']=np.nan
for year in range(1981,2015):
    dt   = StockRetAcct_DT[StockRetAcct_DT['year']==year]
    inds = np.where(StockRetAcct_DT['year']==year)
    StockRetAcct_DT['LargeStock'].iloc[inds] = rankdata(-dt['lnME'])<=500

EW_BM_MutualFunds_yr_Large=pd.DataFrame({'bm_vingtile_yr':np.arange(1, 21, 1),
                            'MeanExRet':np.empty((20))})
EW_BM_MutualFunds_yr_Small=pd.DataFrame({'bm_vingtile_yr':np.arange(1, 21, 1),
                            'MeanExRet':np.empty((20))})
StockRetAcct_DT_Large=StockRetAcct_DT[StockRetAcct_DT['LargeStock']==1]
StockRetAcct_DT_Small=StockRetAcct_DT[StockRetAcct_DT['LargeStock']==0]

ret_l,ret_s = np.empty([]),np.empty([])
for year in range(1981,2015):
    dt_l = StockRetAcct_DT_Large[StockRetAcct_DT_Large['year']==year]
    dt_s = StockRetAcct_DT_Small[StockRetAcct_DT_Small['year']==year]
    for bin in range(1,21):
      np.mean(dt_l[dt_l['bm_vingtile_yr']==bin].ExRet)
      np.mean(dt_s[dt_s['bm_vingtile_yr']==bin].ExRet)
      ret_l=np.append(ret_l, np.mean(dt_l[dt_l['bm_vingtile_yr']==bin].ExRet))
      ret_s=np.append(ret_s, np.mean(dt_s[dt_s['bm_vingtile_yr']==bin].ExRet))

EW_BM_MutualFunds_yr_Large['MeanExRet'] = np.mean(np.reshape(ret_l[1:],(34,20)),axis=0)
EW_BM_MutualFunds_yr_Small['MeanExRet'] = np.mean(np.reshape(ret_s[1:],(34,20)),axis=0)
```

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

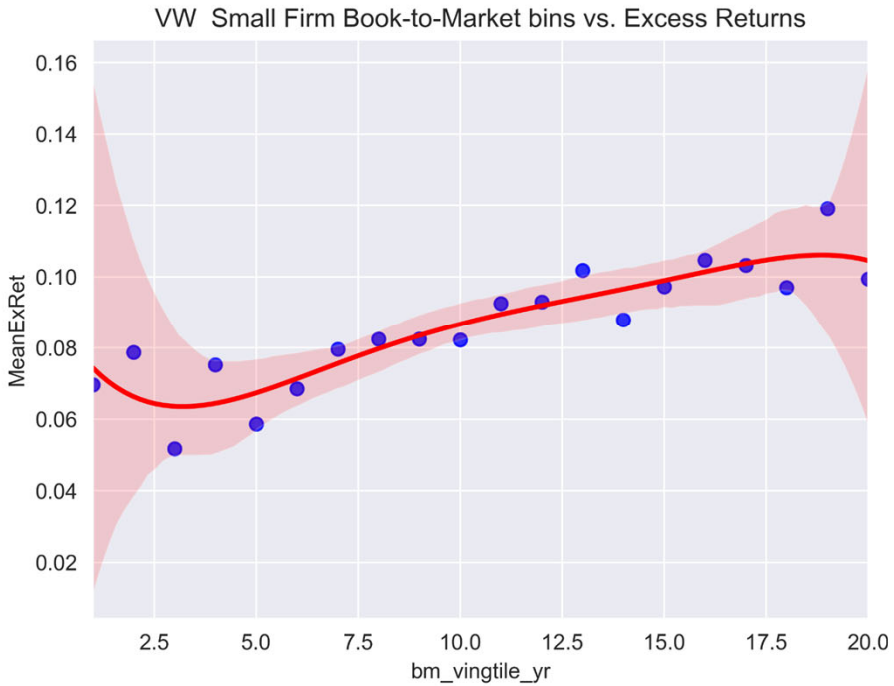
- Note that we value-weight within each portfolio each year, relying on beginning-of-year market values of the stocks (MEwt)
  - ***We then equal-weight across the years***

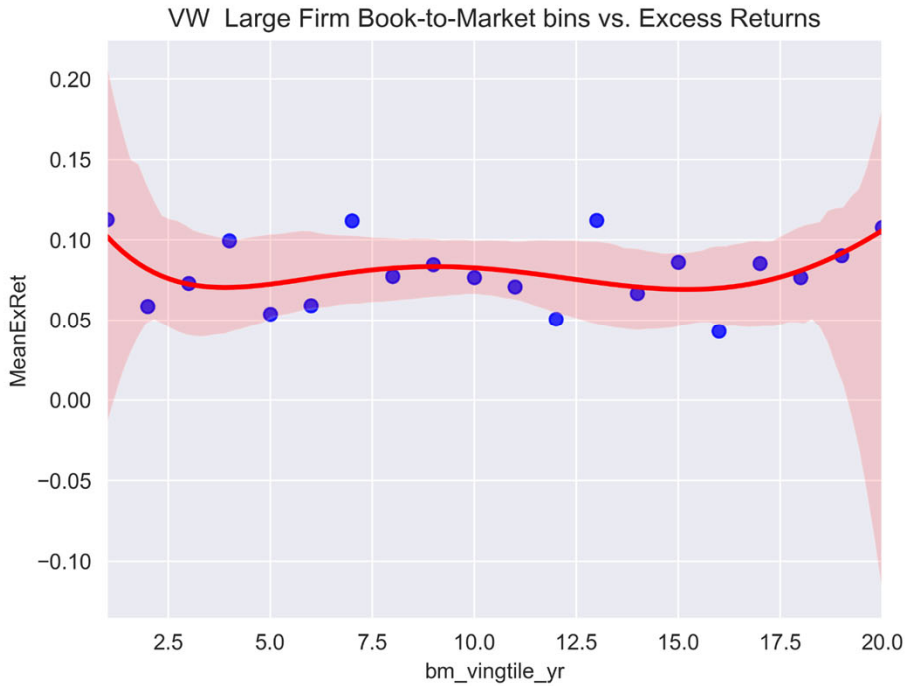
```
ret_l,ret_s = np.empty([]),np.empty([])
for year in range(1981,2015):
   dt_l  = StockRetAcct_DT_Large[StockRetAcct_DT_Large['year']==year]
   dt_s  = StockRetAcct_DT_Small[StockRetAcct_DT_Small['year']==year]
   for bin in range(1,21):
      wt_l  = dt_l[dt_l.bm_vingtile_yr==bin].Mewt / dt_l[dt_l.bm_vingtile_yr==bin].MEwt.sum()
      wt_s  = dt_s[dt_s.bm_vingtile_yr==bin].Mewt / dt_s[dt_s.bm_vingtile_yr==bin].MEwt.sum()
      ret_l = np.append(ret_l,sum(dt_l[dt_l.bm_vingtile_yr==bin].ExRet*wt_l))
      ret_s = np.append(ret_s,sum(dt_s[dt_s.bm_vingtile_yr==bin].ExRet*wt_s))

EW_BM_MutualFunds_yr_Large['MeanExRet']=np.mean(ret_l[1:].reshape((34,20)),axis=0)
EW_BM_MutualFunds_yr_Small['MeanExRet']=np.mean(ret_s[1:].reshape((34,20)),axis=0)
```

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
  - Conclusion: *non-linearity* between value, size, and expected returns!





- So, is value-investing more or less dead?