Data Analytics, R, Scaling, and Comet Examples

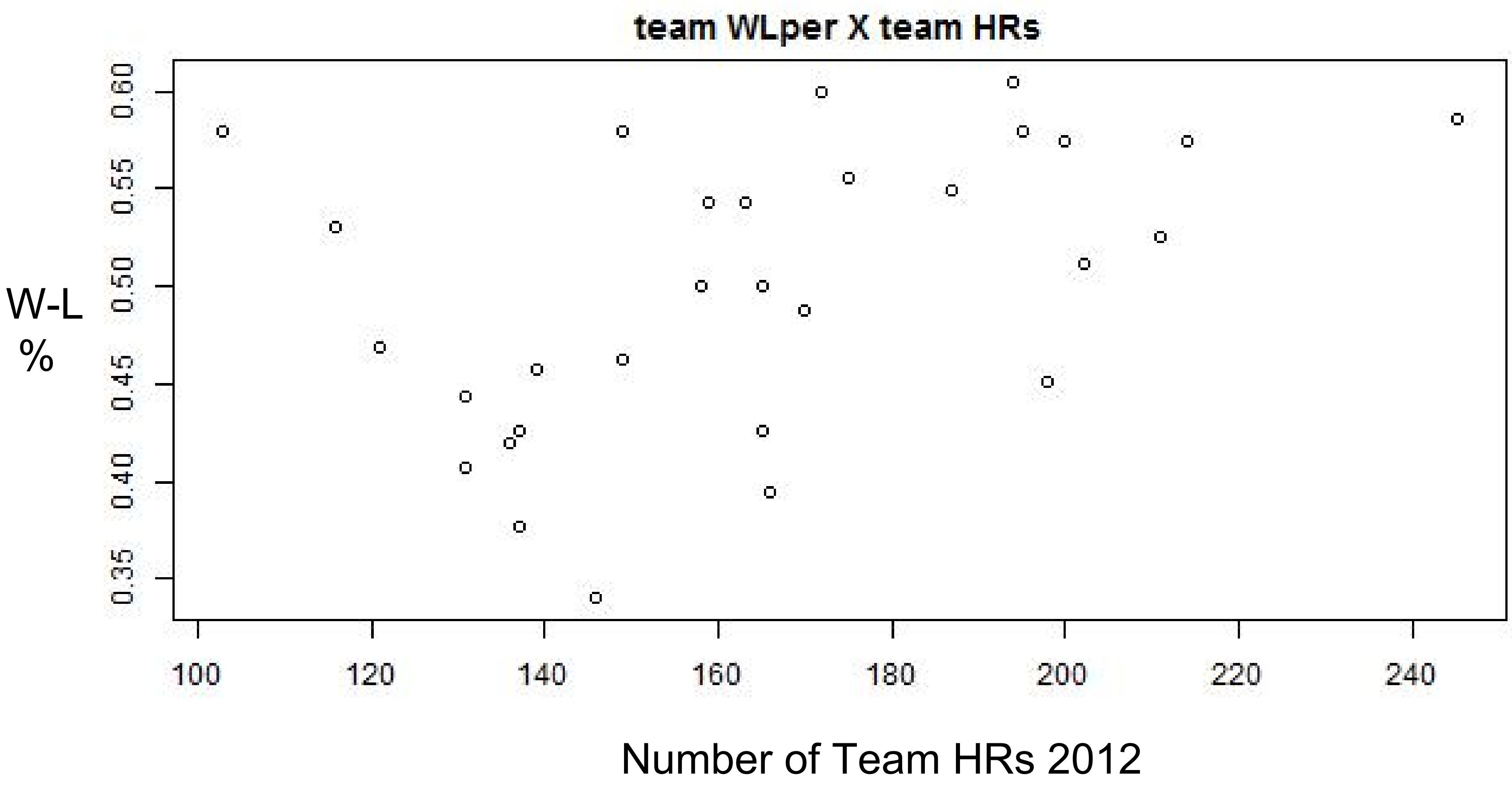
Outline

I. Data Analytics
   - Machine Learning in a Nutshell
   - Data Mining also

II. R, Scaling in R, Parallel R

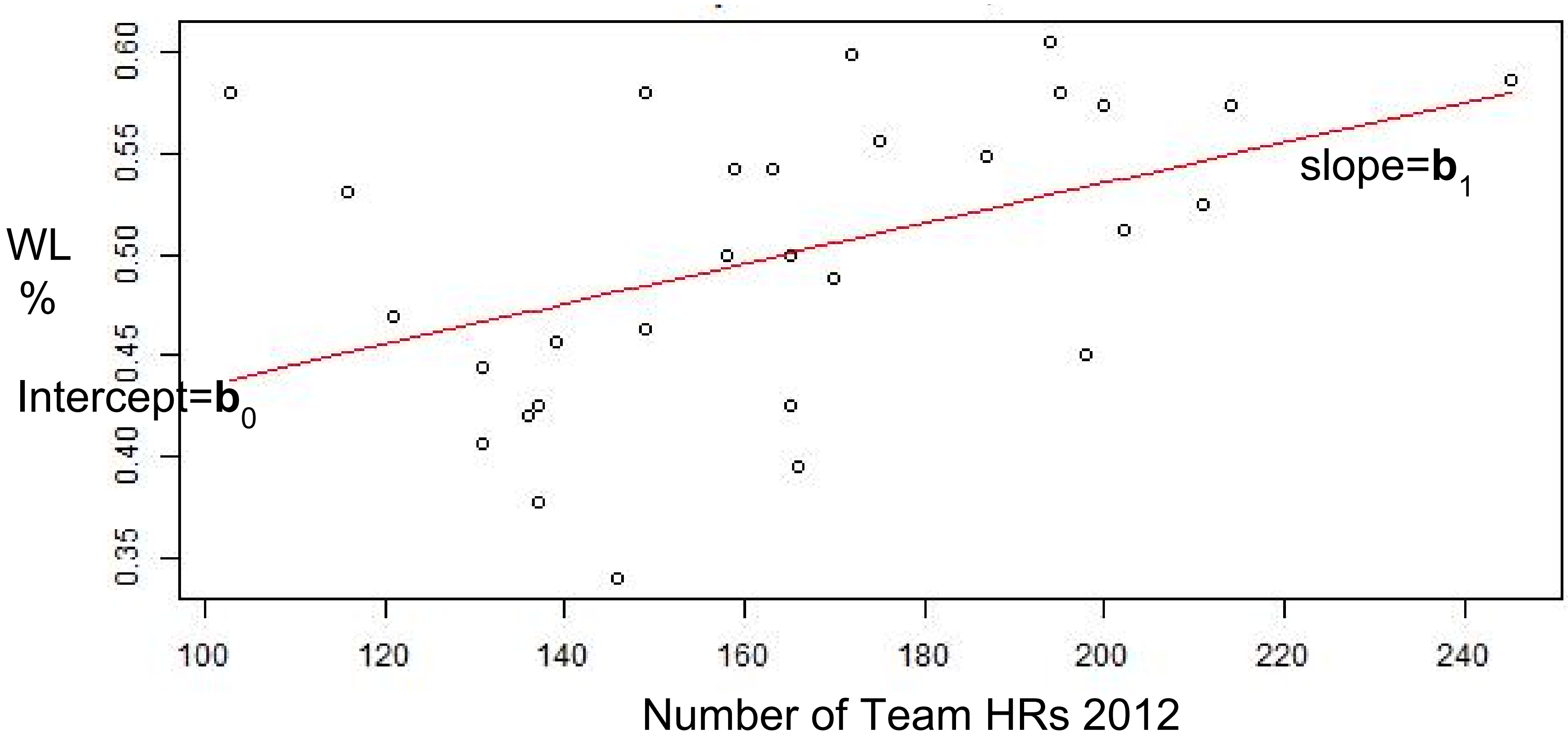
III. Analytics Cases in Comet

A data example:Home Runs and W-L



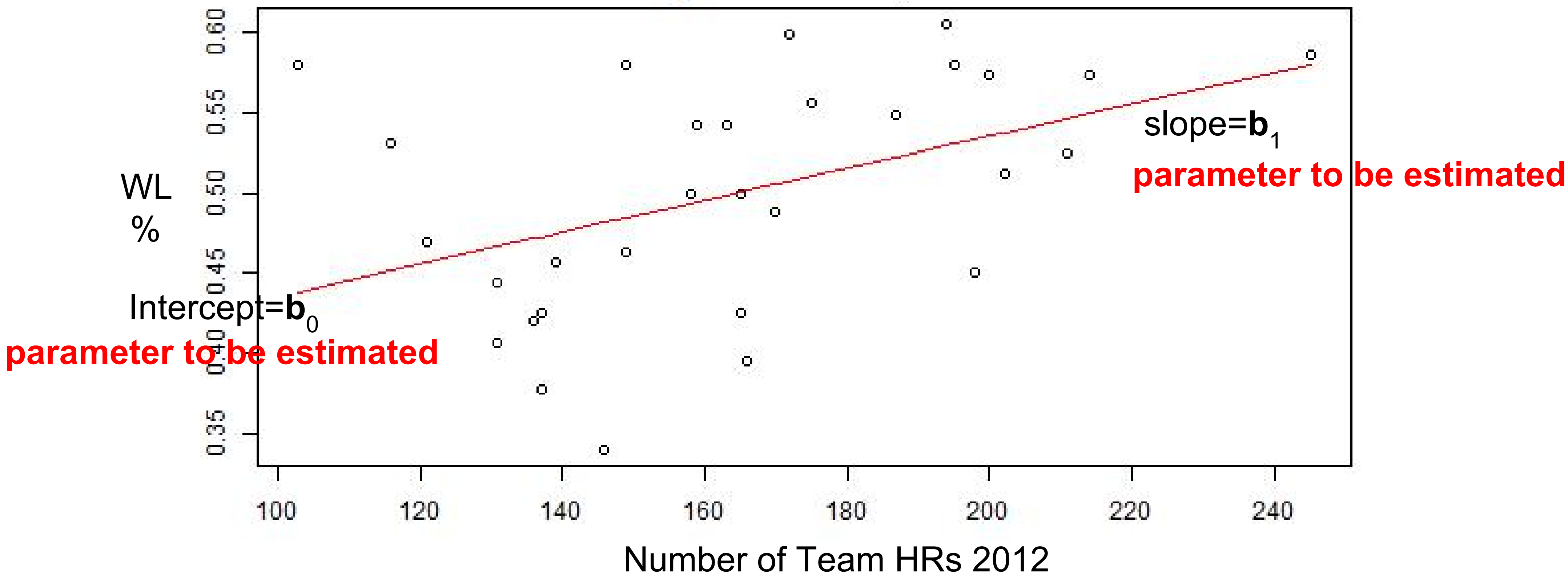
Recall Linear Regression is Fitting a Line

the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i$ (Won-loss percent as function of team home runs)



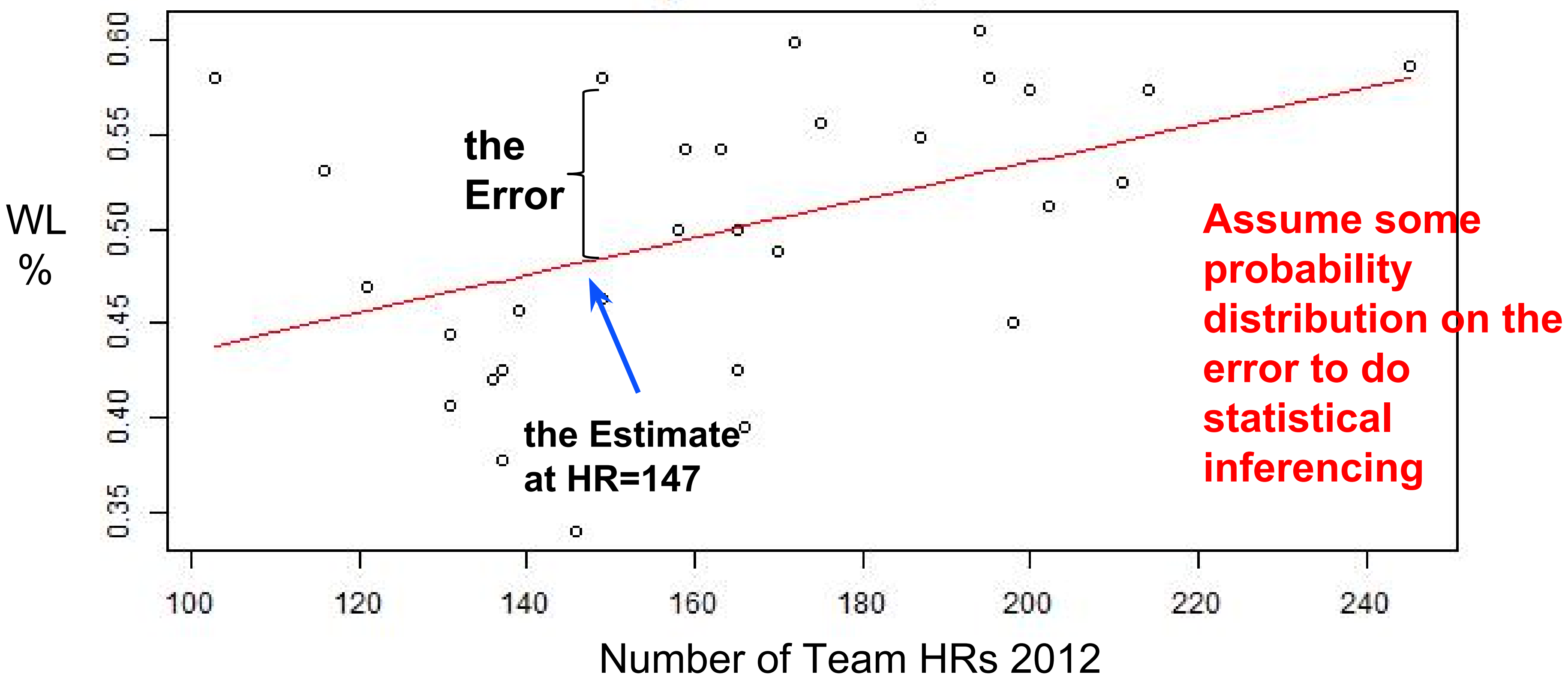
Recall Linear Regression is Fitting a Line

the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i$ (Won-loss percent as function of team home runs)



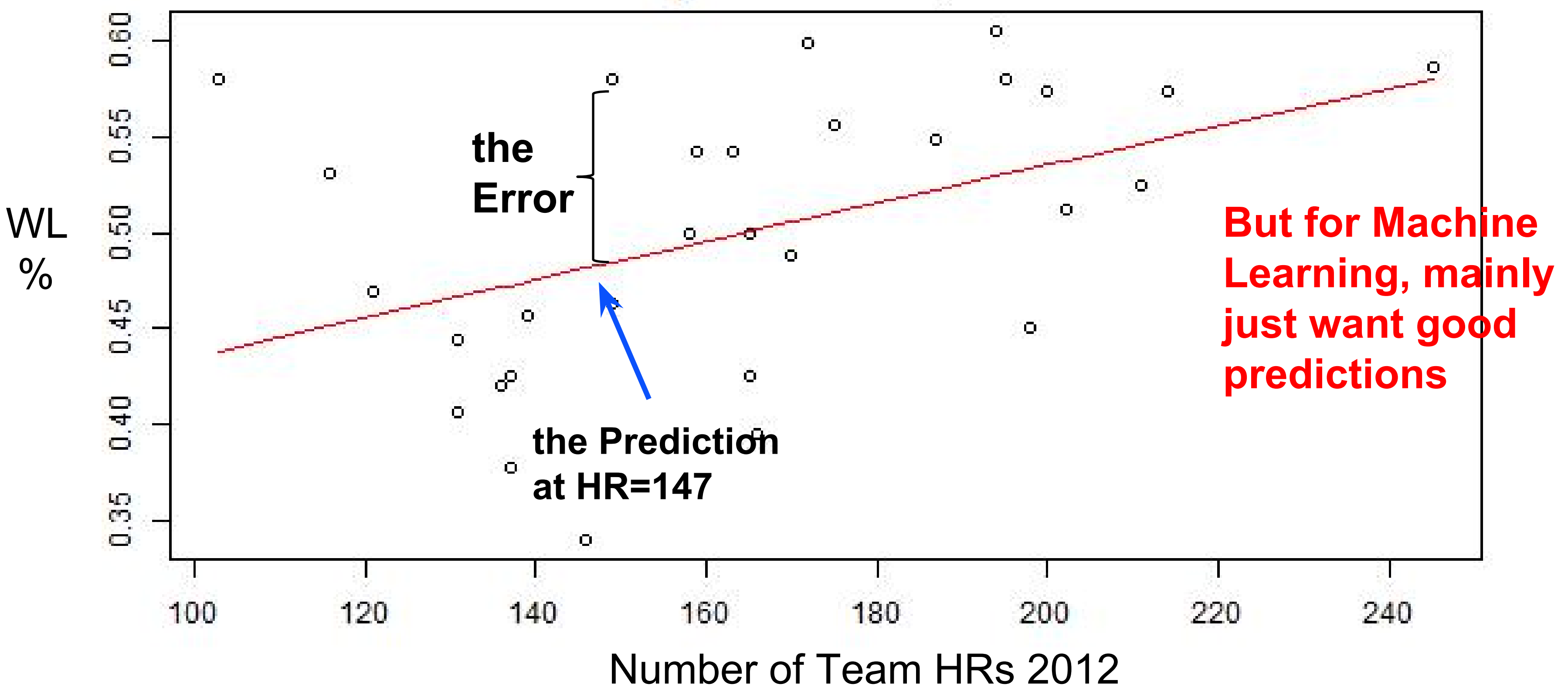
Recall Linear Regression is Fitting a Line

the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i$ + *Gaussian error*



Recall Linear Regression is Fitting a Line

the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i$

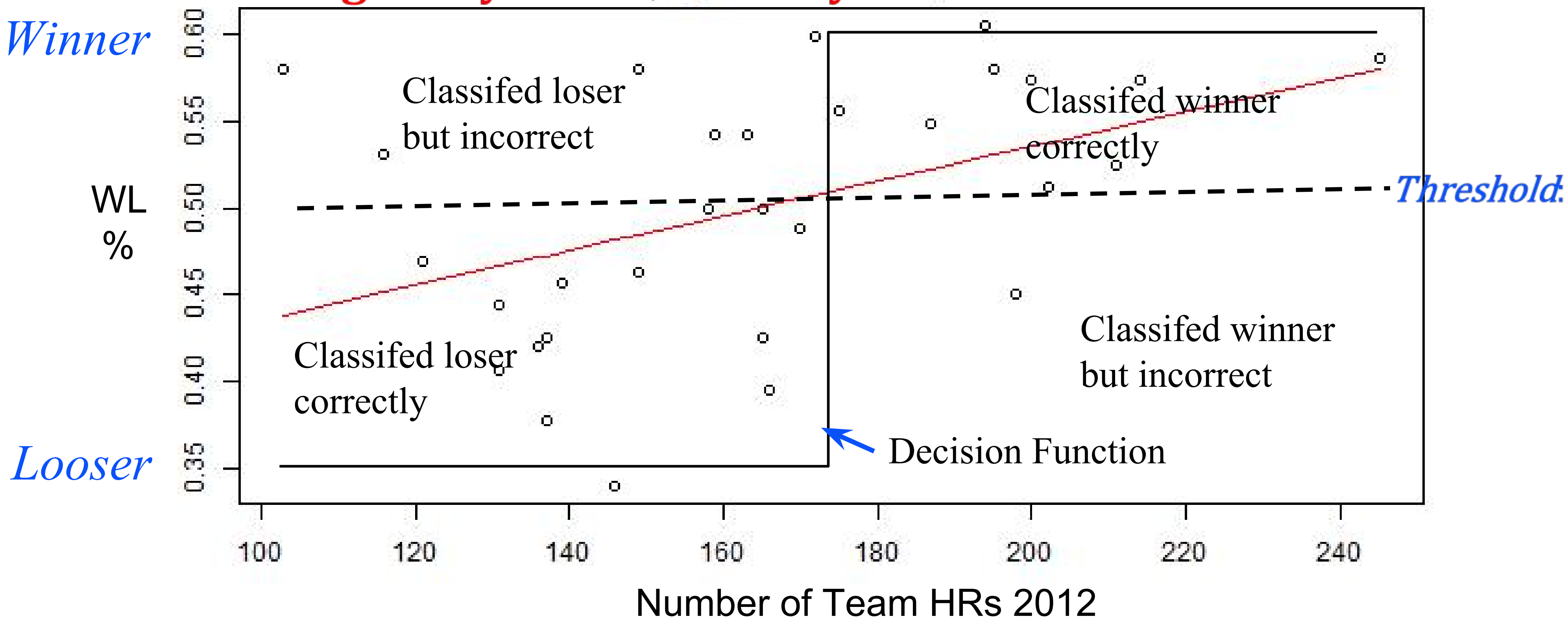


Linear Regression with a Decision Threshold

the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i$

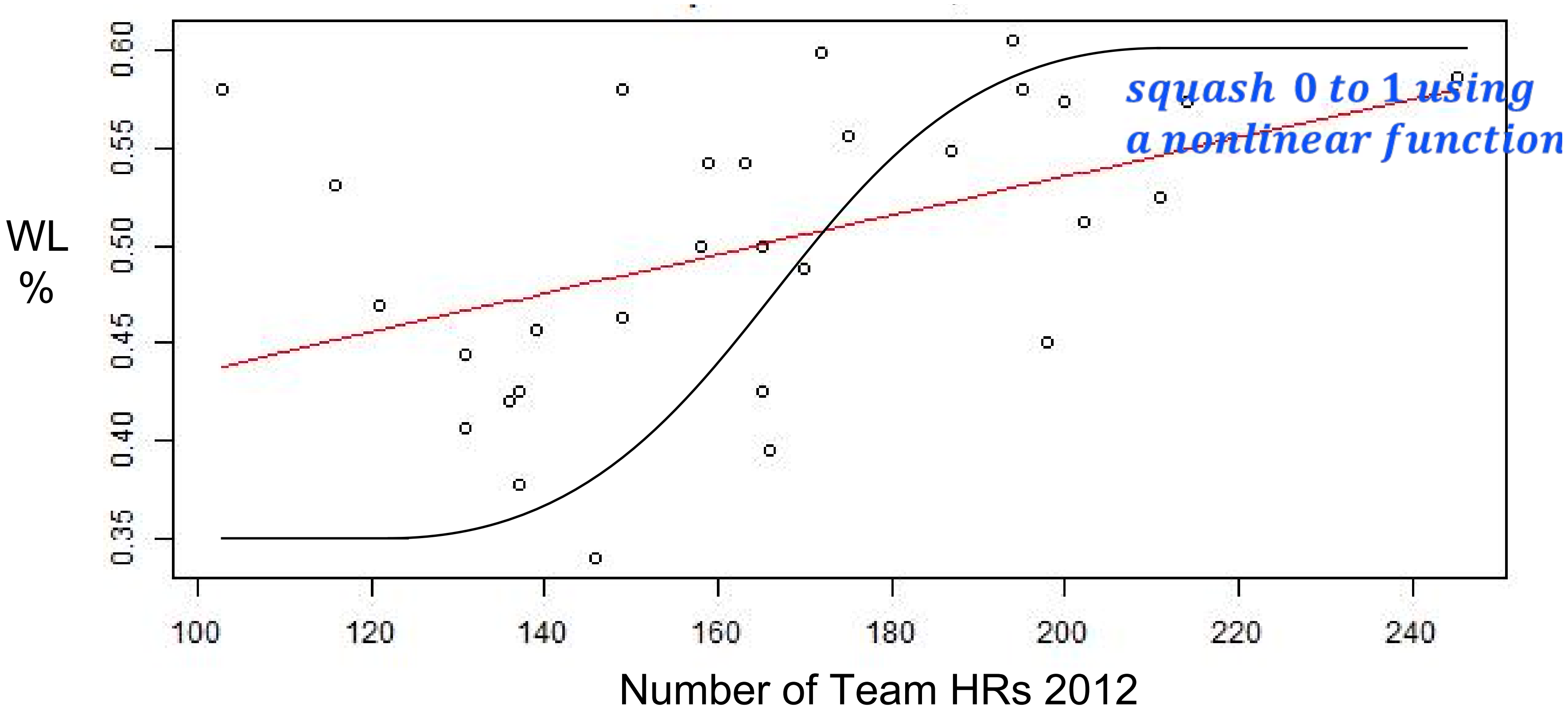
For Classification choose a decision function:

e.g. For y > 0.5, Classify as Winner



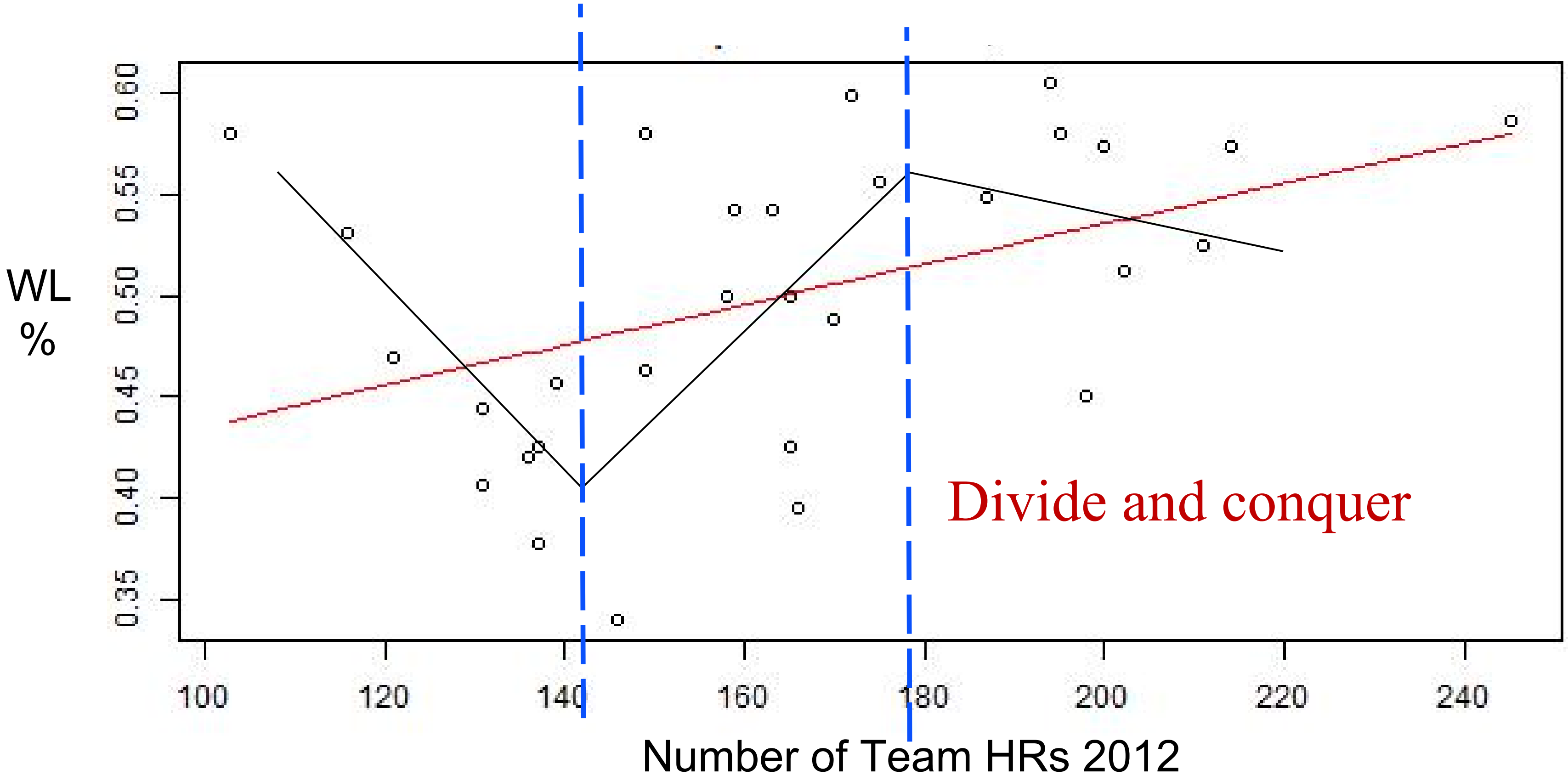
Regression with Logistic Function gives Probability

$$\textbf{the Model}: P(y_i = winner|x_i) = squash\ (b_o * 1 + b_1 * x_i\ )$$



Regression with Logistic Function gives Probability

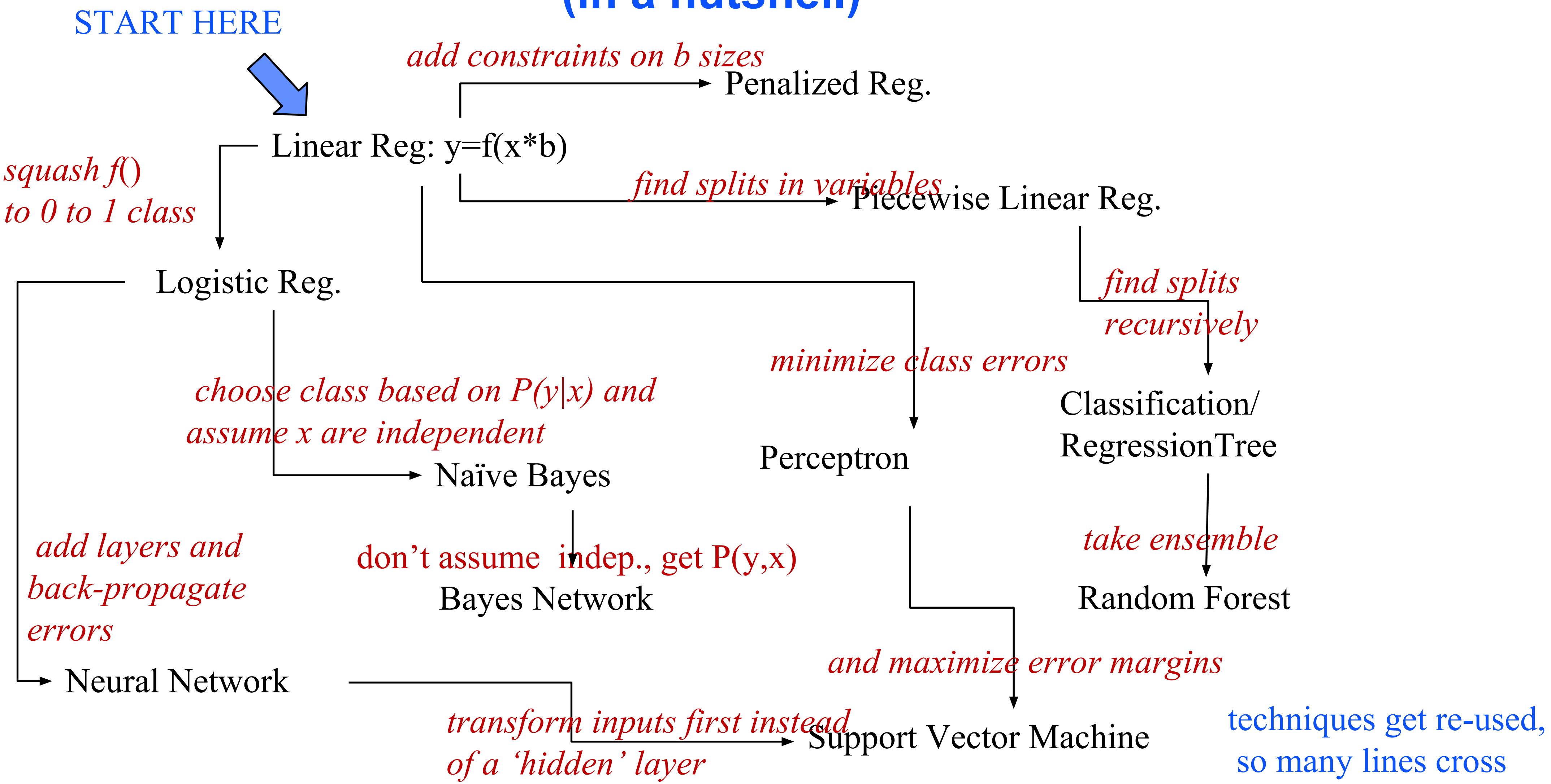
the Model: $y_i = f(x, b) = b_o * 1 + b_1 * x_i \quad \textit{for each x segment}$



Machine Learning Models Are Just Different Functions and Optimizations

- **What kinds of functions to use**
  - E.g. Linear vs NonLinear
  - E.g. divide input into pieces
- **What to Optimize**
  - Minimize Prediction Error
  - Minimize Classification Errors
  - Maximize Probabilities
- **How to Optimize**
  - Solve directly, take derivatives, or search solutions
  - Use constraints and heuristics

A Map of Machine Learning Models
(in a nutshell)



Machine Learning In practice

- **Complexity: more parameters => more complex,**
  - You usually need some heuristics

- **Tradeoff, complexity usually => more potential to overfit (and more sensitivity to data)**

  validation procedures can help

Data Mining refers to Modelling Workflow

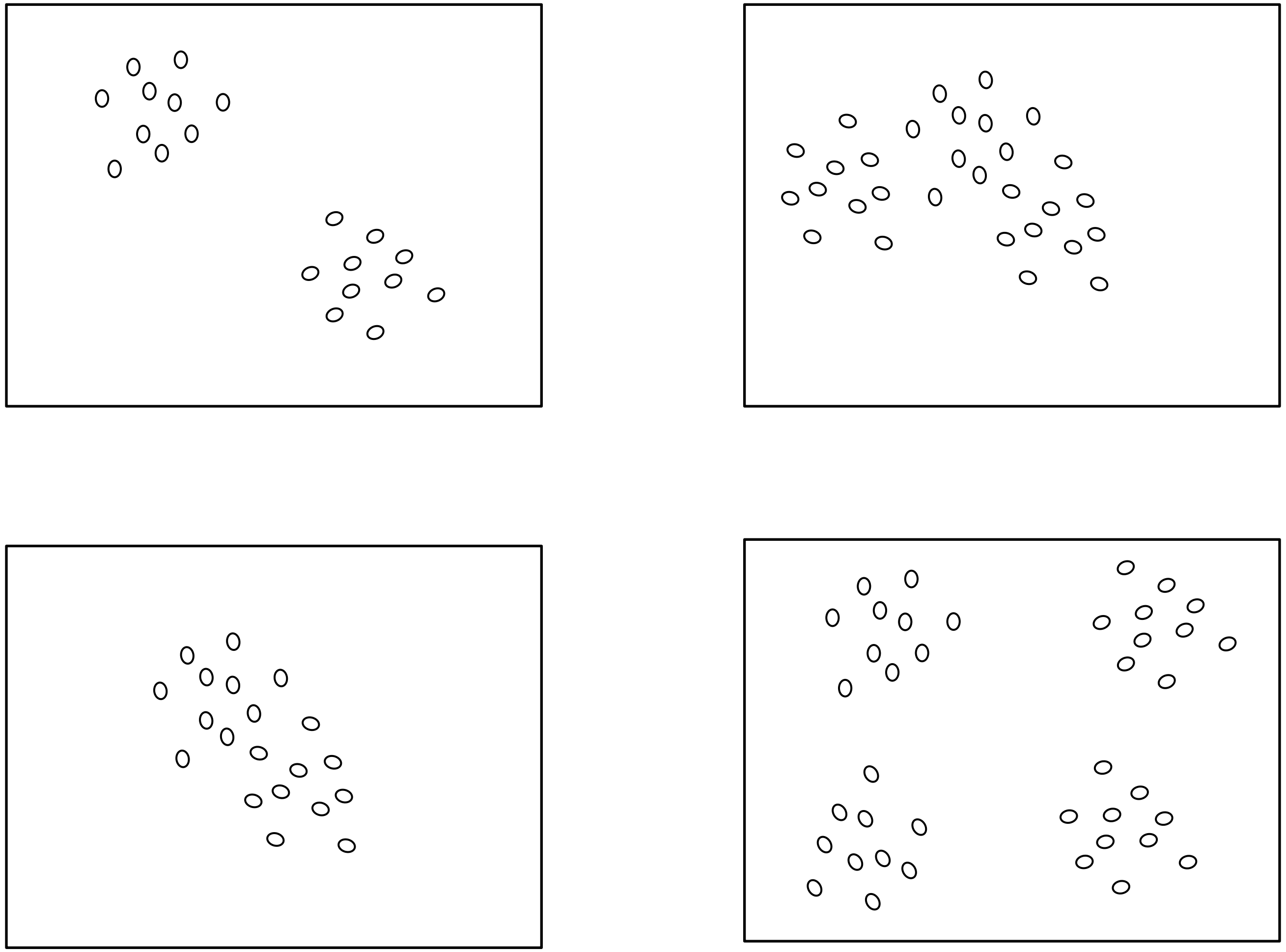
1. **Gathering and ‘Wrangling’ Data**
2. **Exploratory Data**
   - Review Variables
   - Clustering
     - e.g. Kmeans finds K groups with high inter-group, low intra-group variance
   - Dimension Reduction/Factor Analysis
     - e.g. Principal Components find good projections that ‘line up’ with data variance directions
3. **Data Preparation**
   - Selecting, transforming variables
4. **Build and Evaluate Model**

Data Mining also:

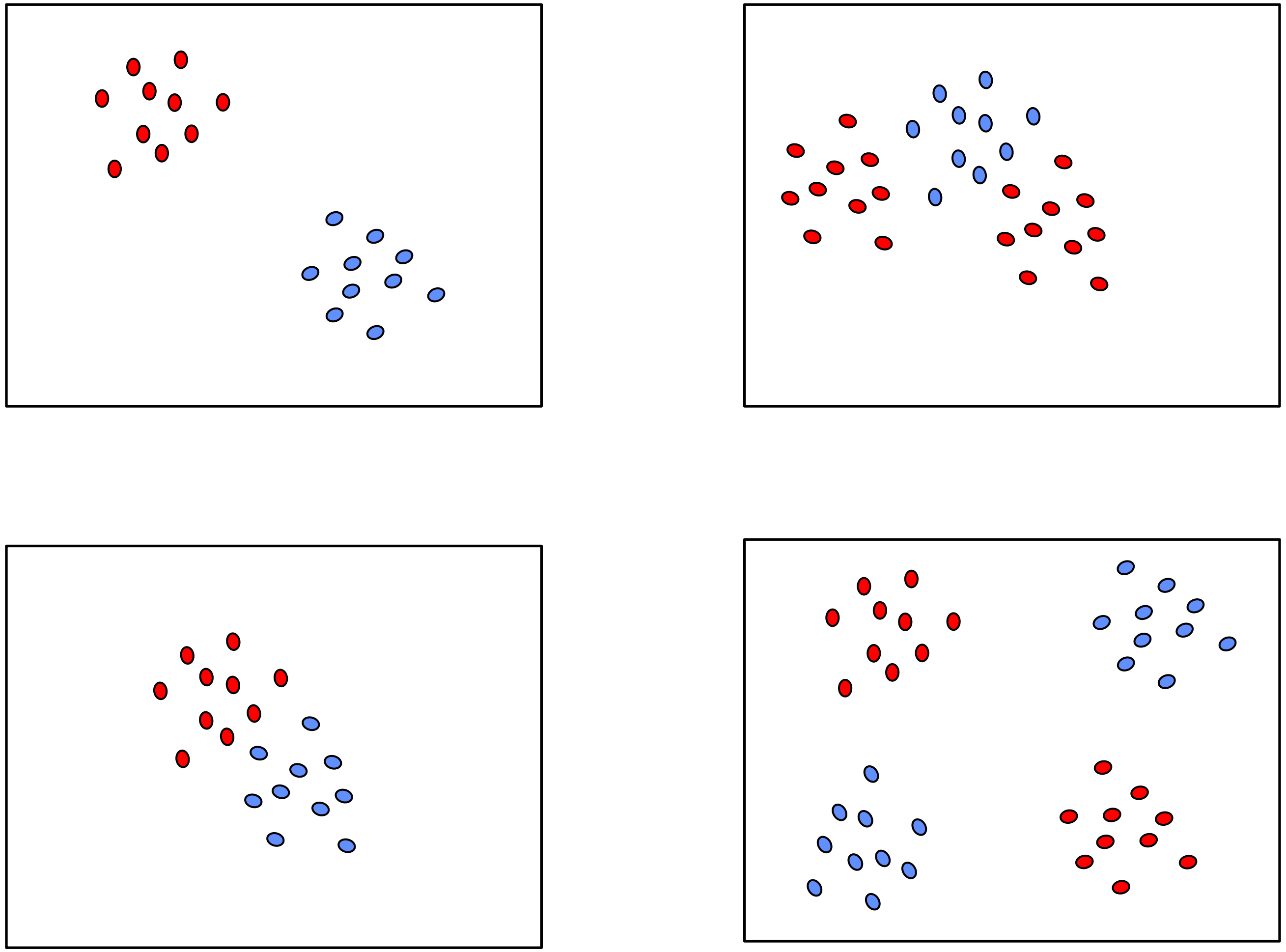
- **Data Mining is often used generally to encompass related analysis, such as:**
  - Text Analytics
    - e.g. word clouds and topic modelling
  - Association Learning
    - e.g. what consumer shopping choices are associated
  - Network Models
    - e.g. which friends have more influence in a social network

A note about clustering

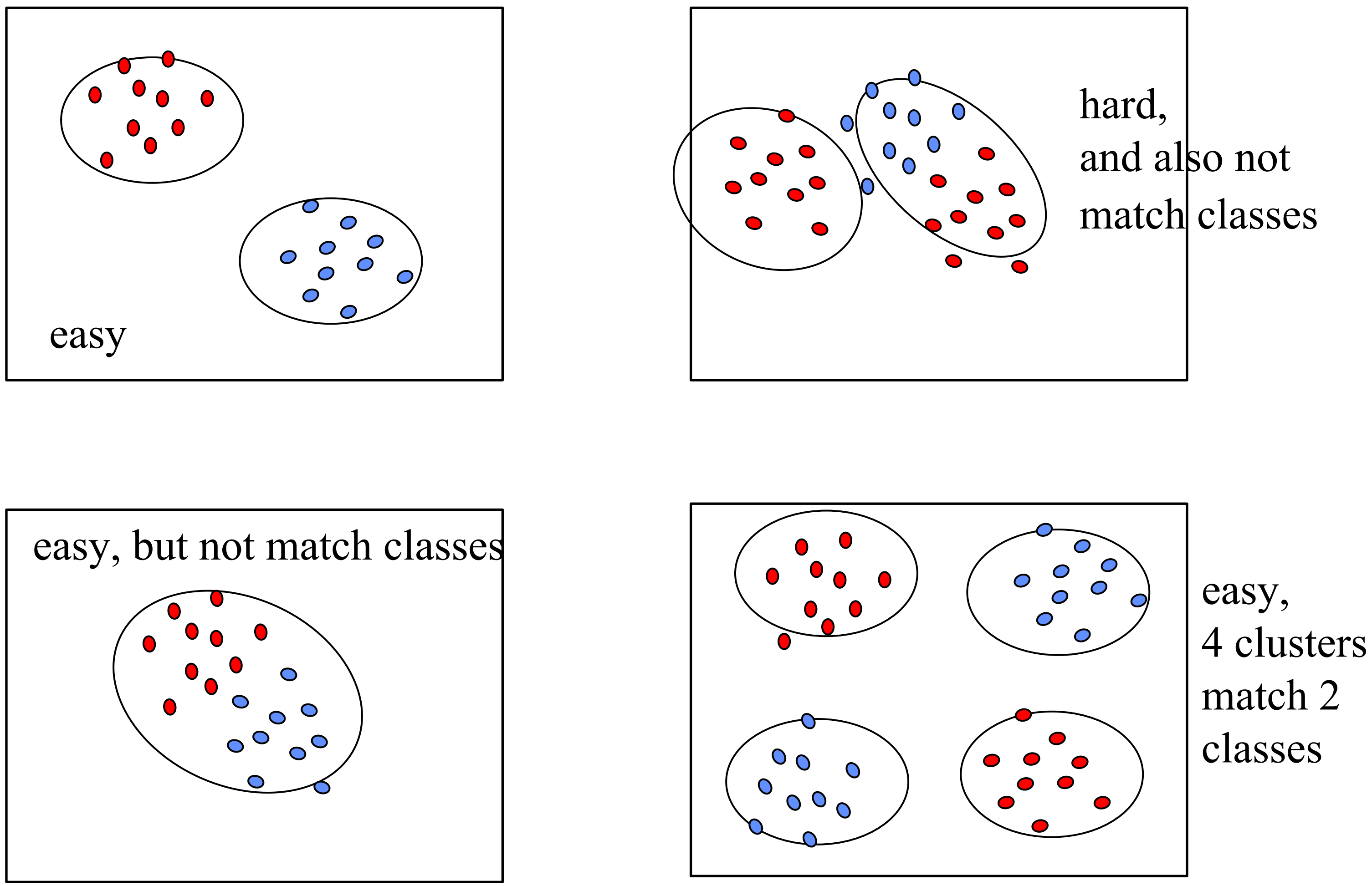
Imagine these 2 dimensional input spaces:
Which of these is easy or hard to cluster?
(e.g. separate into groups)



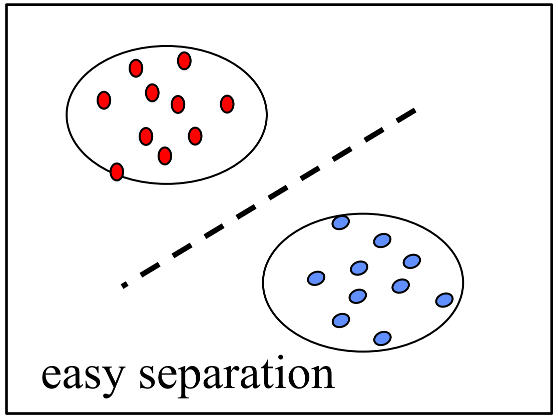
Now imaging there are two classes

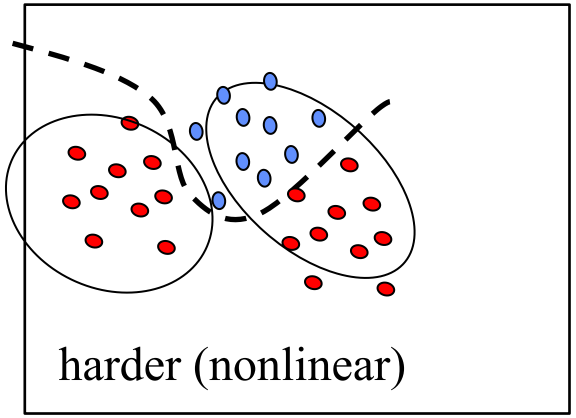


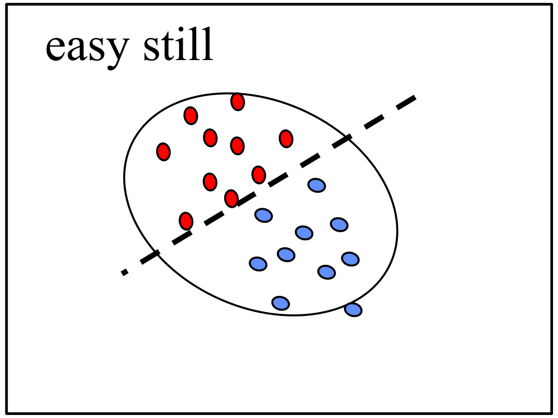
Potential clusters

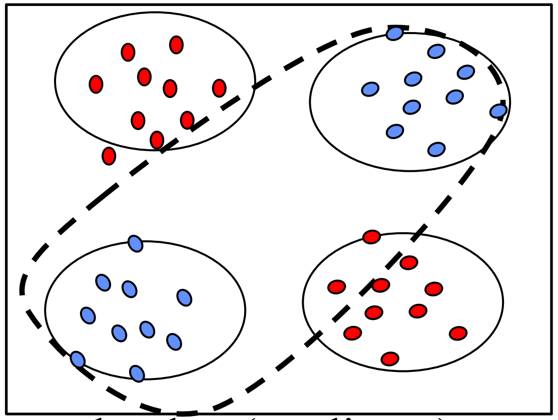


Which are easy or hard to classify? (ie separate red or blue with lines)









harder (nonlinear)

Upshot:
No easy relationship between clusters and classification

Pause

R, Scaling R, Parallel R

- A Glimpse of R
- R strengths
- R and Scaling
- Parallel options for R

The What and Why of R

- **A statistical computing environment**
  - Full set of Statistical/Mathematical functions
  - Programming Language for complete data manipulation
- **Free, Open Source**
- **Extended with user written packages**
- **Widely used in academic and increasingly in industry**

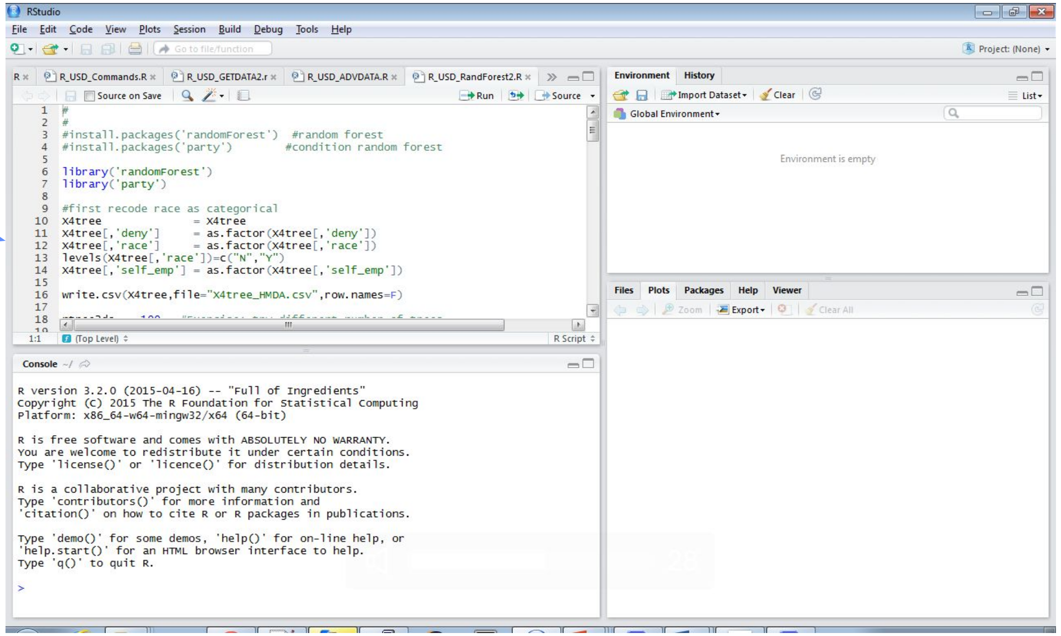
A typical R development workflow

- **R studio: An Integrated development environment for R on your local machine – good for development**

Menu tab

Edit window to Build scripts

R console



Environment Information on variables and command history

Plots, help,

R commands in brief

- A typical R code workflow:

```
#READ DATA (housing mortage cases)
X                =read.csv('hmda_aer.csv',header=T,stringsAsFactors=T)
#SUBSET DATA
indices_2keep =which(X[,'s13'] %in% c(3,4,5)))
X                =X[unique(indices_2keep),]
#CREATE/TRANSFORM VARIABLES
pi_rat           = as.numeric(X[,'s46']/100)          #debt2income ratio
race             = as.numeric(X[,'s13'] %in% c(3,4)) #race category
deny             = as.numeric(X[,'s7']==3)             #dependent variable
#RUN MODEL and SHOW RESULTS
lm_result        =lm(deny~race+pi_rat)              #lm is 'linearmodel'
summary(lm_result)
```

R strengths

- **Sampling/bootstrap methods,**
- **Data Wrangling,**
- **Particular Statistical procedures that you won't find implemented anywhere else, e.g.**
  - Multiple Imputation methods,
  - Instrument Variable (2 stage) Regression

R and missing data

- **Several packages, such as ‘mice’ , ‘amelia’**
- **Produces multiple data sets**
- **Iterates over missing data estimates and linear model estimates**
  - Mice uses Gibbs sampling (slower)
  - Amelia uses Expectation Maximization (faster)
- **Beware of correlation in variables**
  - Matrices not invertible (affects Amelia)

R and missing data

- **Sample R code using Amelia:**
  Data: UN conflict data in pairs of countries
  300K rows ~ 1 hour on Gordon compute node (not run on the user's PC)
  1K-100K entries missing per col for about 20 of 50 cols

time variable *crosssection*

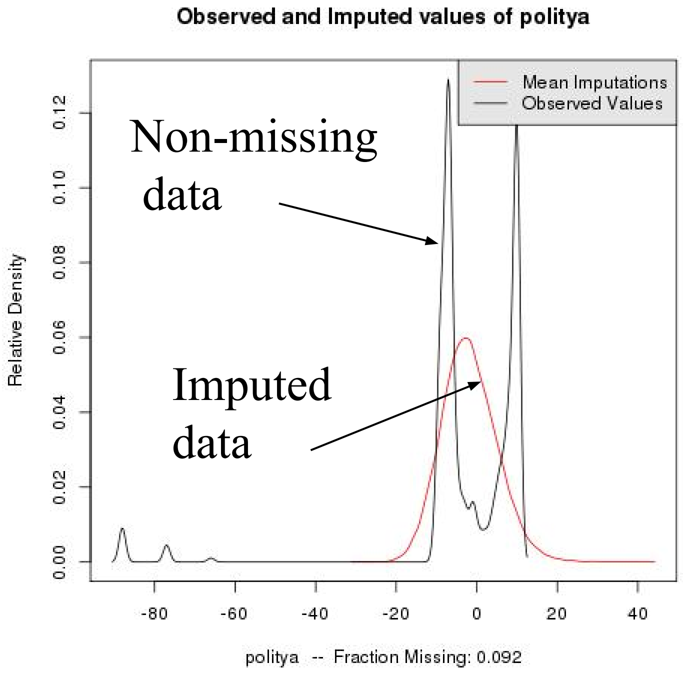
ignore 'id' variables

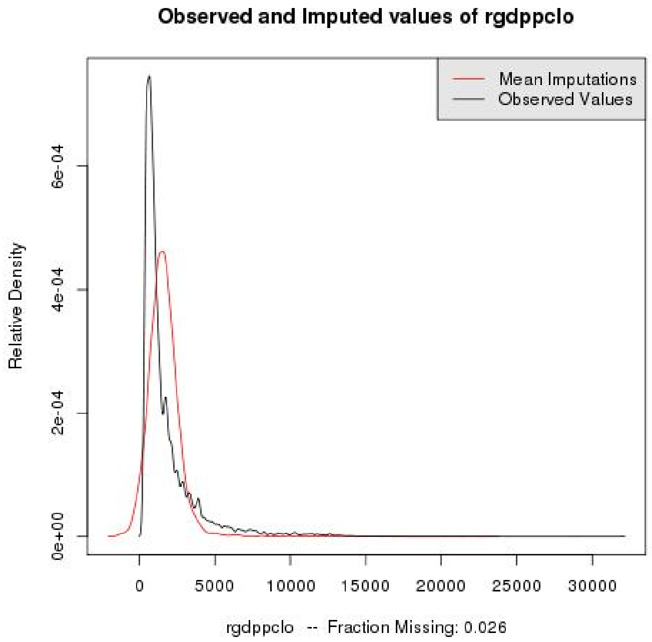
```
# run the imputation
library('amelia')
 a.out <- amelia(data, ts = "year", cs = "dyadid",
                 idvars = c("dyadidyr", "cntryera", "statea", "stateb"),
                 intercs=FALSE, p2s = 2, m=10, parallel = "multicore")
```

interactions *parallel options*

R and missing data

#Perform QA on missing data by comparing density of imputated & original data
compare.density(a.out, var="polityа")
compare.density(a.out,var='rgdpcontg')



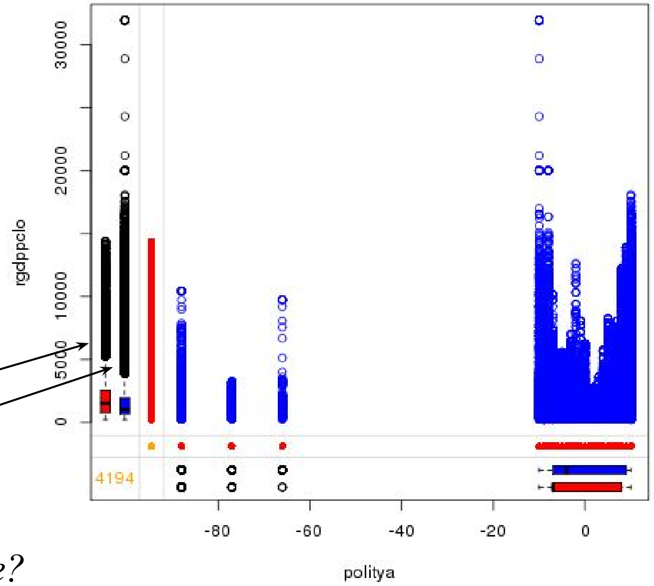


R and missing data

 # Useful library for printing margin plots, to compare histograms for single variables # and histograms conditional on missing/non-missing data

```
library('VIM')
marginplot(gart2use[,c('politya', 'rgdppclo')],
  col=c('blue','red','orange')
```

dist of rgdpcontg when
politya missing
politya present



What would you want to see here?

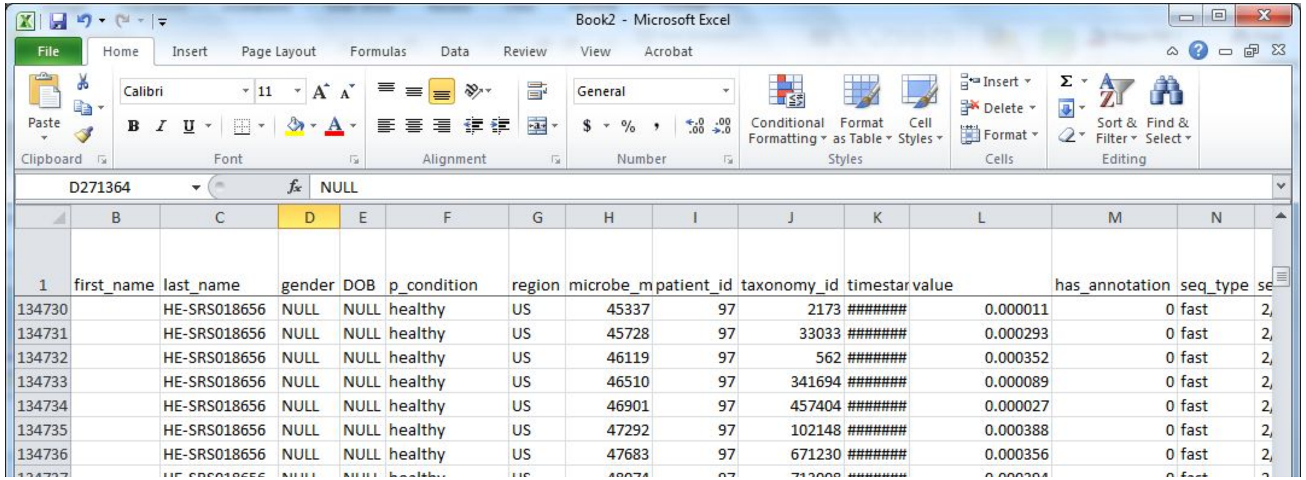
A Data Wrangling example

tables joined

PatientID	Age
HE.1	50
HE.2	25
HE.3	…

Patient ID	Microbe	Value	…
HE.1	Ecoli	0.1	
HE.1	Bacteriia 1	0.01	
HE.1	Bacteria 2	0.001	
…			

Issue:
~3000 rows for each subject (1 row for each measurement).
But needed 1 row per subject.



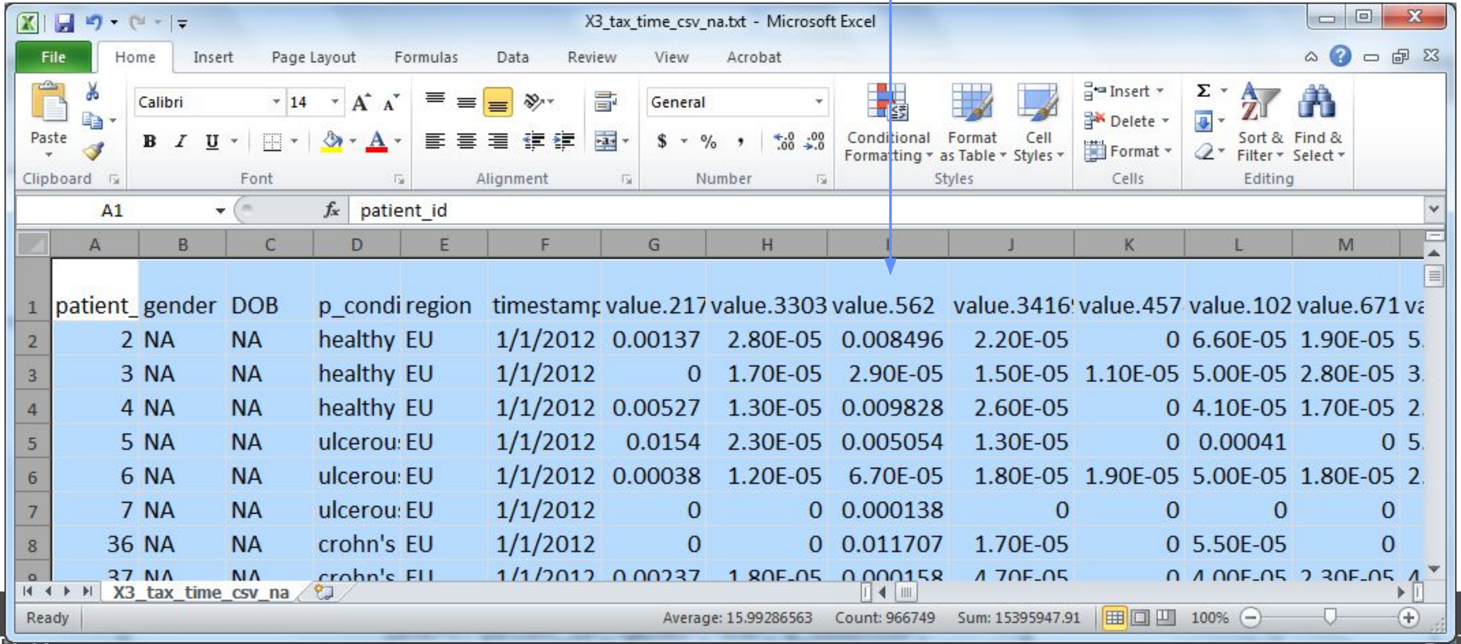
1	first_name	last_name	gender	DOB	p_condition	region	microbe_m	patient_id	taxonomy_id	timestar	value	has_annotation	seq_type	se
134730		HE-SRS018656	NULL	NULL	healthy	US	45337	97	2173	#######	0.000011	0	fast	2,
134731		HE-SRS018656	NULL	NULL	healthy	US	45728	97	33033	#######	0.000293	0	fast	2,
134732		HE-SRS018656	NULL	NULL	healthy	US	46119	97	562	#######	0.000352	0	fast	2,
134733		HE-SRS018656	NULL	NULL	healthy	US	46510	97	341694	#######	0.000089	0	fast	2,
134734		HE-SRS018656	NULL	NULL	healthy	US	46901	97	457404	#######	0.000027	0	fast	2,
134735		HE-SRS018656	NULL	NULL	healthy	US	47292	97	102148	#######	0.000388	0	fast	2,
134736		HE-SRS018656	NULL	NULL	healthy	US	47683	97	671230	#######	0.000356	0	fast	2,

Data Wrangling example

Reshape command to get a ‘wide’ format with repeated measurements in separate columns of the same record

```
Xa        =read.csv('patmeasjoinall_0219.csv',header=T,sep='\t')
Xatest   =reshape(Xa, direction = "wide",
                    idvar=c("patient_id","gender","DOB","p_condition","region"),
                    v.names="value", timevar="taxonomy_id", sep=".")
write.csv(Xatest,file="Xa_reshaped");
```

Row values are now columns



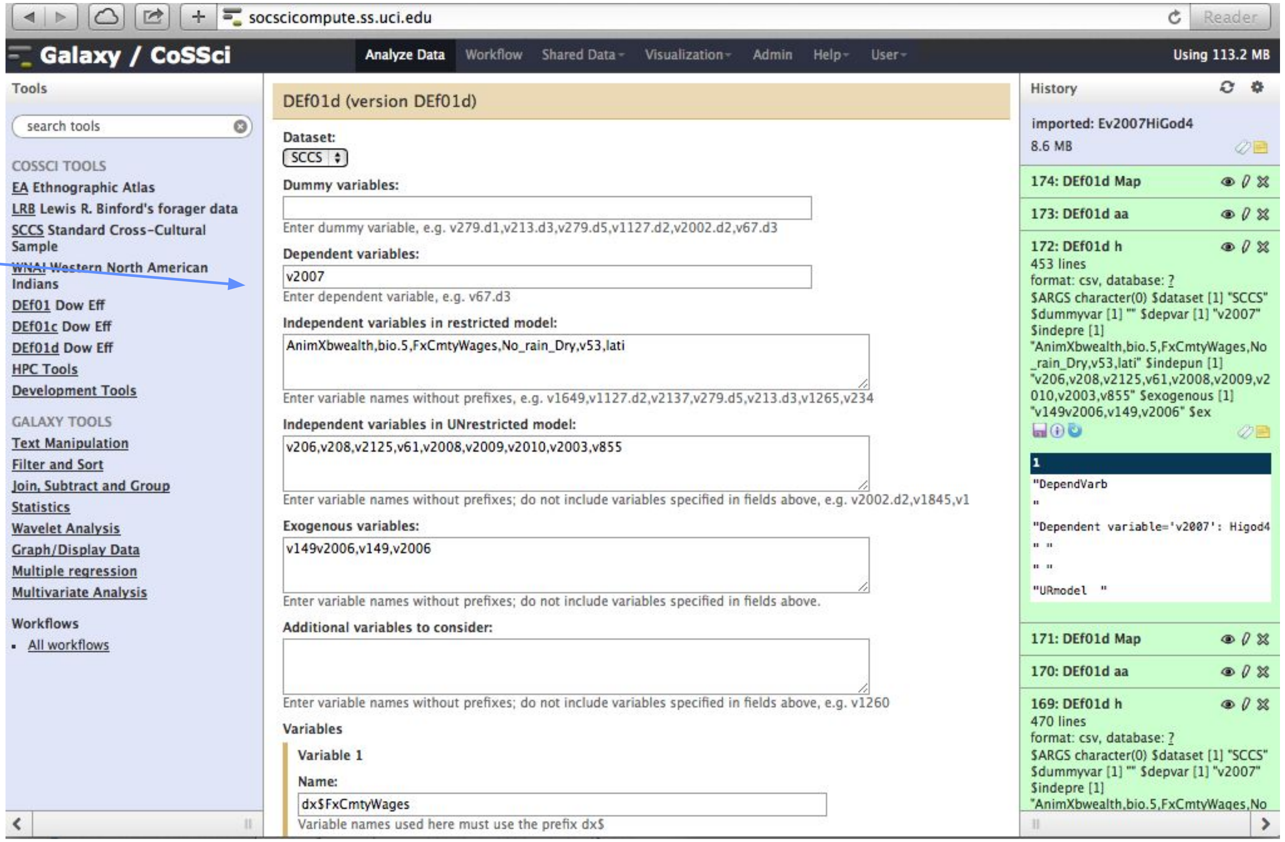
patient_	gender	DOB	p_condi	region	timestamp	value.217	value.3303	value.562	value.3416	value.457	value.102	value.671	va
2	NA	NA	healthy	EU	1/1/2012	0.00137	2.80E-05	0.008496	2.20E-05	0	6.60E-05	1.90E-05	5.
3	NA	NA	healthy	EU	1/1/2012	0	1.70E-05	2.90E-05	1.50E-05	1.10E-05	5.00E-05	2.80E-05	3.
4	NA	NA	healthy	EU	1/1/2012	0.00527	1.30E-05	0.009828	2.60E-05	0	4.10E-05	1.70E-05	2.
5	NA	NA	ulcerou	EU	1/1/2012	0.0154	2.30E-05	0.005054	1.30E-05	0	0.00041	0	5.
6	NA	NA	ulcerou	EU	1/1/2012	0.00038	1.20E-05	6.70E-05	1.80E-05	1.90E-05	5.00E-05	1.80E-05	2.
7	NA	NA	ulcerou	EU	1/1/2012	0	0	0.000138	0	0	0	0	
36	NA	NA	crohn's	EU	1/1/2012	0	0	0.011707	1.70E-05	0	5.50E-05	0	
37	NA	NA	crohn's	EU	1/1/2012	0.00237	1.80E-05	0.000158	4.70E-05	0	4.00E-05	2.30E-05	4.

P.R. SDSC UCSD

Complex Social Science Gateway – a tool for cross-cultural analysis in R

Select dataset,
Select varialbes,
Submit analysis

http://socscicompute.ss.uci.edu/

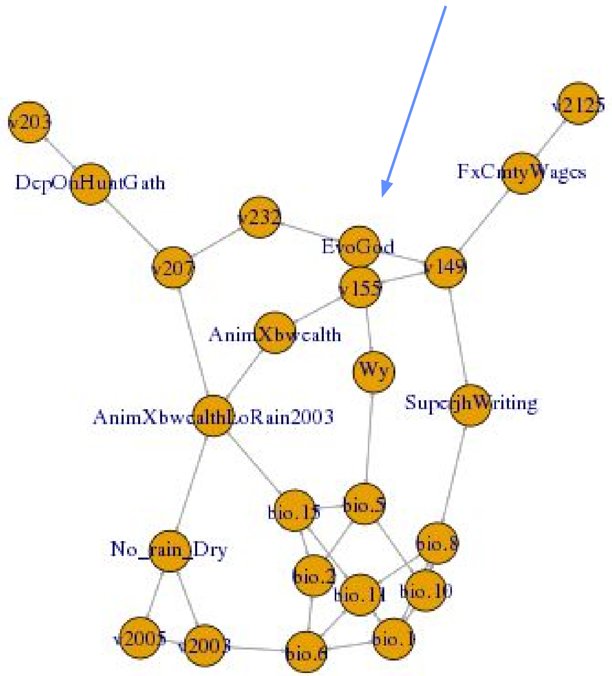


R Analysis options

- **Two-stage least squares to handle spatially correlated errors (OLS, logit, multinomial logit)**

- **Bootstrap sampling of Bayesian network (package bnlearn) to confirm OLS effects, or suggest other moderating/mediating effects**

Depend. var



Scaling, practically

- **Scaling (with or without more data):**
  - more complex analysis (ie optimizations)
  - more sampling (ie more trees in Random Forest)

- **Sometimes easy to parallelize (like with sampling),**
- **Sometimes too much communication between parts (matrix inversion)**

Scaling In a nutshell

- R takes advantage of math libraries for vector operations
- R packages provide multicore, multinode (snow), or map/reduce (RHadoop) options
- However, model implementations not necessarily built to use parallel backends
  - Some models more amenable to parallel versions

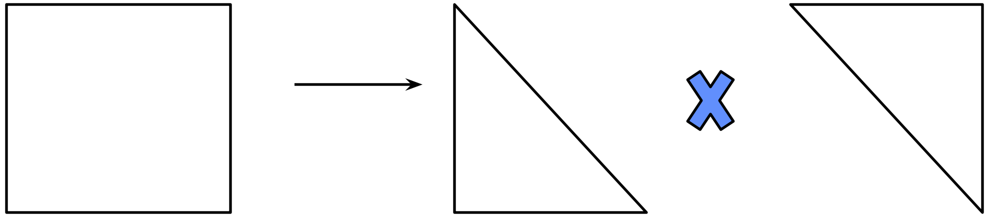
Consider Regression Computations

- **Linear Model:** $Y = X * B$

  where Y=outcomes , X=data matrix

- **Algebraically, we could:**
  - take “inverse” $of\ X * Y = B$ (time consuming)
  - Or take gradient descent (for likelihoods and generalized models)

- **Or, better:**
  - QR decomposition of *X* into triangular matrices (easier to solve but more memory)



Consider Regression models in R

- **Related Models and Functions :**

  lm() #Linear Model

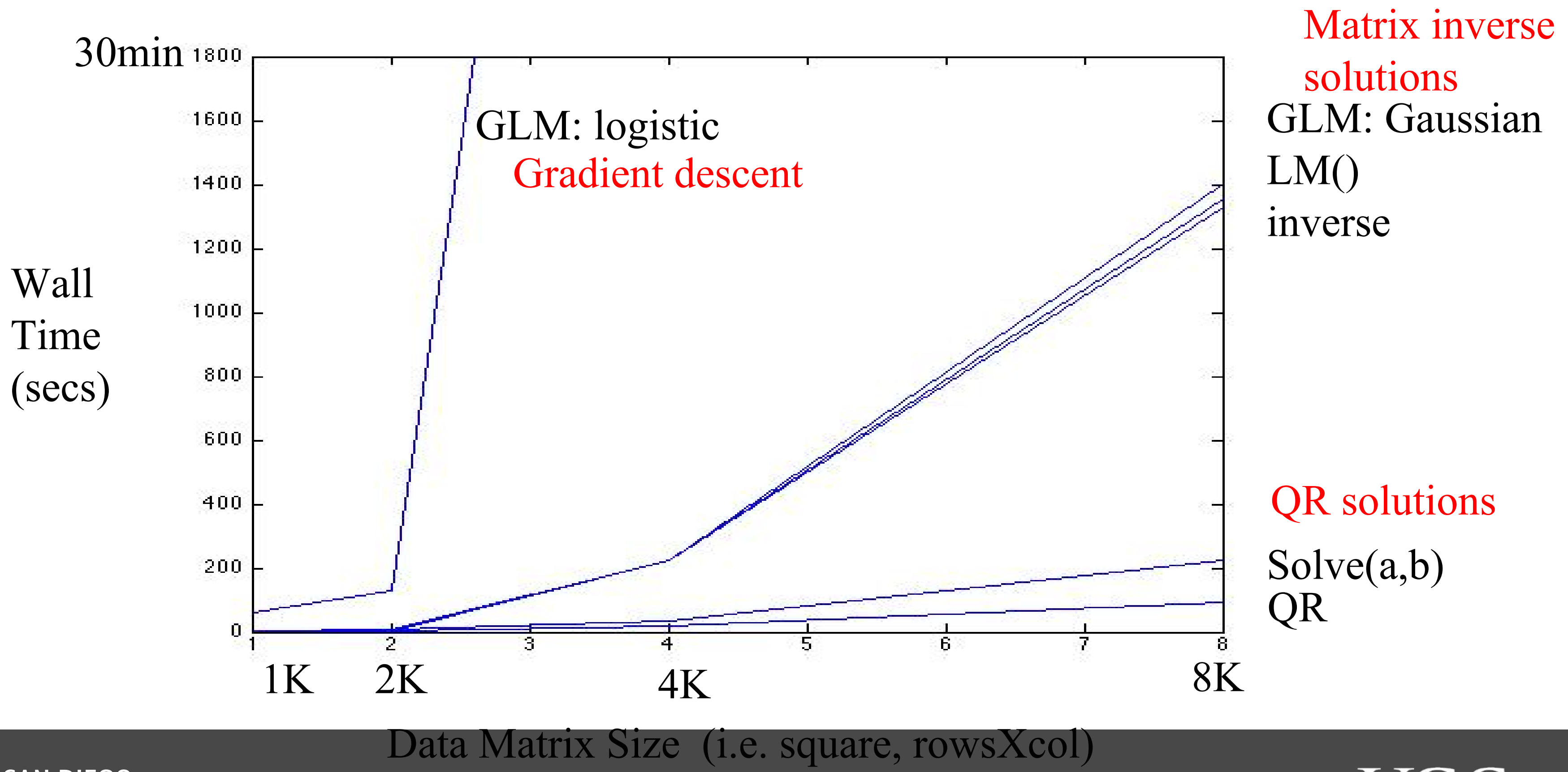
  glm() #Generalized Linear Model (logistic regression, etc)

  aov() #Analysis of Variance (returns ANOVA table of F-scores)

All these work on system of equations

Solving Linear Systems
Performance with R, 1 compute node

R:
glm(Y~X,family=gaussian) #gaussn regrssn (like lm)
glm(Y~X,family=binomial) # logistic regrssn (Y=0 or 1)



R multicore

- Intel Math Kernel Libraries provides fast operations for vector operations

- Uses threads across cpu cores to pass data & commands

R multicore

- **Run loop iterations on separate cores**

```
install.packages(doMC)
library(doMC)
registerDoMC(cores=24)
getDoParWorkers()
```

allocate workers

%dopar% puts loops across cores,
(loops are independent)
%do% runs it serially

```
results = foreach(i=1:24,.combine=rbind) %dopar%
  {   … your code here

      return(  a variable or object )
})
```

returned items 'combined' into list by default

specify to combine results into array with row bind

R multinode: parallel backend

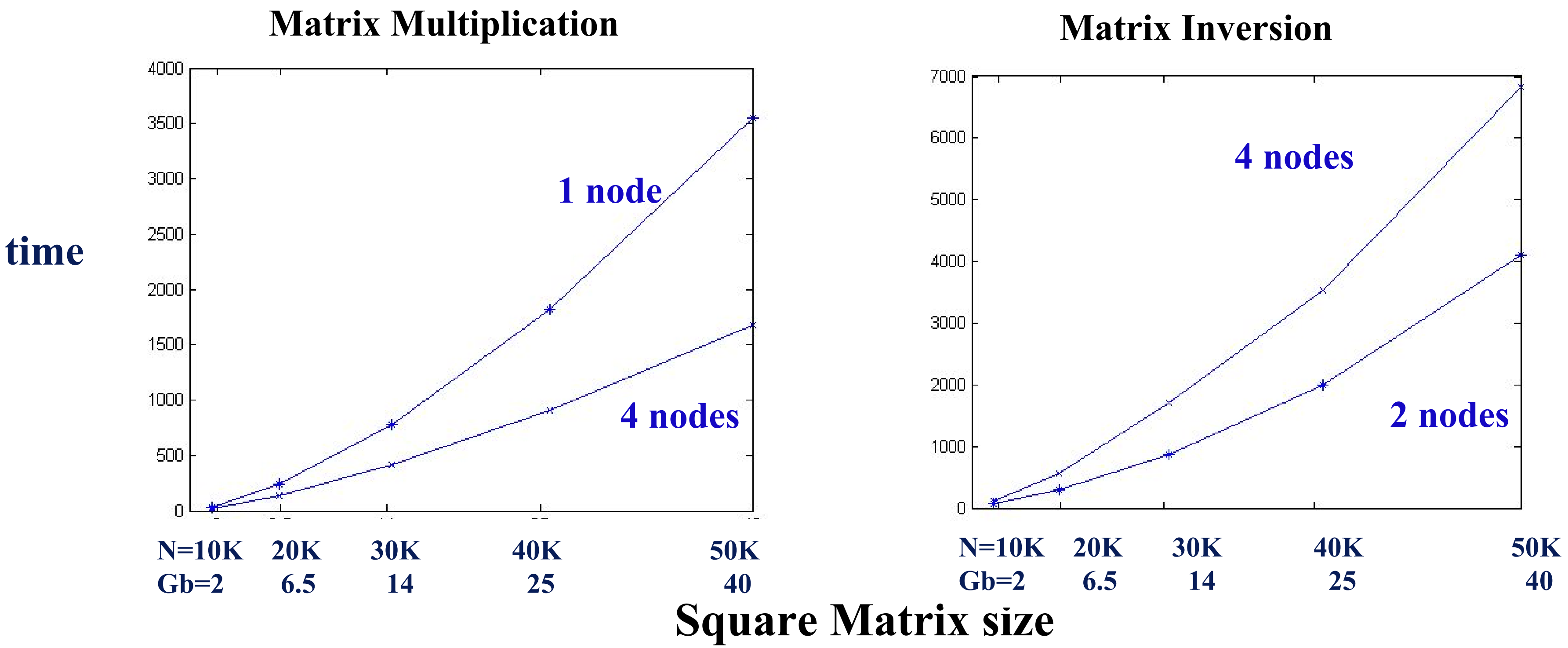
- **Run loop iterations on separate nodes**

```
install.packages('doSNOW')
library('doSNOW')
…
cl <- makeCluster( mpi.universe.size()-1, type='MPI' )
clusterExport(cl,c('data'))
registerDoSNOW(cl)

results = foreach(i=1:47,.combine=rbind) %dopar%
  {   … your code here



      return(  a variable or object )
})
stopCluster(cl)
mpi.exit()
```

allocate cluster as parallel backend

%dopar% puts loops across cores and nodes

Multiple Compute Nodes not always help
(tested on Gordon)



multinodes: more nodes is less time for multiplication,

less nodes is better for inversion

Another Parallel option:

- Serially packing R jobs onto cores

1. batch script starts a job and calls MPI run utility
2. MPI utility executes a Perl script on each core
3. Perl script executes R with argument=cpu-id
4. R uses cpu-id to process some particular input

Serial Packing with large Random Forest (ensemble of decision trees) job

- Option 1: Run separate trees on separate cores

%dopar% puts loops across cores,
(loops are independent)
%do% runs it serially

allocate workers

```
install.packages(doMC)
registerDoMC(cores=15)
getDoParWorkers()
library("randomForest");
results = foreach(i=1:15,.combine=rbind) %dopar%
  {RF1 <-randomForest(formula,data=X,na.action=na.omit,
       importance=TRUE,
       ntree=100000,
       do.trace=1,
       nodesize=1)
classRF1$importance
})
```

returned items
'combined' into list

Sampling on large data could be huge

Serial Packing with large Random Forest job

- **Option 2: split sampling to make it embarrasingly parallel**
  ie run R script on separate cores and average results

- **And, for very large number of parameters, run each tree on subset of variables**
  ie take samples of columns, run lots of trees
  Can speed up processing without losing interesting variable combinations

Serial Packing with large Random Forest job

- *A GWAS (genome wide analysis) study – RandomForest Sampling in stages:*
  1. Take 1000 samples out of ~80K variables (SNPs)
  2. Run 50 trees on each sample => 50K total trees
  3. Run 1 instance of R on each core => on 4 compute nodes (64 cores) < 16hours (on Gordon)

- Runs better than using R multicore with foreach b/c less total memory across nodes

Installing your own R Packages

- **In R, install.packages('package-name') to get specific functionality you want**
  - See https://cran.r-project.org/
- **But on Comet**
  - Need to specify URL when prompted and say 'Y' to personal library;
  - If compiling is required you might get an error, call user support

Other R packages:

- Rspark - R interface to Spark
- pdbR - higher level over R-MPI, distributed matrix support and other
- Rgputools – GPU support
- R openMP , better data mgt than dopar, parallel (mclapply)

Spark ML for bigger data:

- **Spark MLlib –**
  - Many standard Machine Learning models that are easiest to parallelize
    - Matrix Factorization
    - Naïve Bayes
    - Linear/NonLinear Regression Models with gradient descent optimization
    - Kmeans
  - Some support for large matrix operations

Other Examples on Comet

Image Analysis of Rural Photography

Computer vision and text analytics with 171,021 depression era photographs from Library of Congress.

Feature extraction to database to interface to visual data mining

Title: "Barber and shop"
Location: Omaha, Nebraska
Photographer: John Vachon.
Date Created: 1938.

Image Gray Scale:

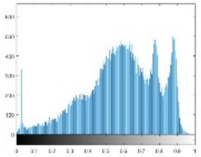
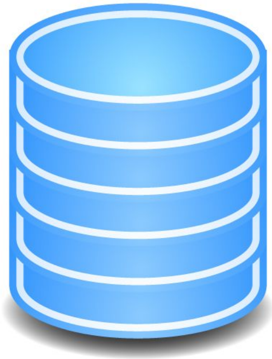


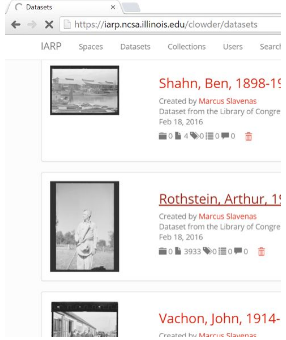
Image Content

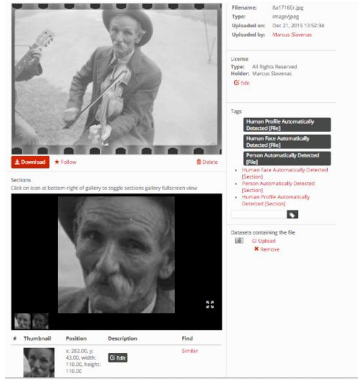
OCR + RandomForest :
BARBER SHOP;
ENGLISH
FACE DETECTION: 1 face

Metadata

SEMANTICS:
<shop::business;structure;entity>
<barber::worker;person>
GEO : 41.2°N, 95.9°W
etc…

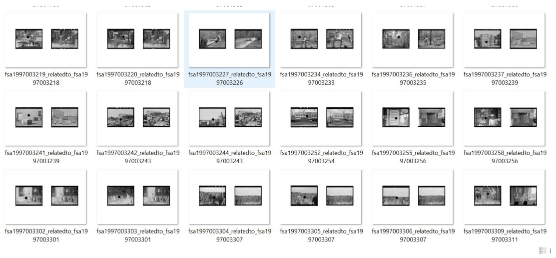


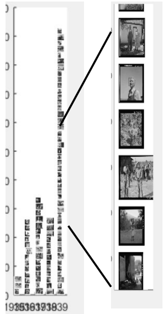


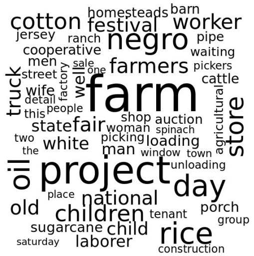


SQL with visualizations

Data Mining on American life, visual rhetoric, and aesthetics.

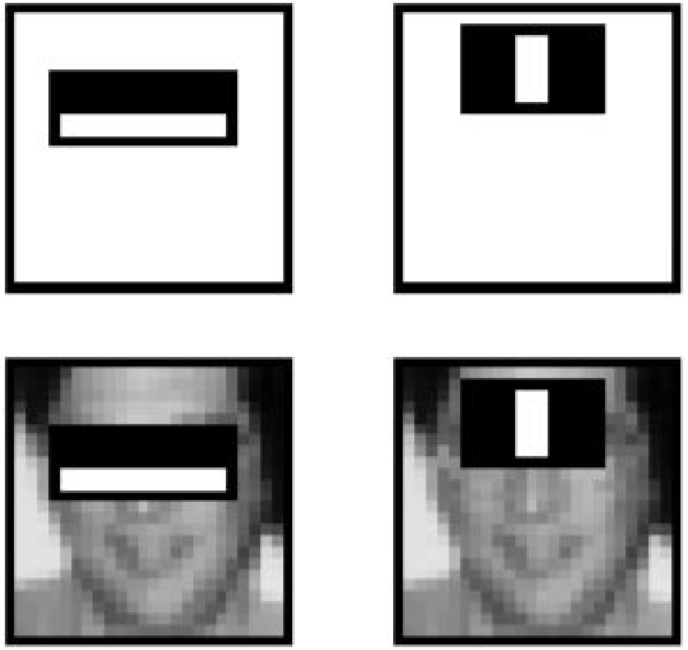






Face Detection detail:

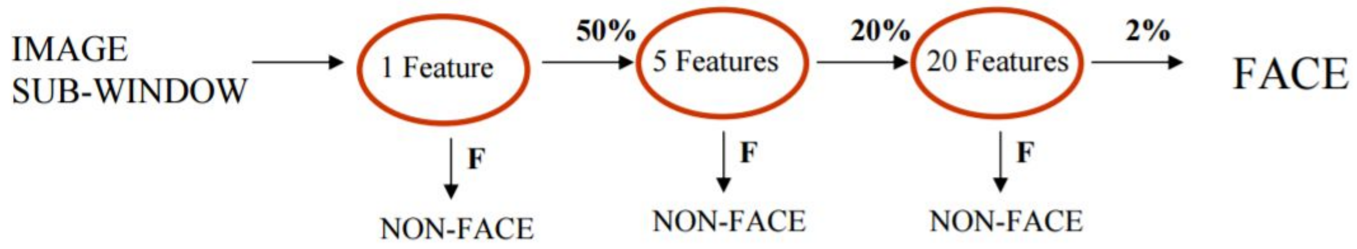
- **Using Python and OpenCV package**
- **Features are combinations of on/off pixels**
  - a dark patch above a light patch, as with eyebrows and eyes
- **Uses a large number (100's) of these features.**



Combining features efficiently:

- Sweep of different scales
- Sweep over image subwindows.
- Feature combined in a chain so that if the most important features are not found above some threshold at some point in the chain, the subwindow is discarded,

Cascaded Classifier



Profiles, or obscured faces do not work well.

Frontal faces work well. False positives are often small face-like patterns.

Overall performance is about 95% accuracy for these digitized photos. Performance is often higher for contemporary images.
171K 1kx1k images take about 12 hours on 1 Comet node.

Text Analytics processing of metadata:

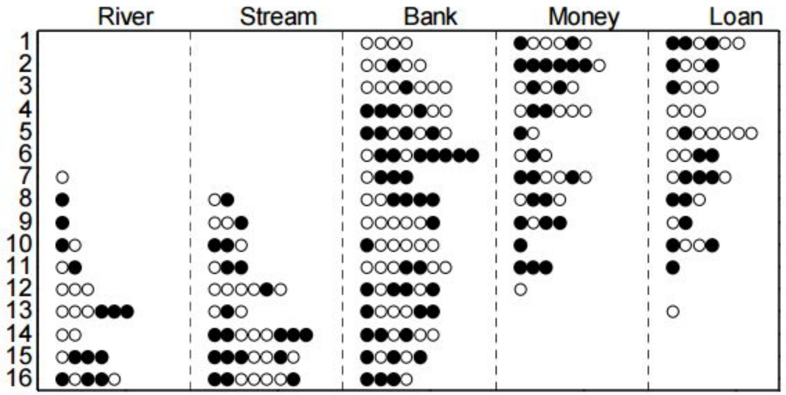
- **Parse, tag speech, search ontology**
  - using Stanford NLP tools, Wordnet ontology, Python NLP toolkit, 101K titles ~ 96 hours on 1 comet node
- **Several words identify ‘person’**
- **SQL: give me all pictures by Lange with possible ‘person’ and num_faces > 0**

Title:

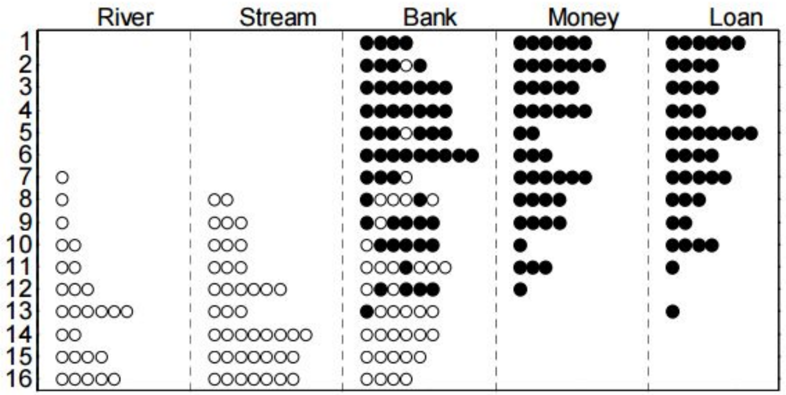
”Destitute pea pickers in California. Mother of seven children.” By D. Lange, 1936, California, [metadata]

Topic Modelling with Latent Dirichlet Allocation

- **Example, 15 documents, 5 words:**
  - Each circle is 1 word occurrence
  - 2 topics
  - Start with random assignments (ie randomly filled/empty circles)



- **After learning, topics are well formed**



LDA optimization

- **Start with initial guess of topic=*t*, and parameters**
- **Until convergence do this:**

  Compute the expected frequency of word=*w for each t*

  Compute the parameters that maximize likelihood L of *t given w*
- **Result is list of topics and word probabilities:**

  P(word|topic)

  P(topic|document)

Topic Modelling with LDA on Comet

- **R LDA package: wraps C programs for Gibbs sampling or EM**

  Slow for > 5000 journal articles,

  Easy to use and interactively explore

- **Mallet: Gibbs sampling, option for multicore, java code**

  Slow for > 50000 articles, easy to use from Unix command line

- **Spark LDA: EM**

  Fastest, big datasets OK, harder to set up, but EM not as robust as Gibbs sampling

- **Asymptotic Distributed LDA : MPI based**

  Faster and Robust, big datasets OK, but difficult to use

- **Example Ouput:**

Sample topic plot (tree map)

THE END