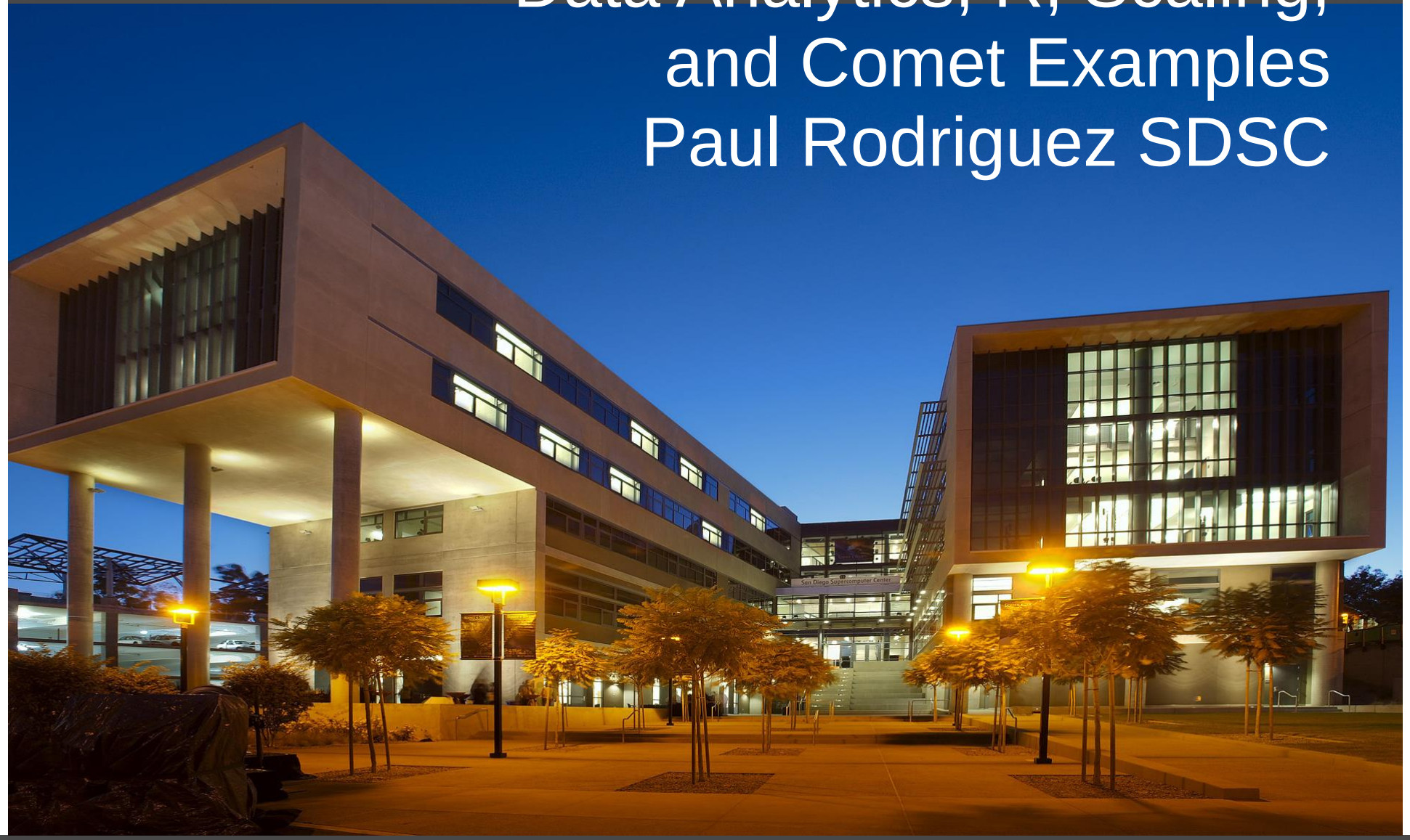
Data Analytics, R, Scaling, and Comet Examples

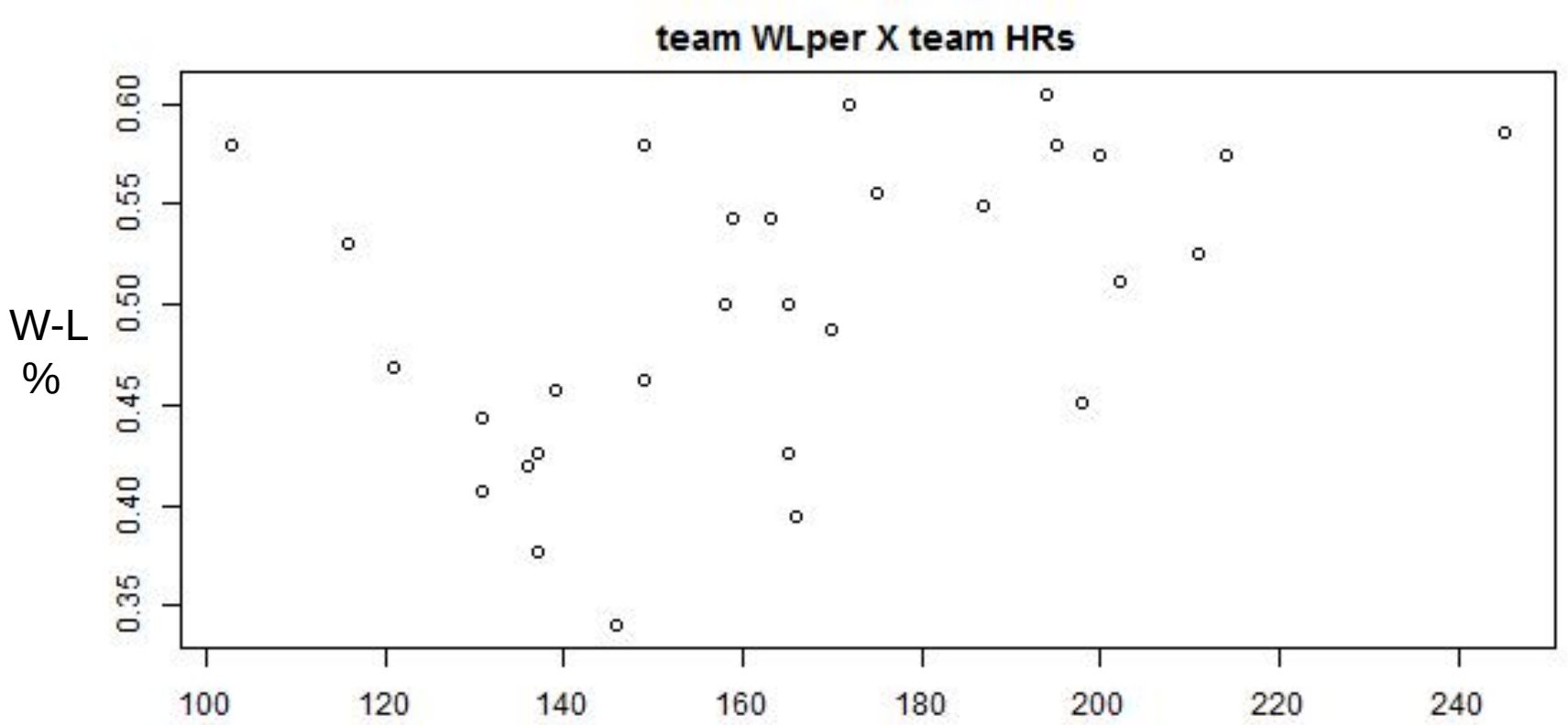
Paul Rodriguez SDSC



Outline

I. Data Analytics
   - Machine Learning in a Nutshell
   - Data Mining also
II. R, Scaling in R, Parallel R
III. Analytics Cases in Comet

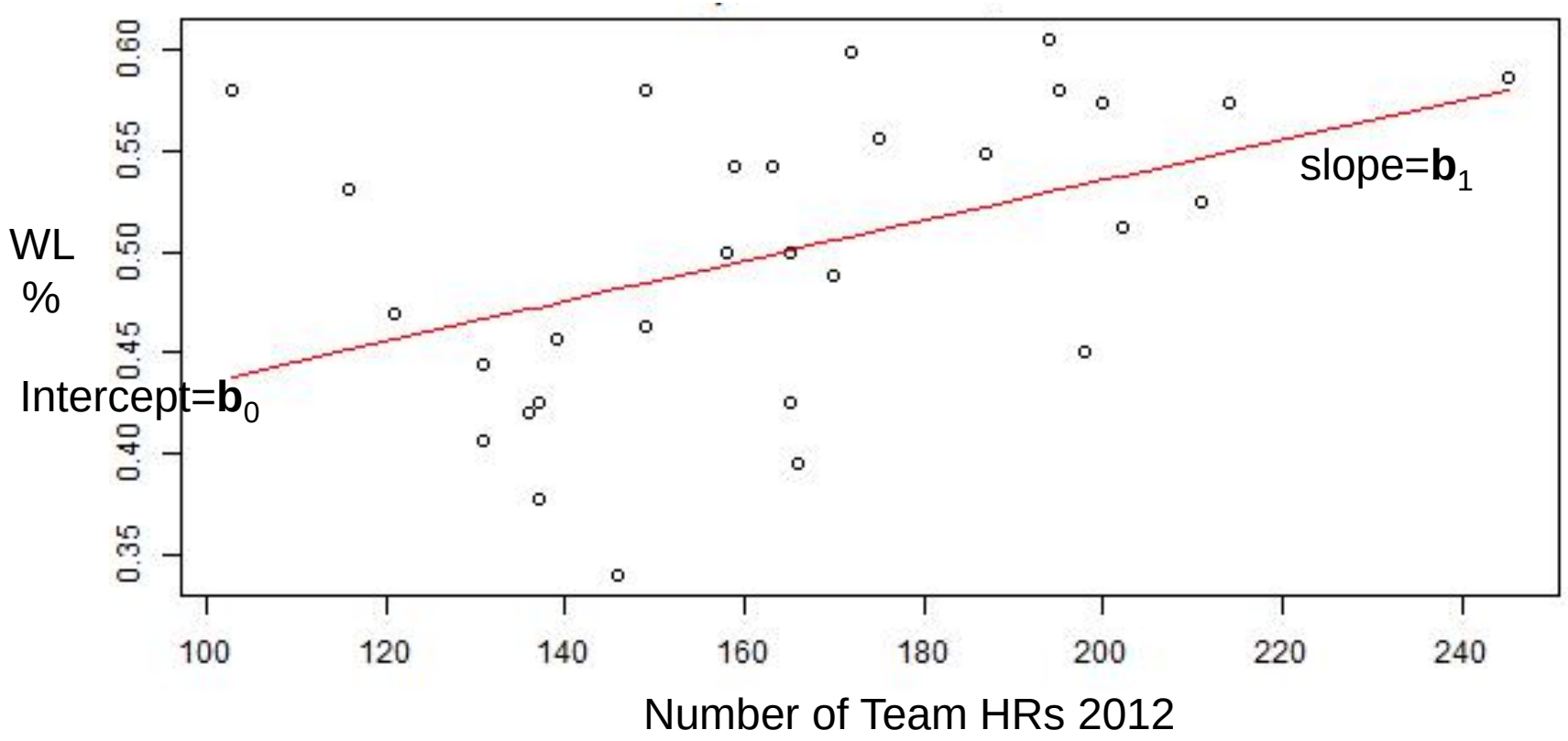
A data example:Home Runs and W-L



Recall Linear Regression is Fitting a Line

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

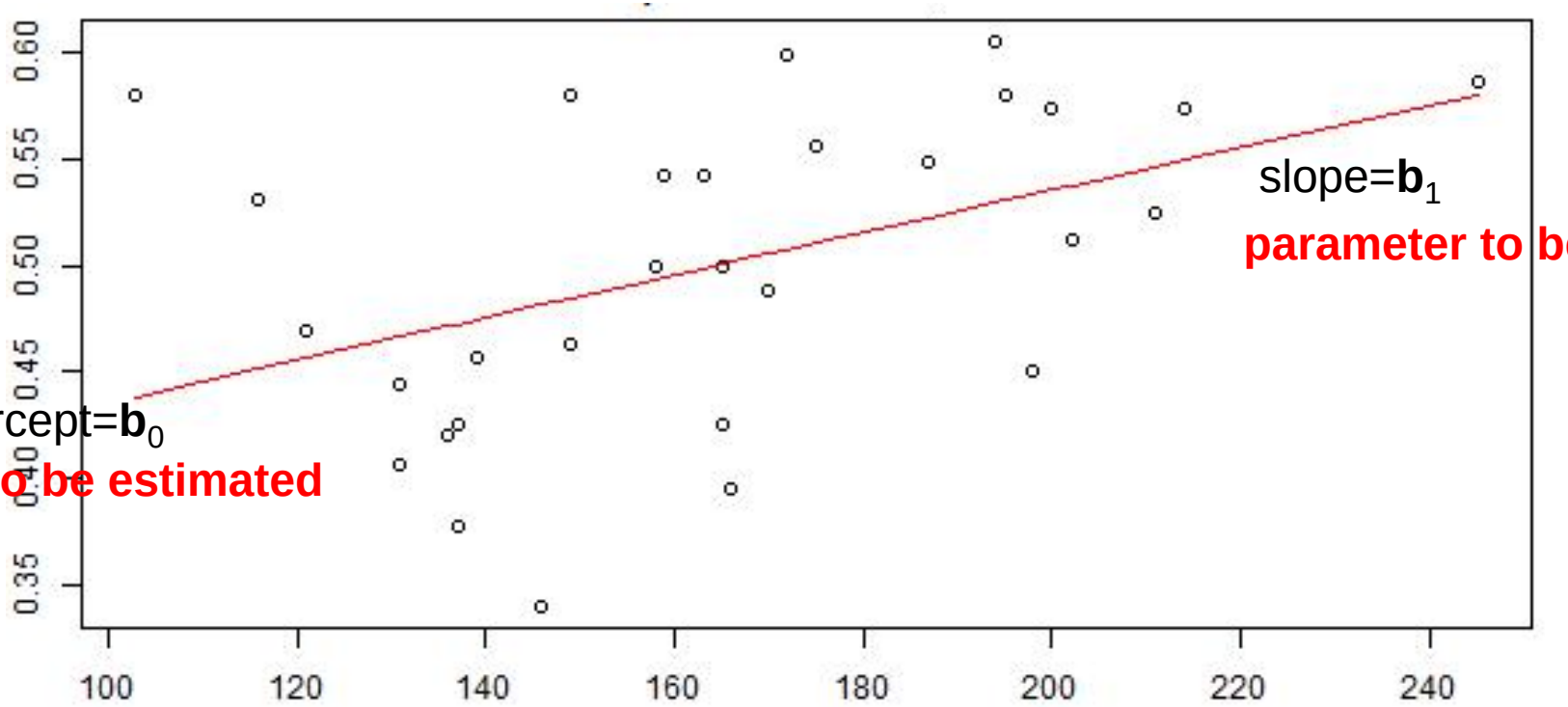
(Won-loss percent as function of team home runs)



Recall Linear Regression is Fitting a Line

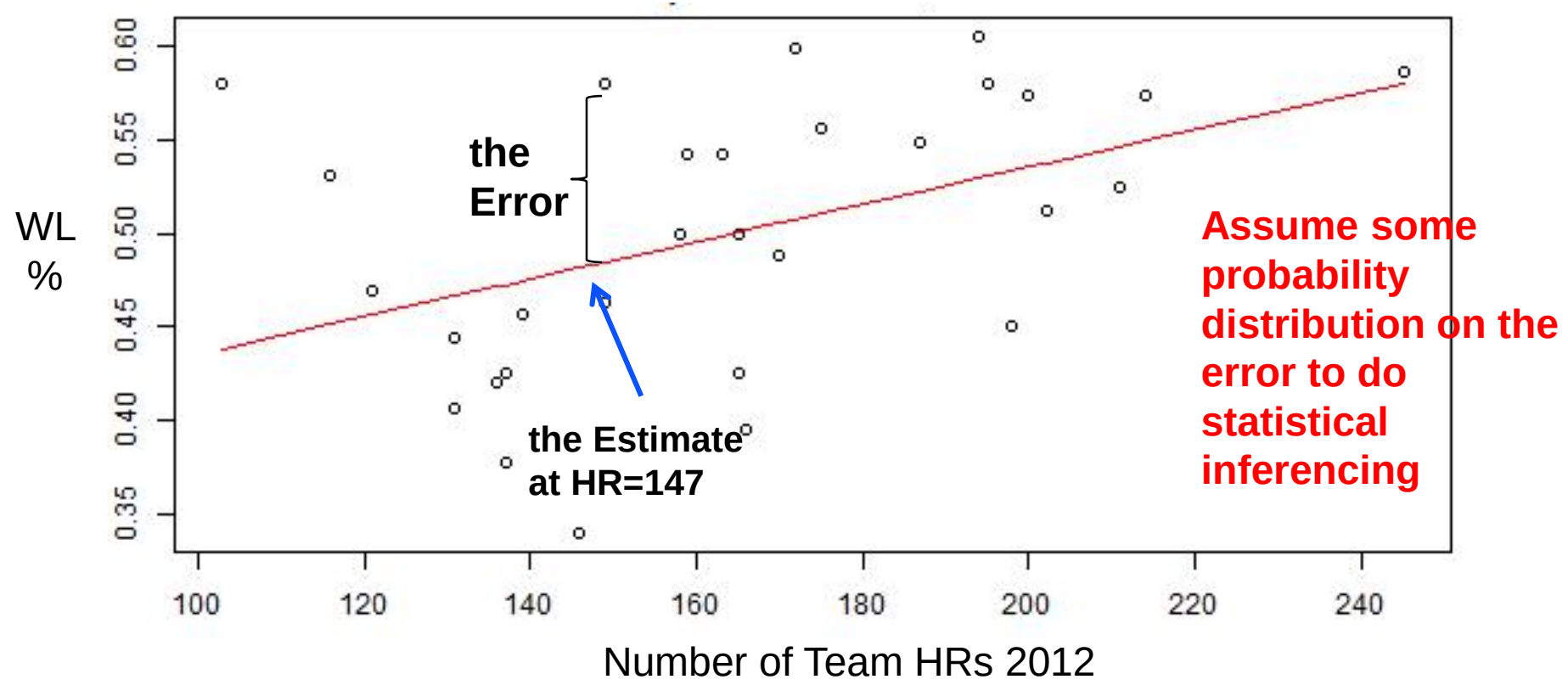
the Model: $y_i = f(x, b) = bo * 1 + b_1 * xi$

(Won-loss percent as function of team home runs)



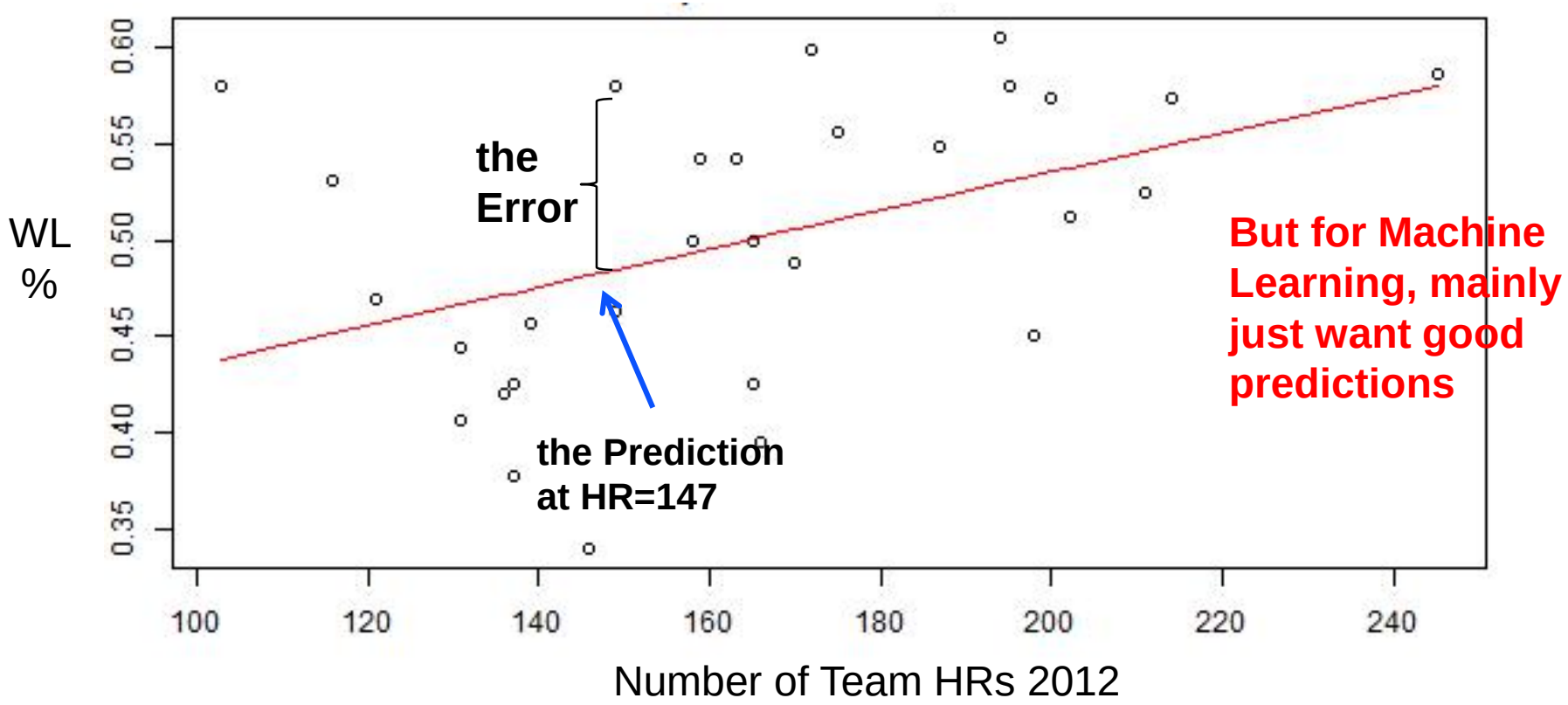
Recall Linear Regression is Fitting a Line

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



Recall Linear Regression is Fitting a Line

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

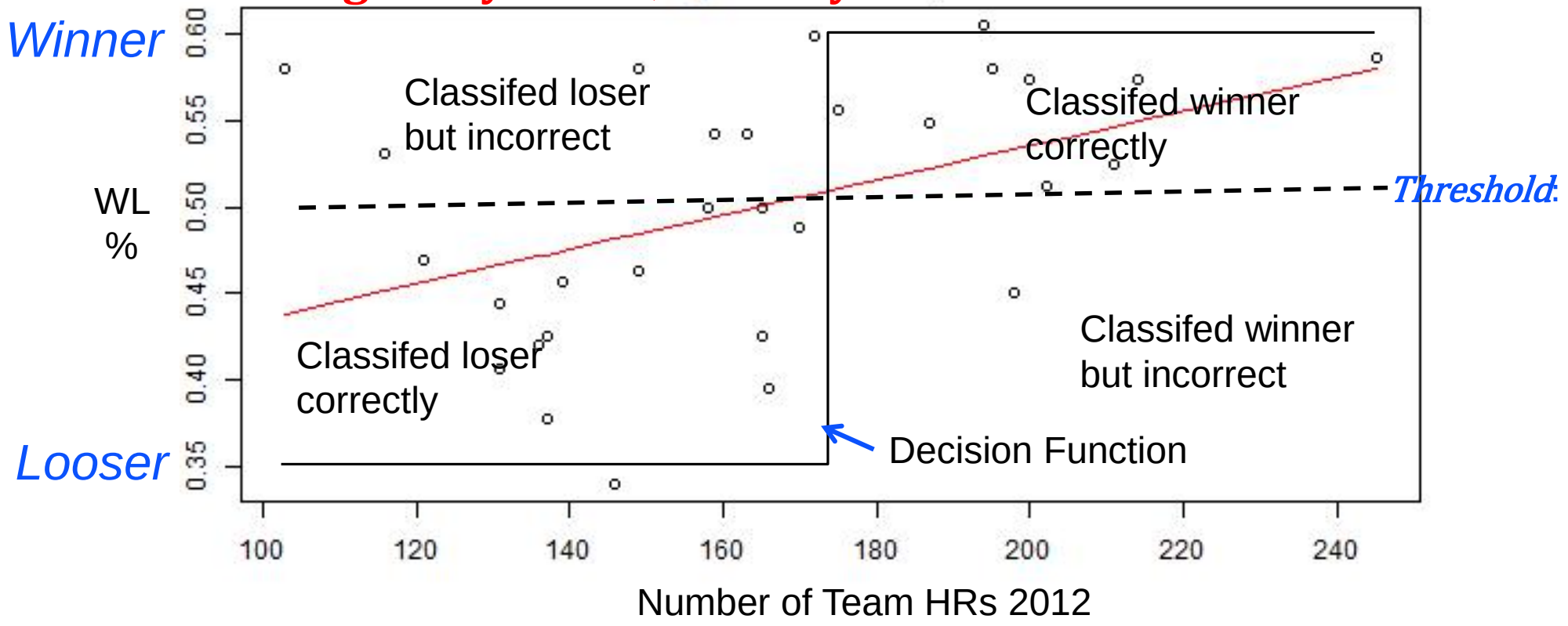


Linear Regression with a Decision Threshold

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

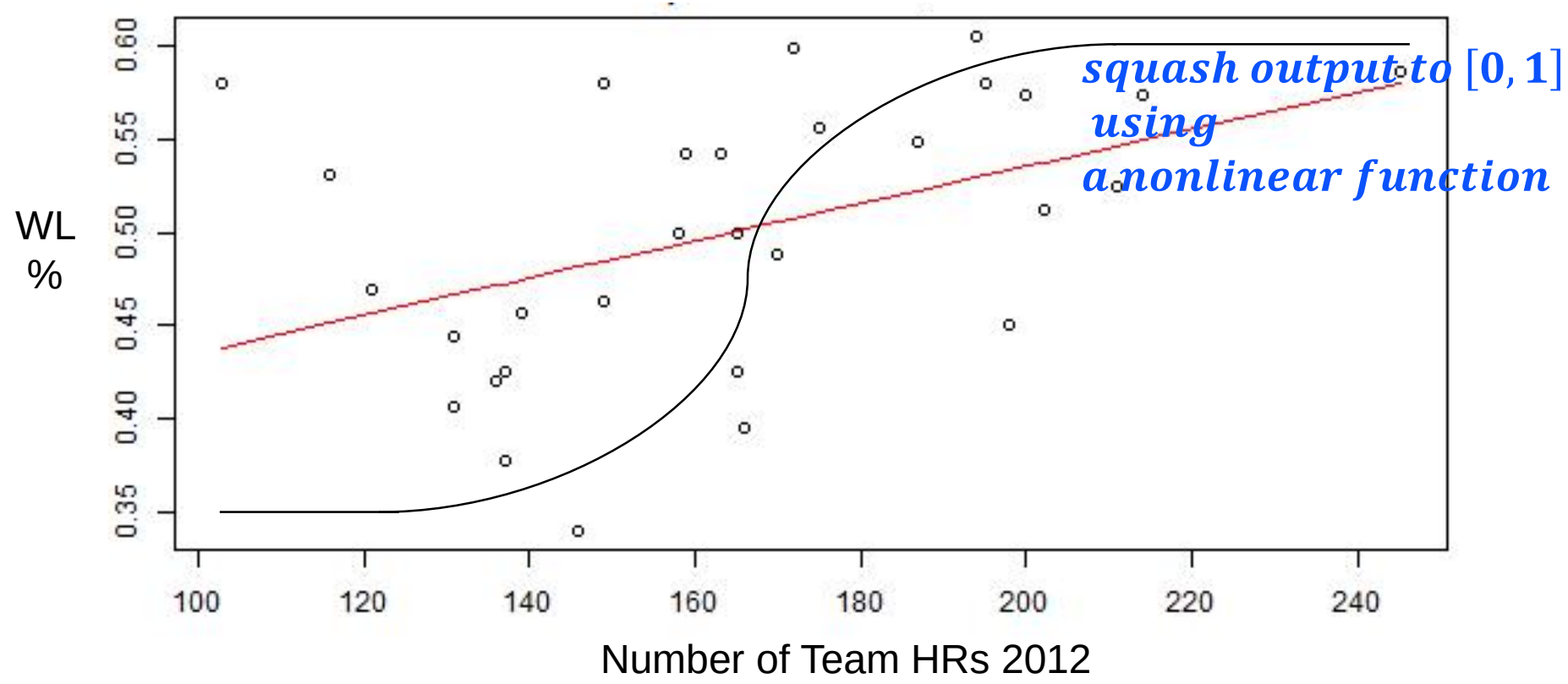
For Classification choose a decision function:

e. g. For y > 0. 5, Classify as Winner



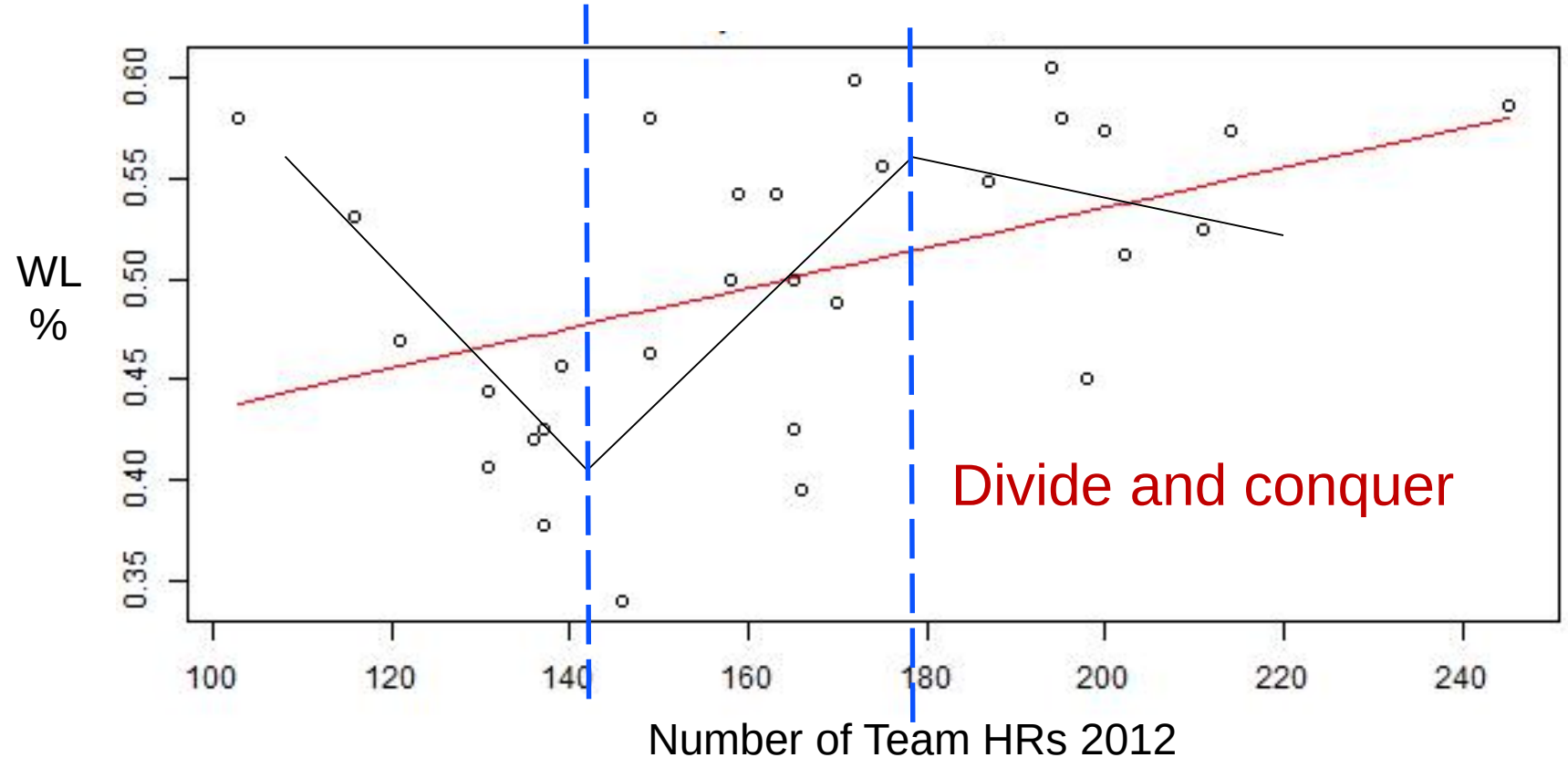
Regression with Logistic Function gives Probability

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



Piecewise Regression

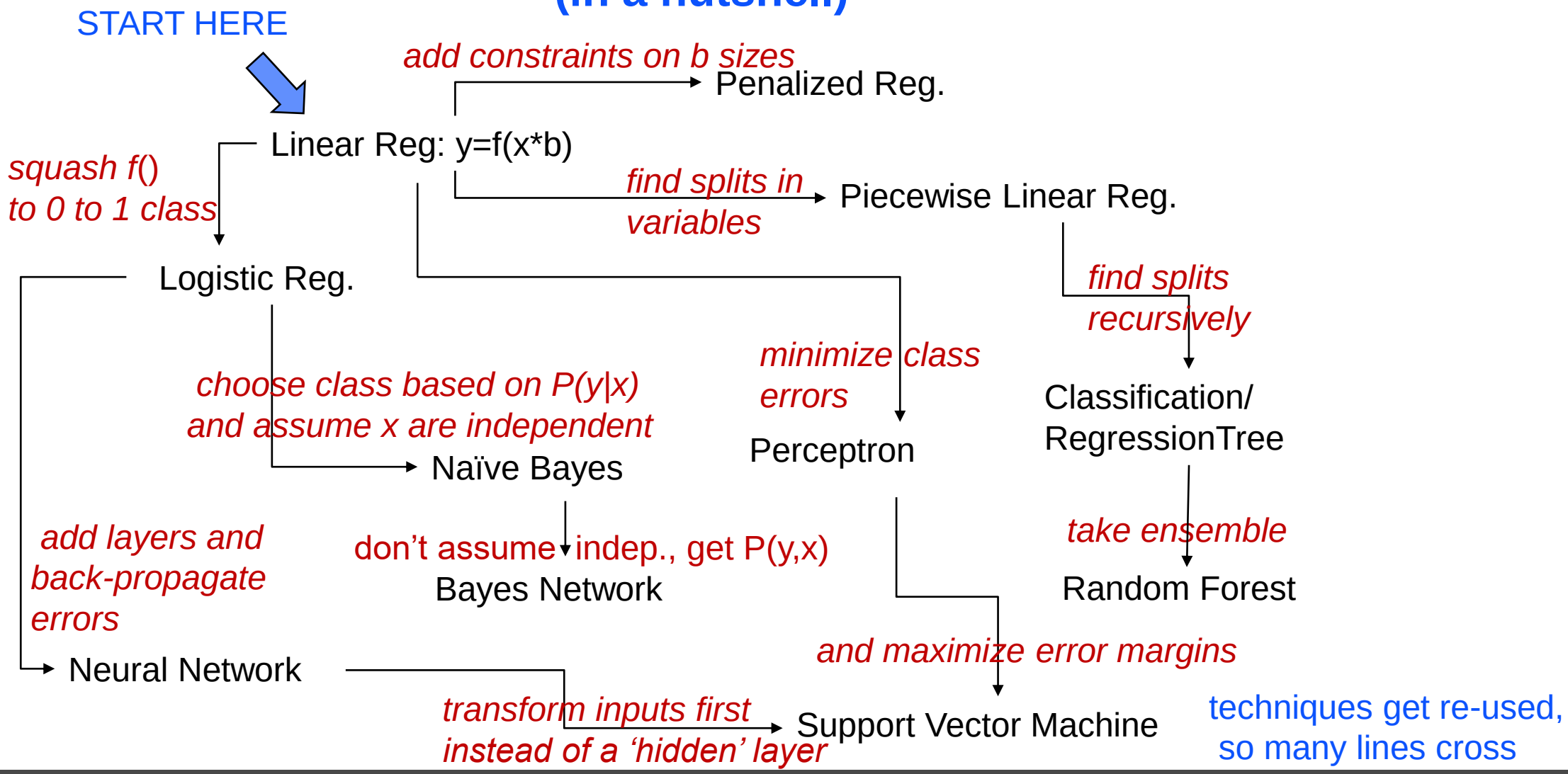
the Model: $y_i = f(x, b) = bo * 1 + b_1 * xi \quad for\ each\ x\ segment$



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

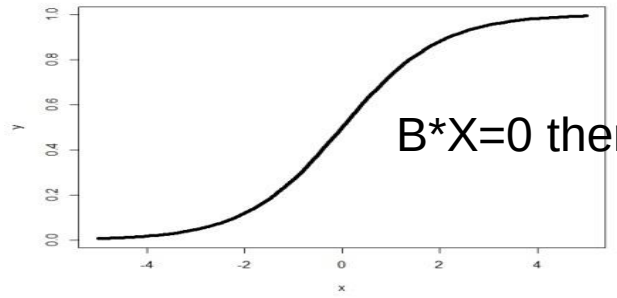
A Map of Machine Learning Models
(in a nutshell)



Linear to Logistic to Neural Network model

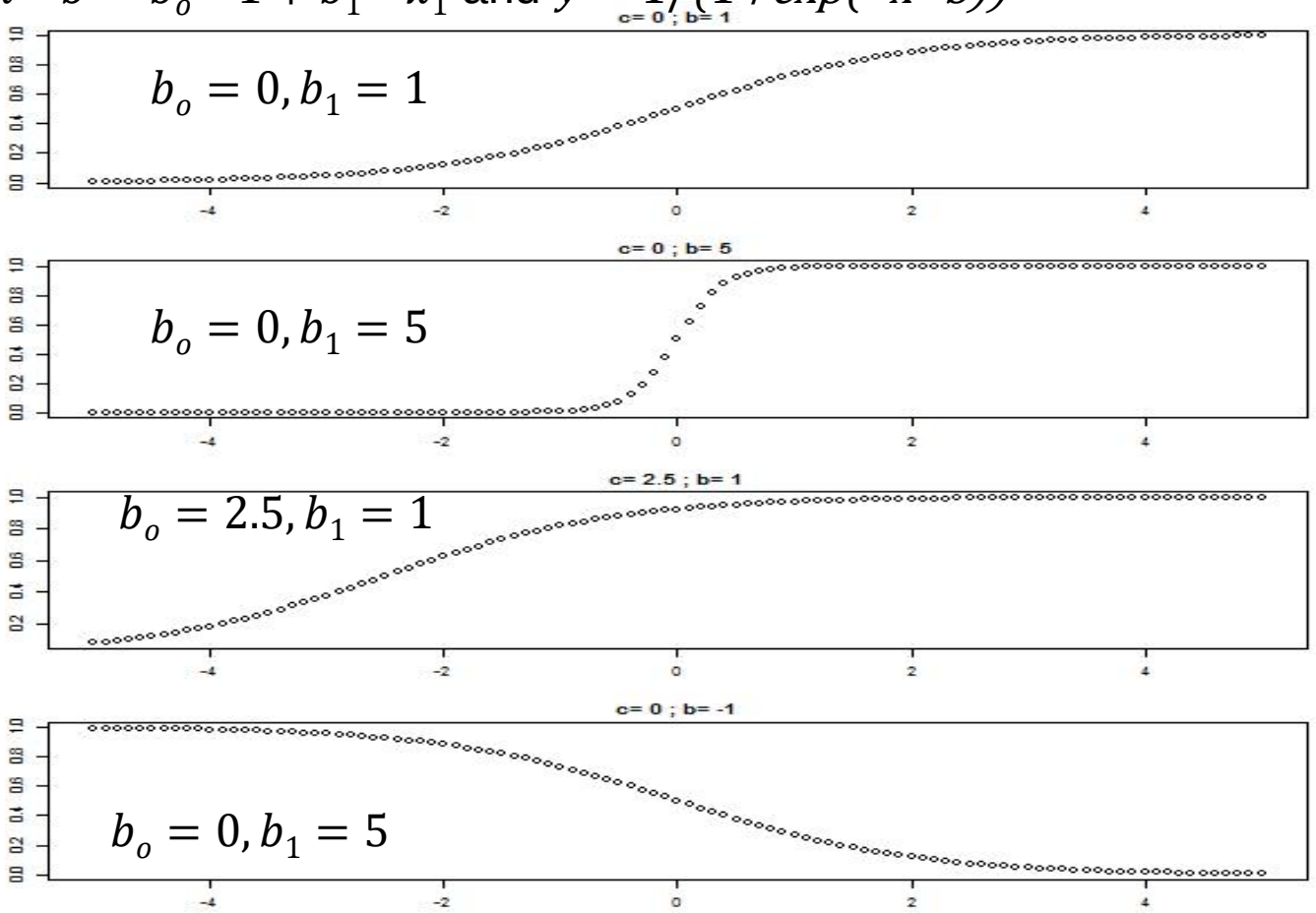
- $y_1 = b_o * 1 + b_1 * x_1 + b_2 * x_2 \ldots$ **= B*X**

- **Squash $b_o * 1 + b_1 * x_1$ to 0,1 range using Logistic Function:**



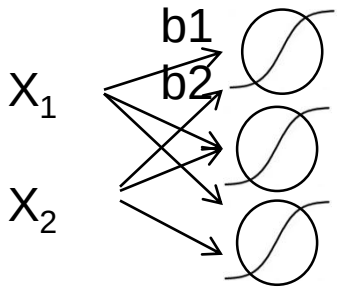
Logistic function w/various B

$for\ x * b = b_o * 1 + b_1 * x_1$ and $y = 1/(1+exp(-x*b))$

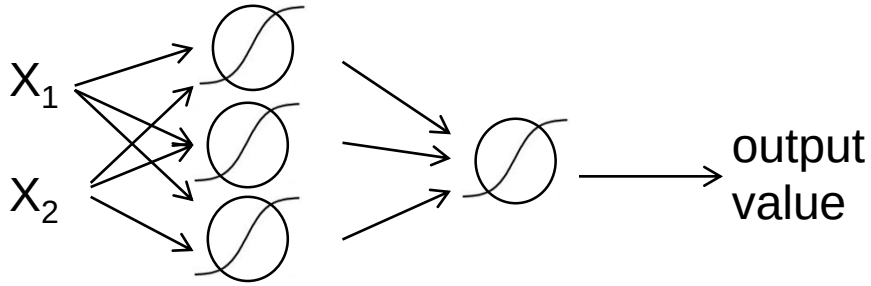


Logistic Regression to Neural Networks

- **Use several squash functions (hidden layer)**



- **Take final combination (output layer)**



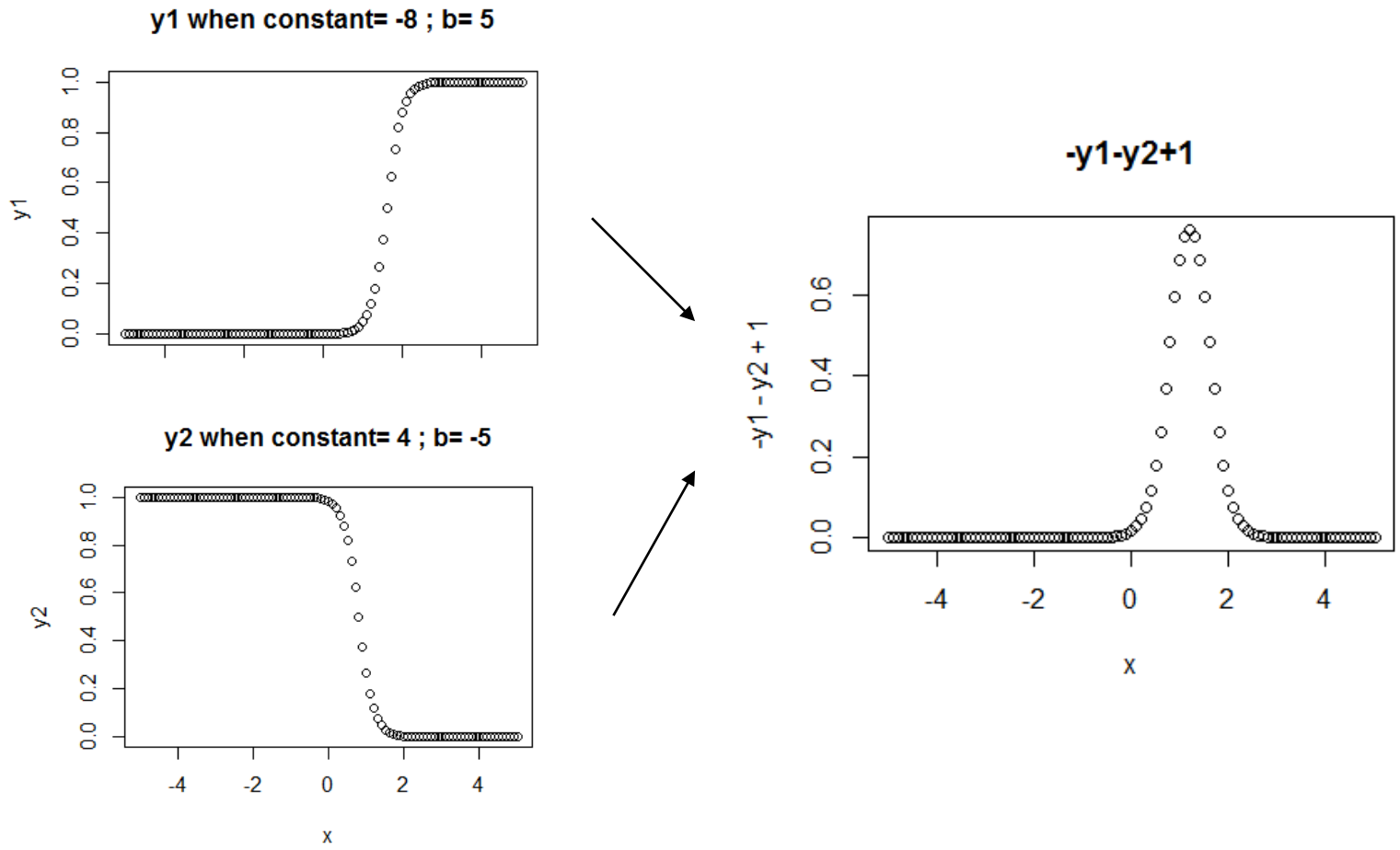
- **More powerful but more complex**

  many parameters, many options, needs more training

Logistic function combination

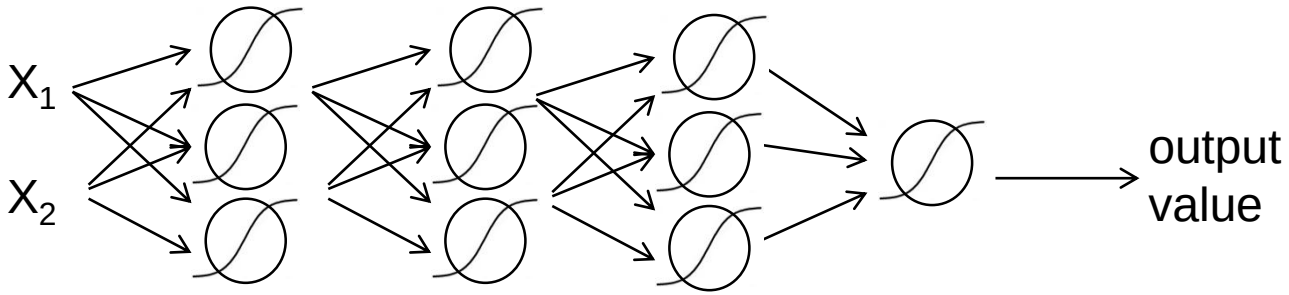
A new ‘layer’ of logistic functions enables highly nonlinear functions.

They can be considered new features.



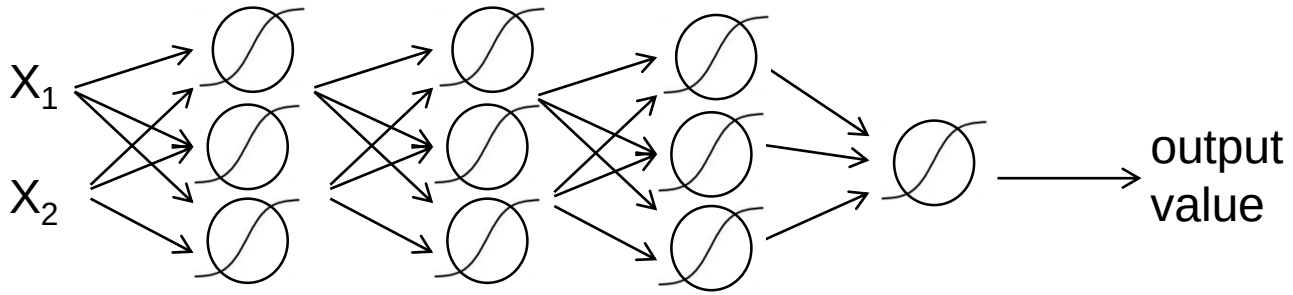
Logistic Regression to Neural Networks

- More hidden layers => More varied features, or ‘Deep’ Learning



Logistic Regression to Neural Networks

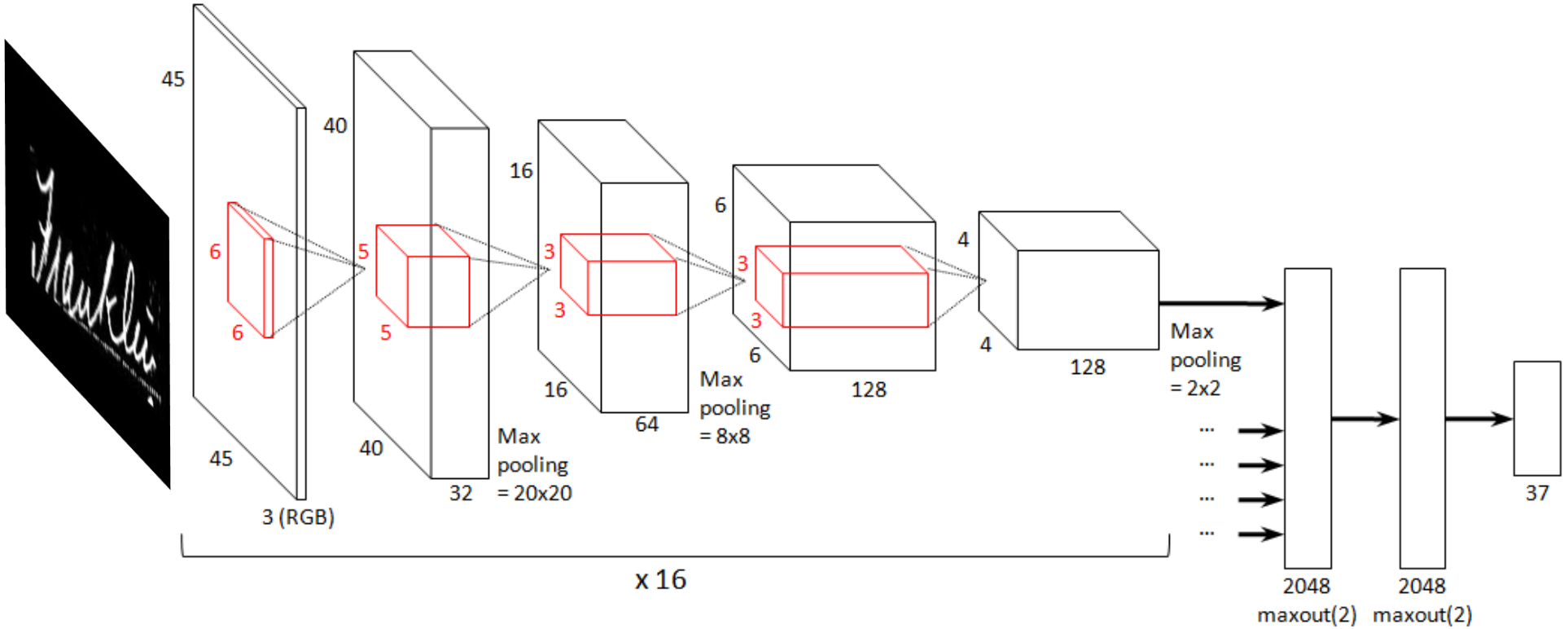
- **More hidden layers => More varied features, or ‘Deep’ Learning**



Many more parameters, and error signal at final output layer gets drowned out for lower layers-

but penalizing weight sizes, network ensembles, and more data help!

Organize connections into cells, add layers (deepen), add special pooling operations at some layers - you get a convolution network – good for image processing



Machine Learning In practice

- **Complexity: more parameters => more complex,**

  Sometimes you need heuristics for optimization procedures (ie training)

  Often good to start simple – it helps with interpreting results

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

  validation with test data 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, scaling variables
4. **Build and Evaluate Model**

Data Mining also:

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

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**

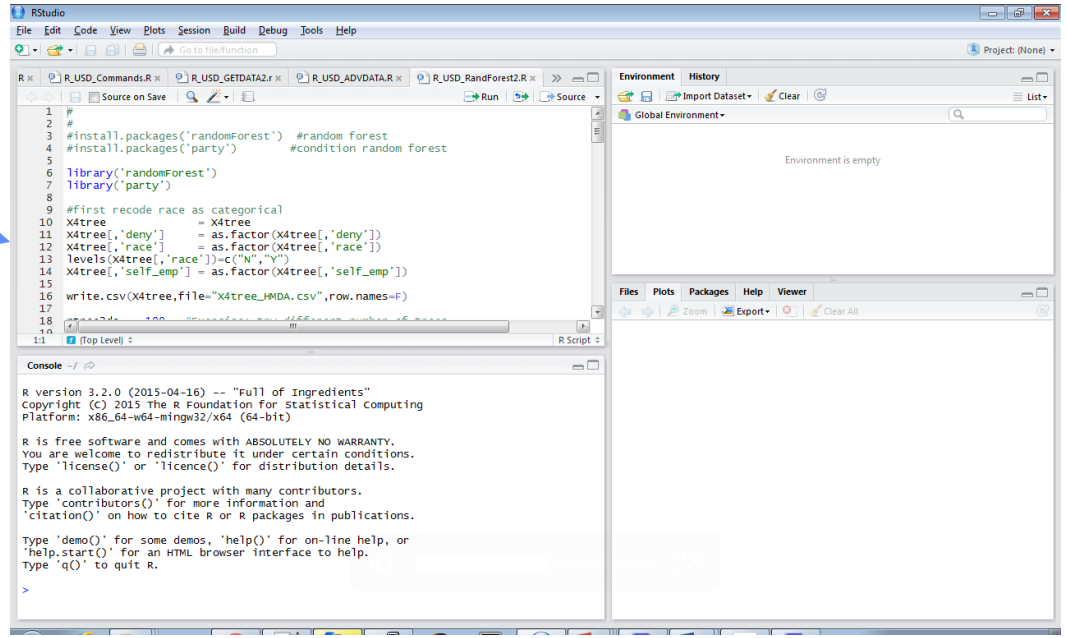
A typical R development workflow

- **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 docs, package lists

R commands in brief

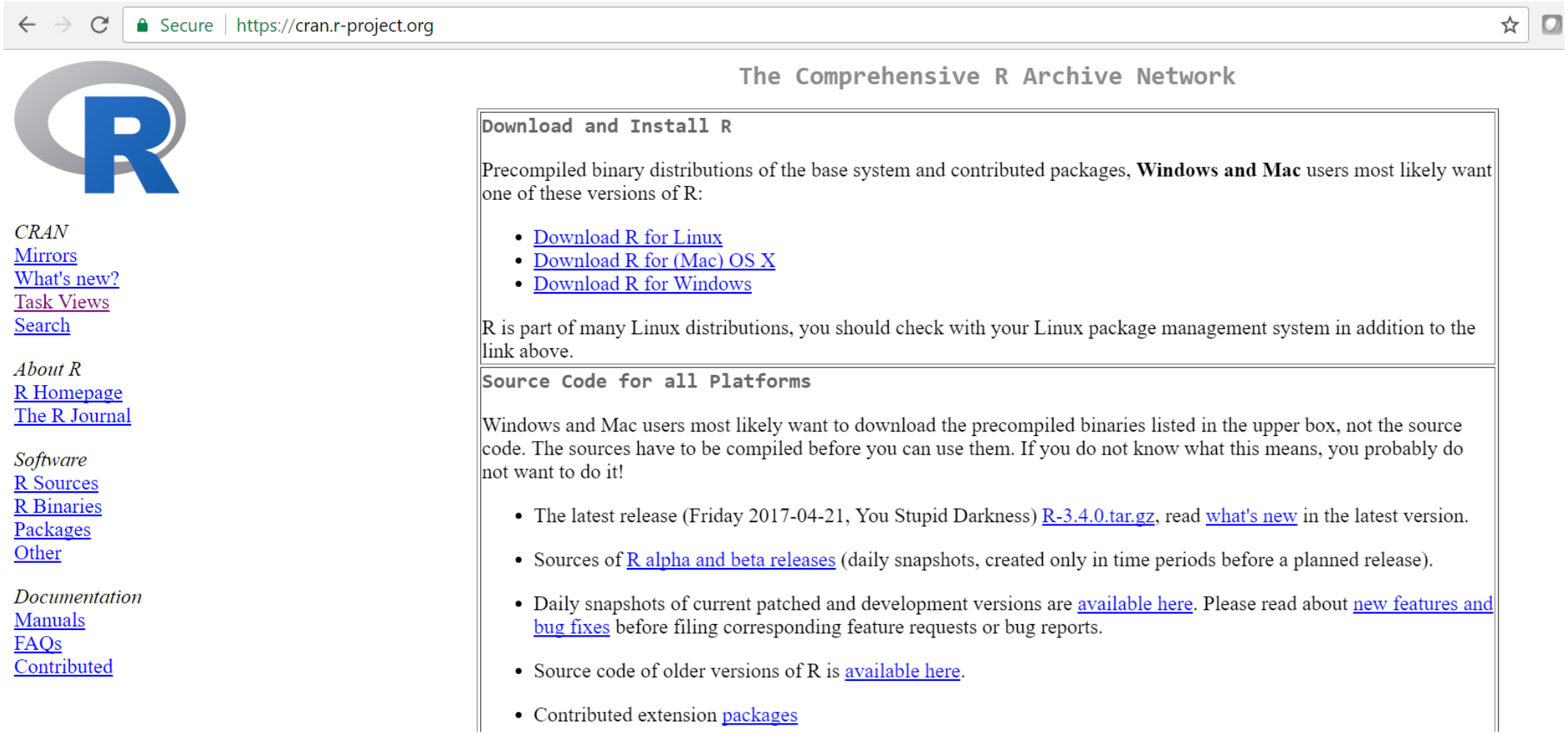
- A typical R code workflow:

```
#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)) #make race values 1-4 into values 0 or 1
deny                 = as.numeric(X[,'s7']==3)            #make deny values into 0 or 1,
                                                                1 only for deny='3'
#RUN MODEL and SHOW RESULTS
lm_result            =lm(deny~race+pi_rat)          #lm is 'linearmodel'
summary(lm_result)
```

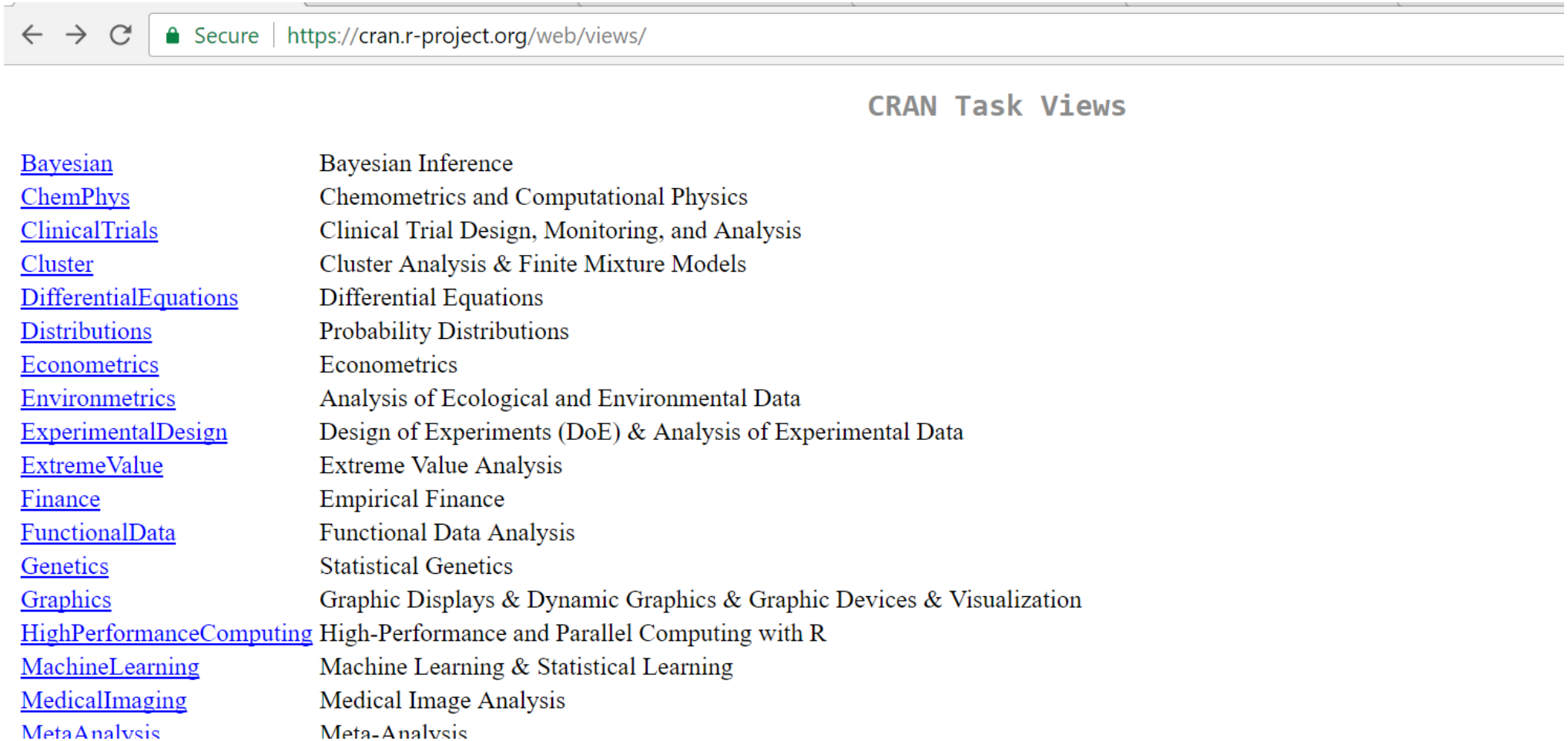
R strengths

- **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
  - Matching subjects for pairwise analysis

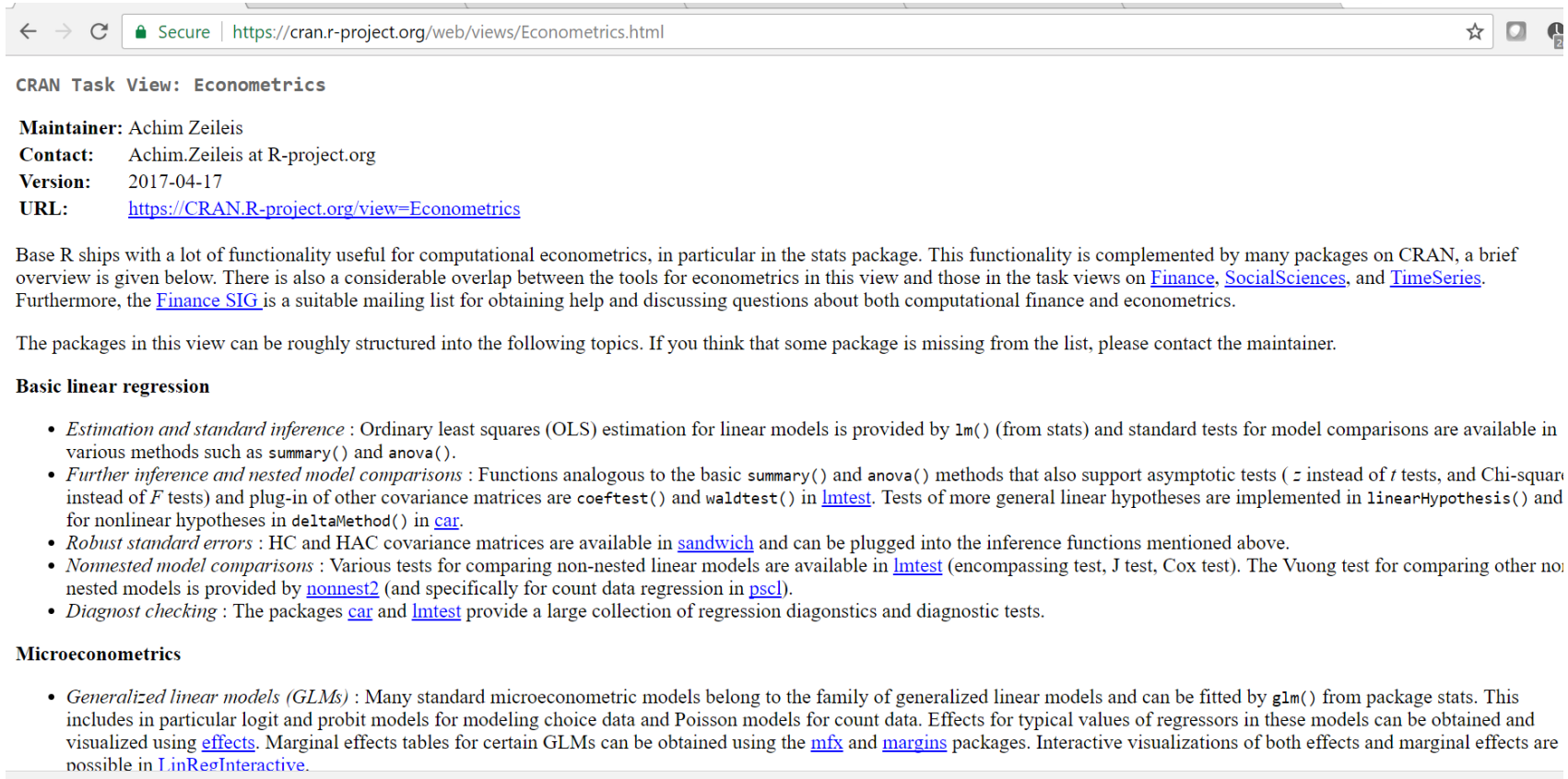
R websitie – everything you want to know…



e.g. CRAN -> Task Views



e.g. CRAN -> Task Views -> Econometrics



Secure | https://cran.r-project.org/web/views/Econometrics.html

CRAN Task View: Econometrics

Maintainer: Achim Zeileis
Contact: Achim.Zeileis at R-project.org
Version: 2017-04-17
URL: https://CRAN.R-project.org/view=Econometrics

Base R ships with a lot of functionality useful for computational econometrics, in particular in the stats package. This functionality is complemented by many packages on CRAN, a brief overview is given below. There is also a considerable overlap between the tools for econometrics in this view and those in the task views on Finance, SocialSciences, and TimeSeries. Furthermore, the Finance SIG is a suitable mailing list for obtaining help and discussing questions about both computational finance and econometrics.

The packages in this view can be roughly structured into the following topics. If you think that some package is missing from the list, please contact the maintainer.

Basic linear regression

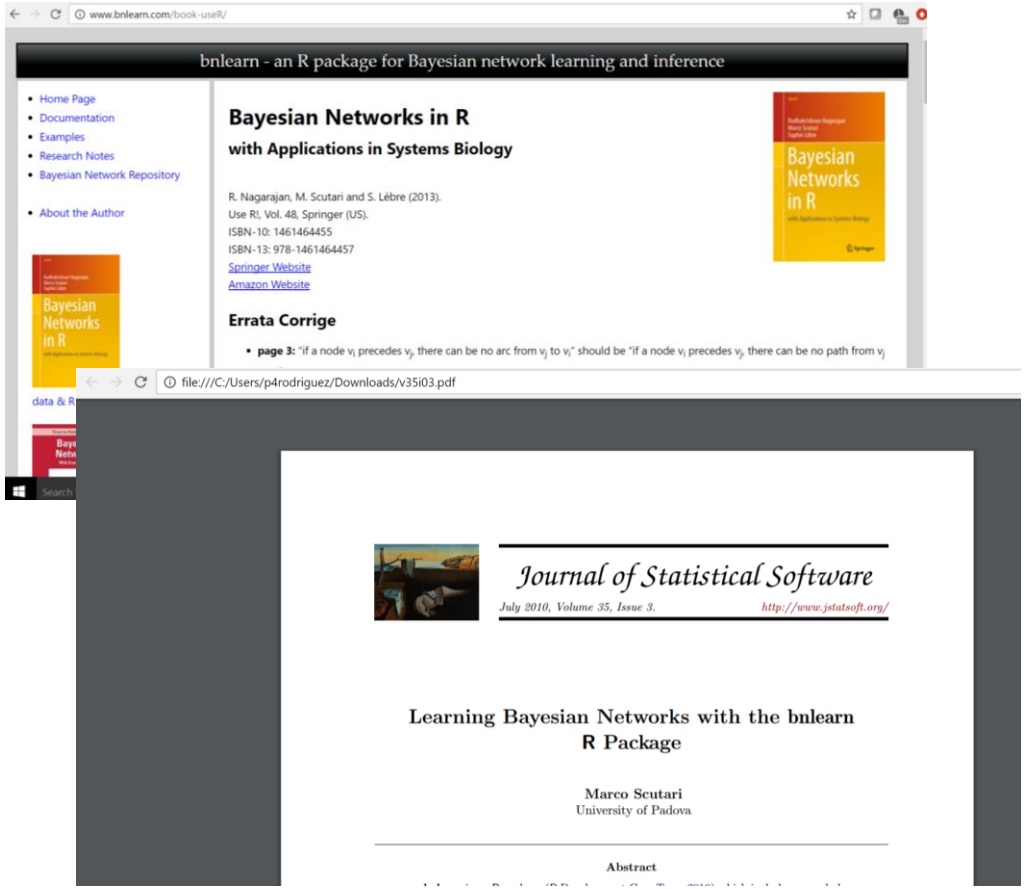
- *Estimation and standard inference* : Ordinary least squares (OLS) estimation for linear models is provided by `lm()` (from stats) and standard tests for model comparisons are available in various methods such as `summary()` and `anova()`.
- *Further inference and nested model comparisons* : Functions analogous to the basic `summary()` and `anova()` methods that also support asymptotic tests (z instead of t tests, and Chi-squar instead of F tests) and plug-in of other covariance matrices are `coeftest()` and `waldtest()` in lmtest. Tests of more general linear hypotheses are implemented in `linearHypothesis()` and for nonlinear hypotheses in `deltaMethod()` in car.
- *Robust standard errors* : HC and HAC covariance matrices are available in sandwich and can be plugged into the inference functions mentioned above.
- *Nonnested model comparisons* : Various tests for comparing non-nested linear models are available in lmtest (encompassing test, J test, Cox test). The Vuong test for comparing other no nested models is provided by nonnest2 (and specifically for count data regression in pscl).
- *Diagnost checking* : The packages car and lmtest provide a large collection of regression diagonstics and diagnostic tests.

Microeconometrics

- *Generalized linear models (GLMs)* : Many standard microeconometric models belong to the family of generalized linear models and can be fitted by `glm()` from package stats. This includes in particular logit and probit models for modeling choice data and Poisson models for count data. Effects for typical values of regressors in these models can be obtained and visualized using effects. Marginal effects tables for certain GLMs can be obtained using the mfx and margins packages. Interactive visualizations of both effects and marginal effects are possible in LinRegInteractive.

But how do you know which package to choose?

Other places to find guidance



Consider: R and missing data packages

- **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)

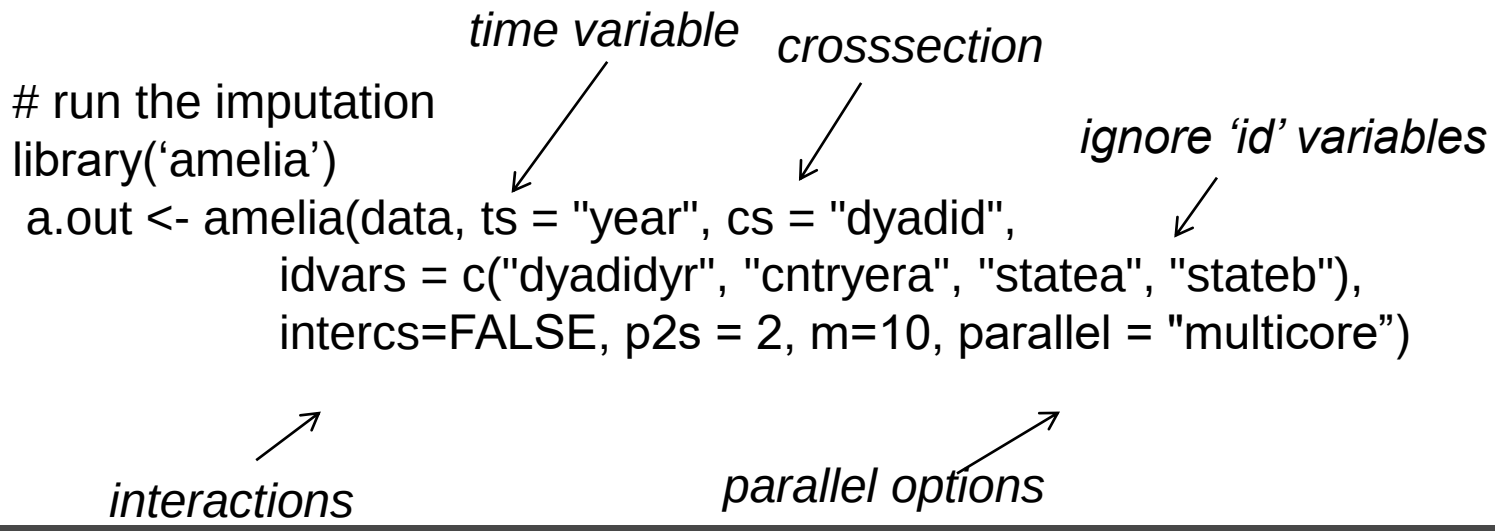
R and missing data

- **Sample R code using Amelia:**

  Data: UN conflict data in pairs of countries

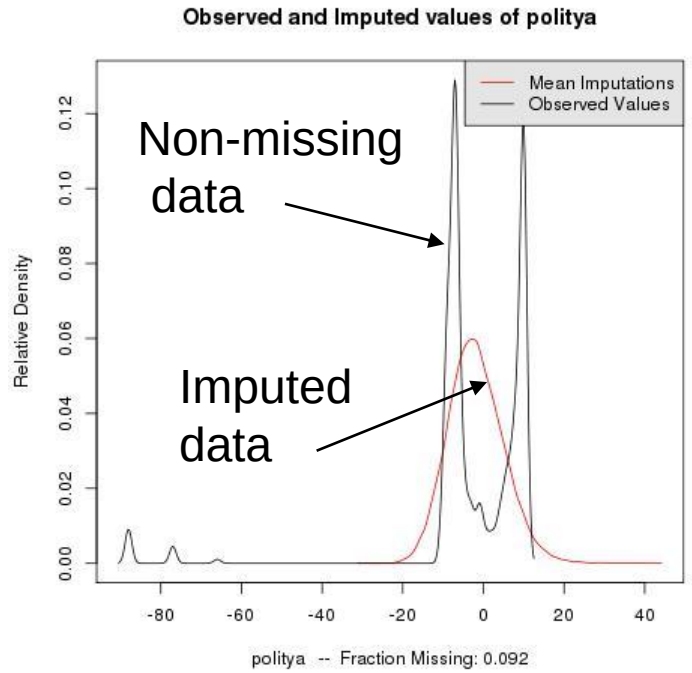
  300K rows ~ 1 hour on Gordon compute node (not run on the user's PC)

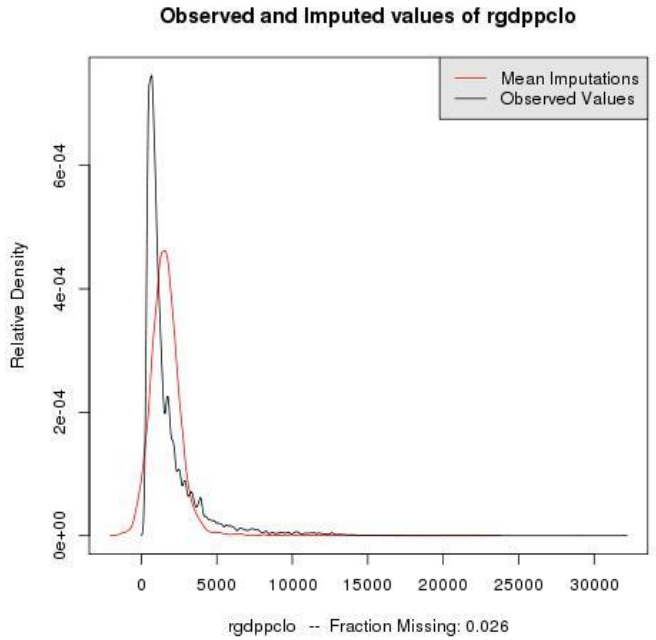
  1K-100K entries missing per col for about 20 of 50 cols



R and missing data

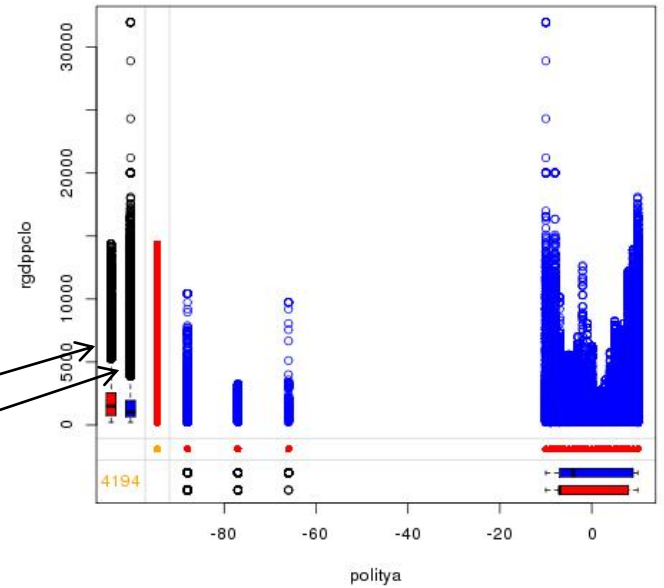
#Perform QA on missing data by comparing density of imputated & original data
compare.density(a.out, var="politya")
compare.density(a.out,var='rgdpcontg')





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')
```

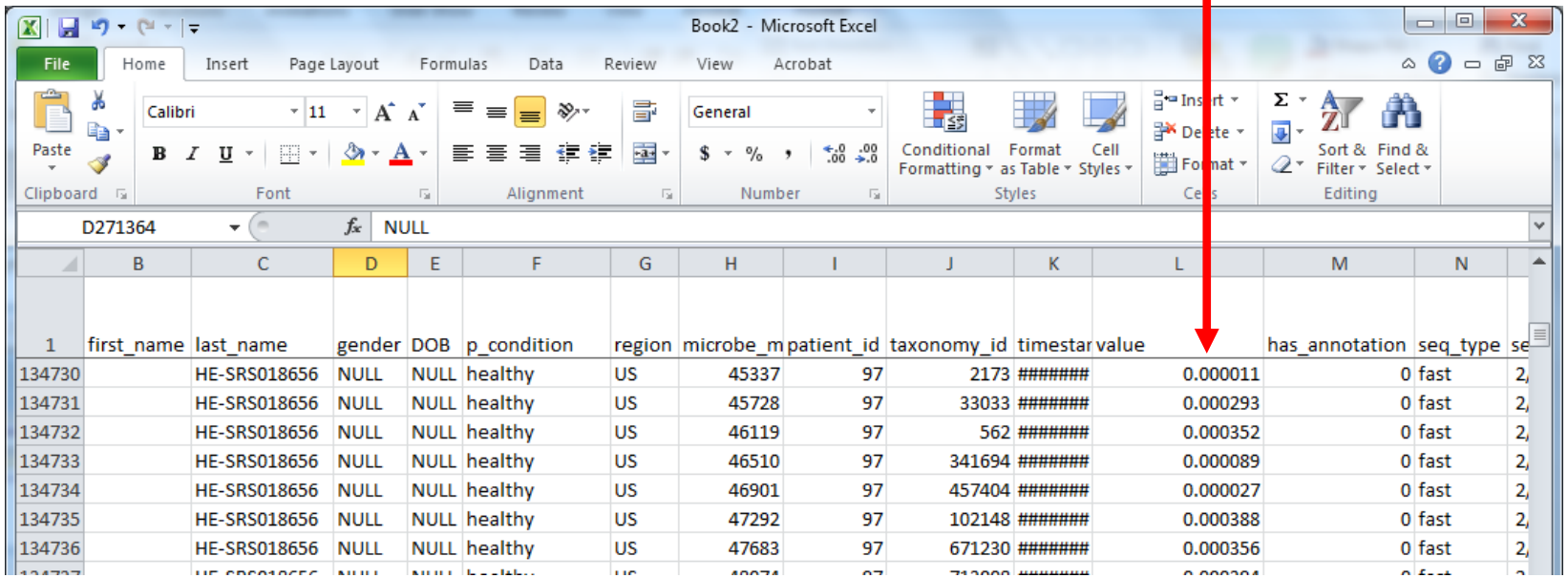


dist of rgdpcontg when politya missing

politya present

A Data Wrangling example

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



	B	C	D	E	F	G	H	I	J	K	L	M	N	
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

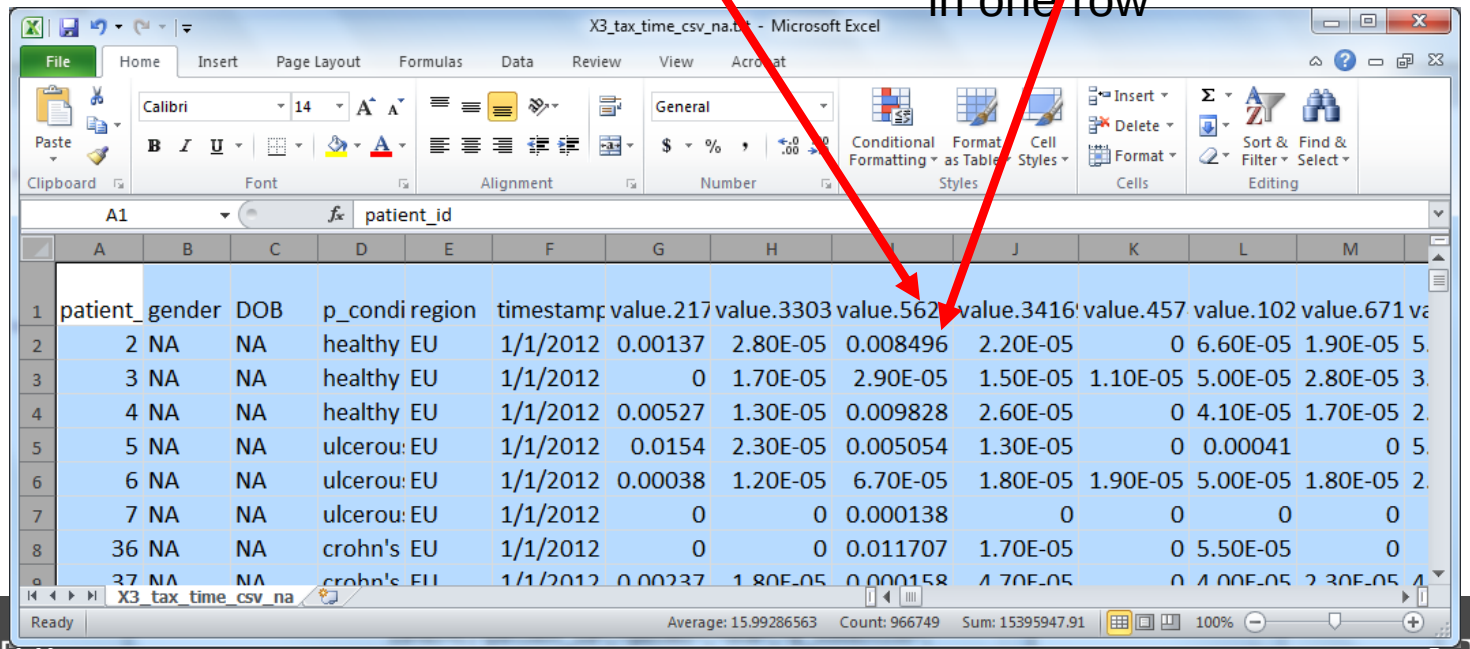
Reshape command to get a 'wide' format with repeated measurements in separate columns of the same record

```
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 keys

column keys

measurement values are now columns in one row

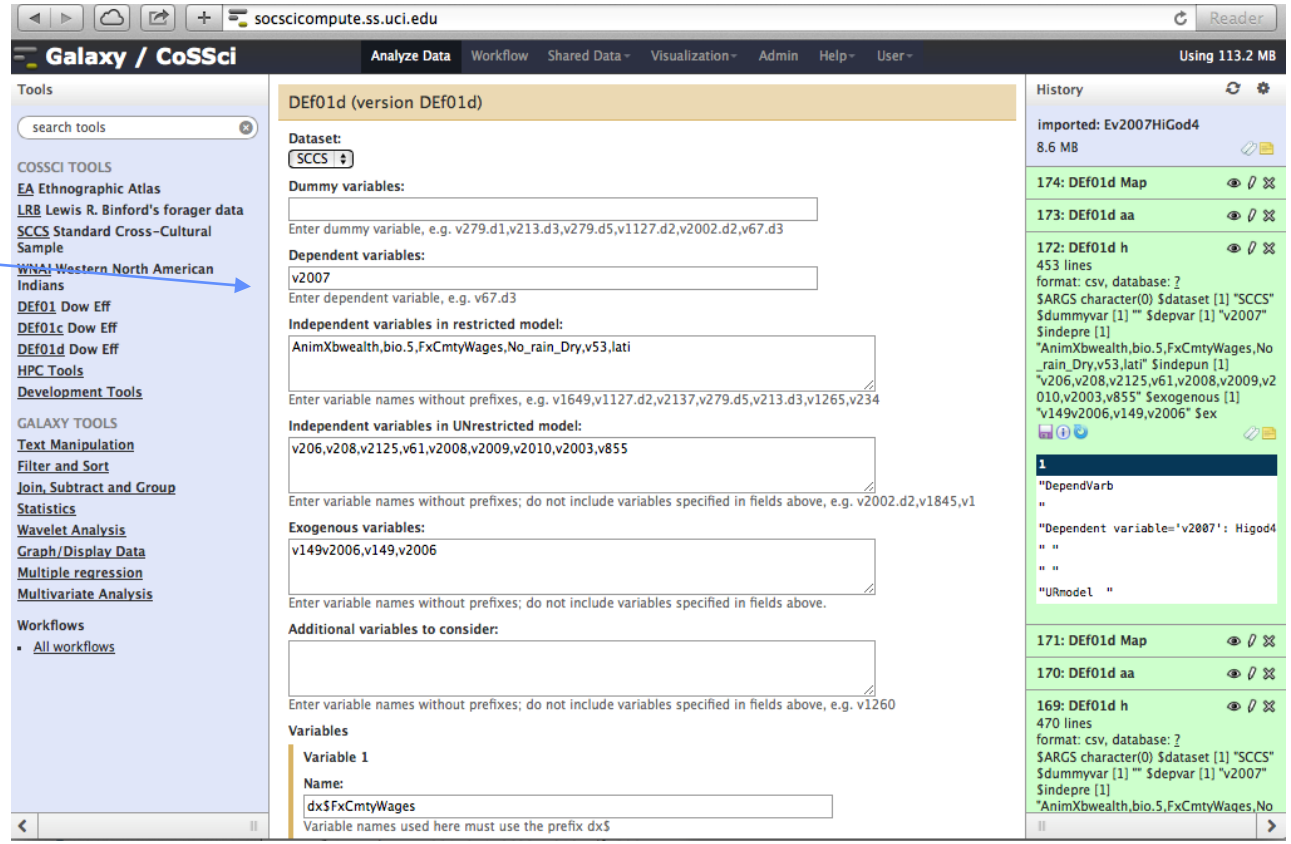


P.R. SDSC UCSD

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

Select dataset,
Select varialbes,
Submit analysis

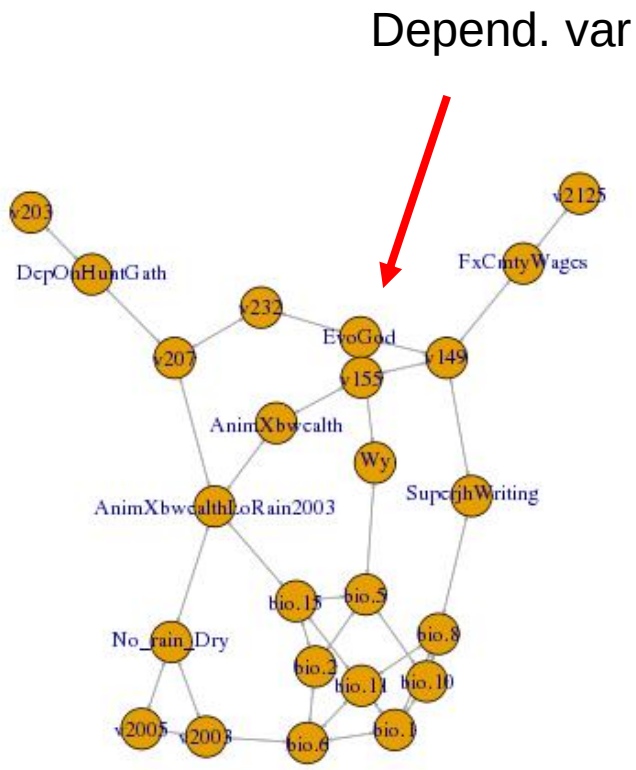
http://socscicompute.ss.uci.edu/



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**



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

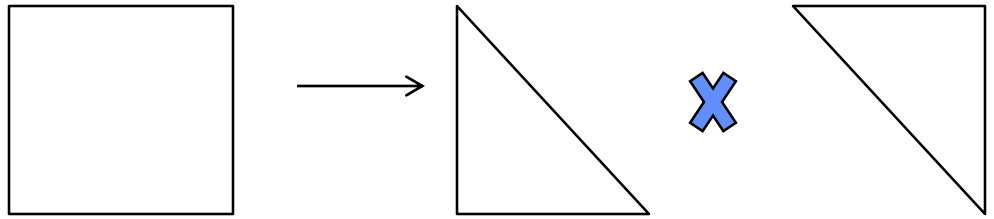
- **R takes advantage of math libraries for vector operations**
- **R packages provide multicore, multinode (snow), or map/reduce (RHadoop) options**
- **However, model implementations not necessarily built to use parallel backends**
  - Some models more amenable to parallel versions

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

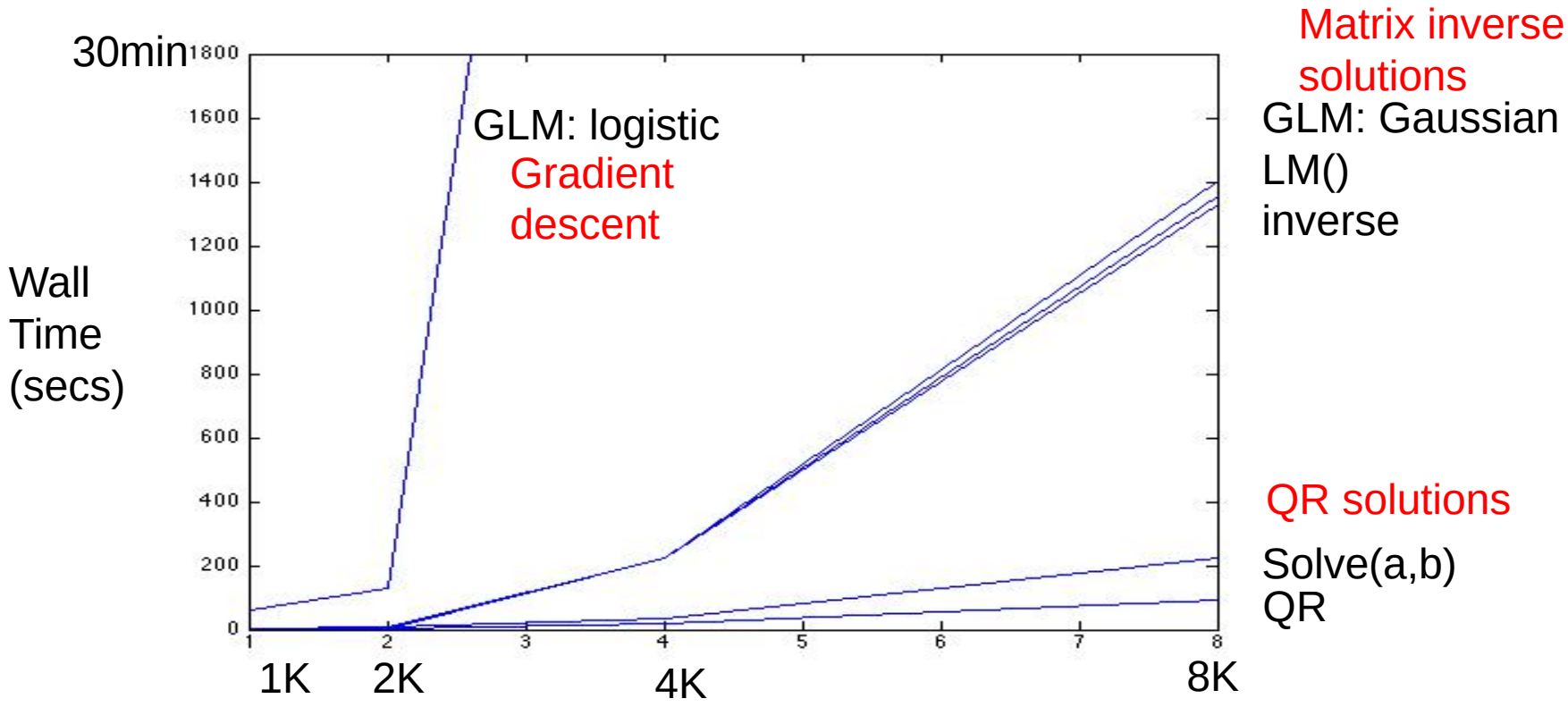
  glm() #Generalized Linear Model
  (logistic regression, etc)

  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()

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

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

allocate workers

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

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()
```

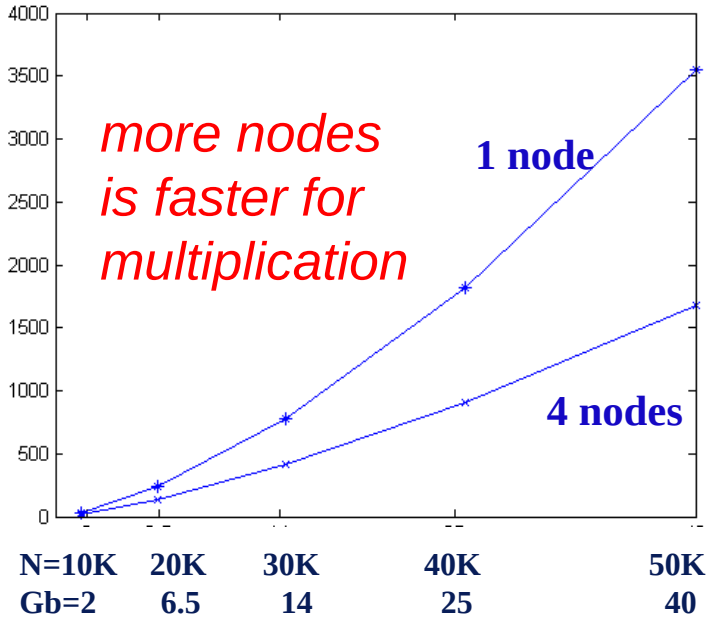
allocate cluster as parallel backend

%dopar% puts loops across cores and nodes

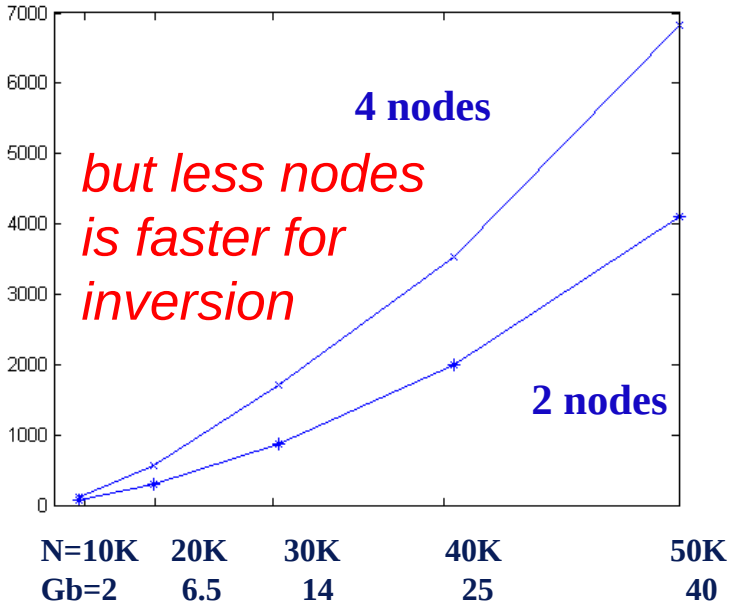
Multiple Compute Nodes not always help

(tested on Gordon)

Matrix Multiplication



Matrix Inversion



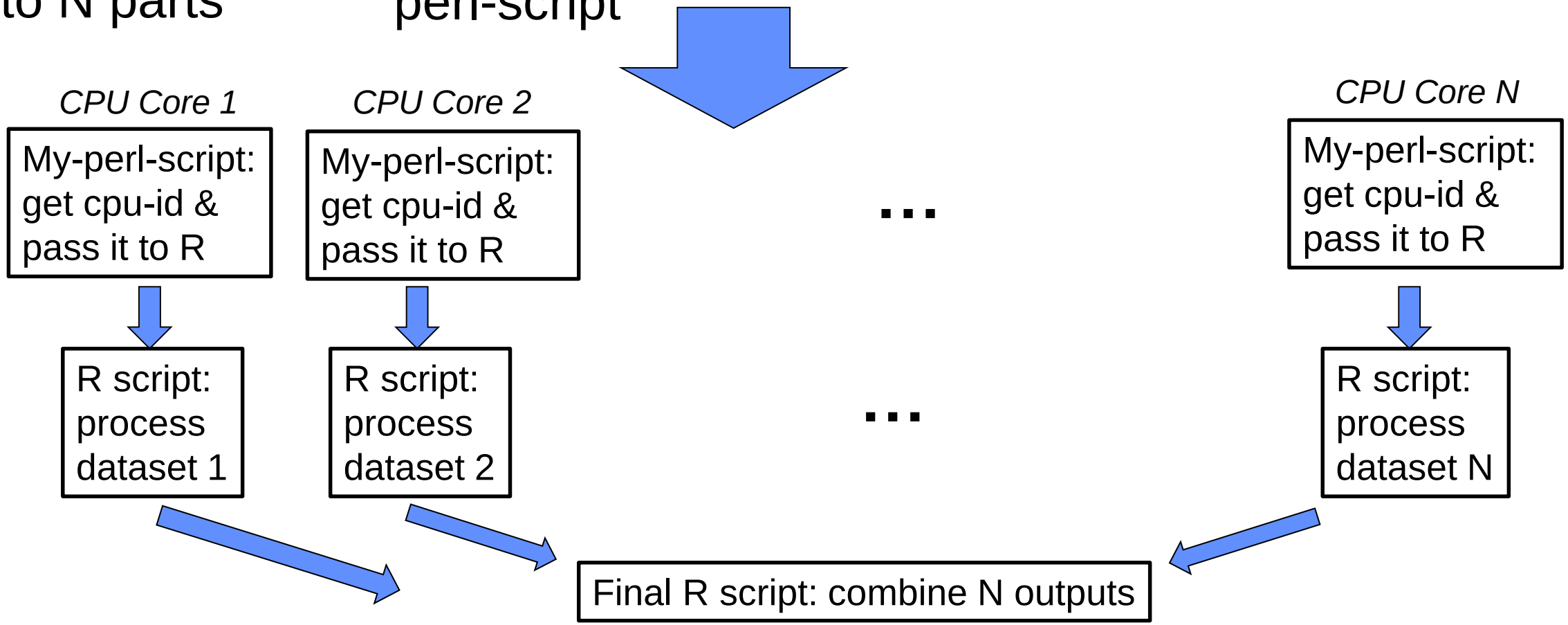
time

Square Matrix size

Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. Slurm Batch Script: mpirun -N processors My-perl-script



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

- **Option 1: Run separate trees on separate cores**

```
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
})
```

allocate workers

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

returned items 'combined' into list

Sampling on large data could be huge

Serial Packing with large Random Forest job

- **Randforest takes bootstrap sampling of data and creates an ensemble of decision trees**

SO Option 2:

split sampling to make it embarrassingly parallel

run R script on separate CPU cores

average results

Serial Packing with large Random Forest job

- *A GWAS (genome wide analysis) study (~20K subjects, ~80K SNPs):*
  1. Take 1000 samples on subjects and SNPs
  2. Run 50 trees (with resampling again) on each sample => 50K total trees
  3. Run 1 instance of R on each core: 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 parallelization 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)

R, Python, or Matlab?

- **My very rough classification:**

Python: free, more ‘computer scientist like’

Matlab: $, more ‘engineer like’

R: free, more ‘statistician/experimentalist like’

All are mature environments and capable, preferences depend on specific needs

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:

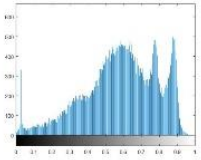
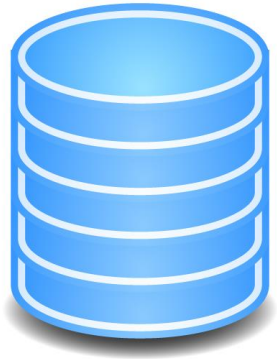
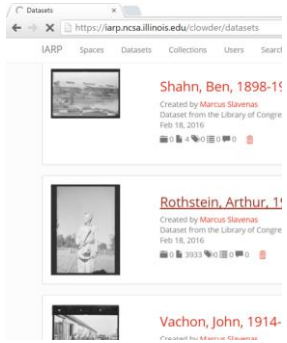
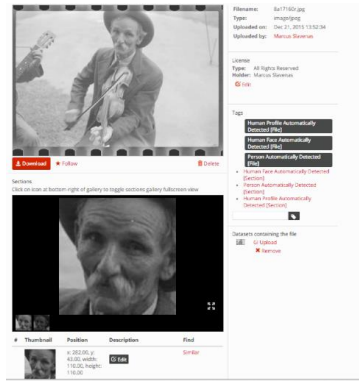


Image Content
OCR + RandomForest : BARBER SHOP; ENGLISH
FACE DETECTION: 1 face
Metadata
SEMANTICS:
<shop::business;structure;entity>
<barber::worker;person>
GEO : 41.2°N, 95.9°W
etc…

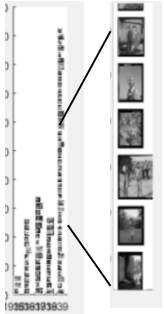






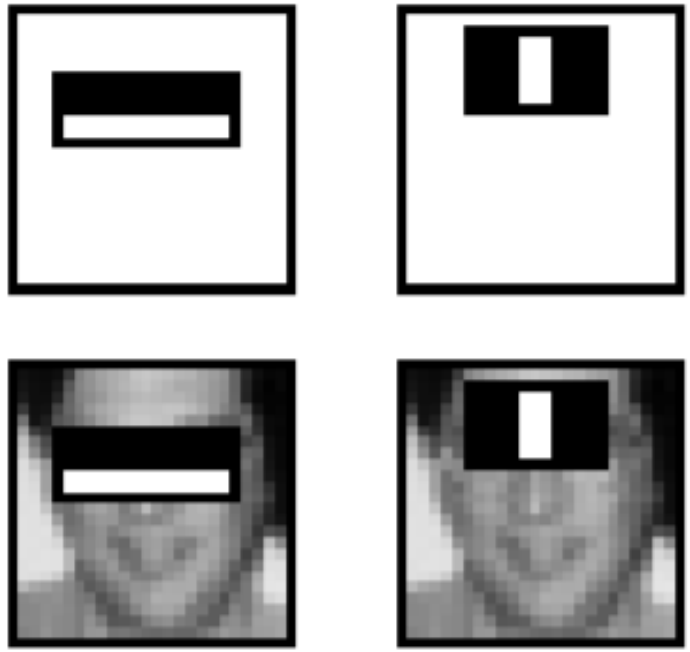
SQL with visualizations

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



Face Detection detail:

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



Turned 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**

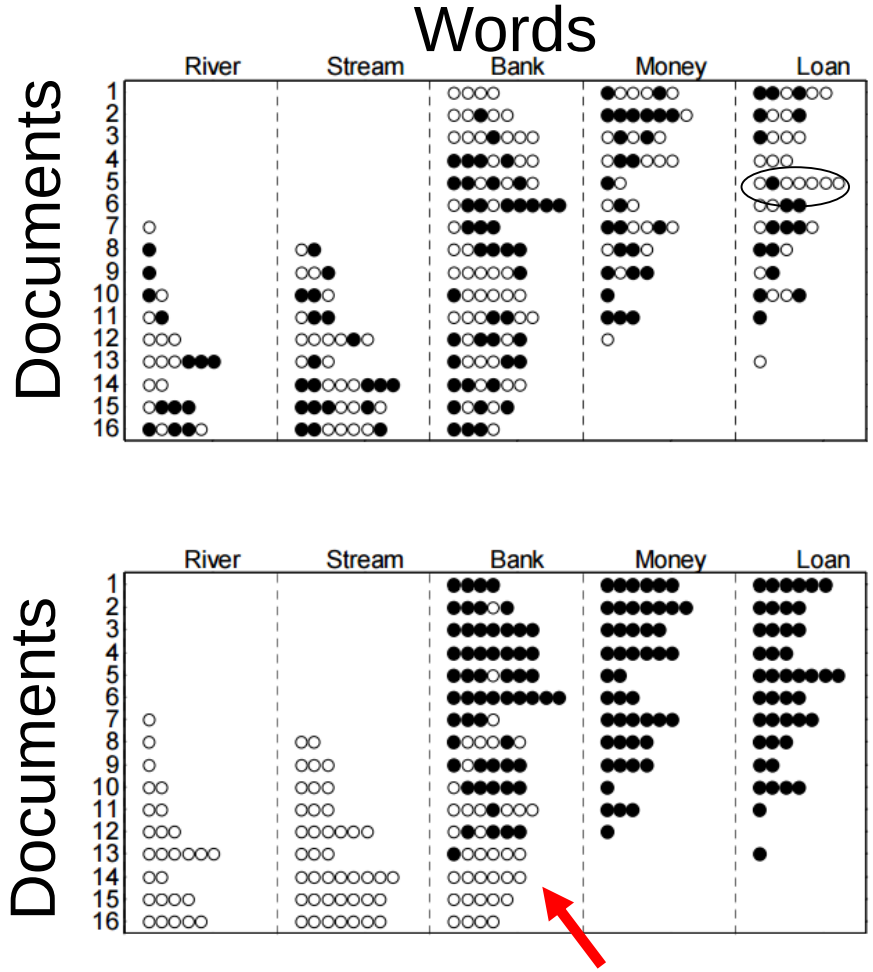
Title:

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

Topic Modelling with Latent Dirichlet Allocation

- **Example:**
  Get docsXwords matrix
  Start with random assignments

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



LDA optimization

- **Model: List of K topics and V word probabilities:**
  - *P(word|topic)*
  - *P(topic|document)*
  - *K,V given by user*
- **Optimization:**
  - Start with initial guess of word, topic assignments
  - Loop until convergence:
    - Compute the expected frequency of word=*w for each* topic=*t*
    - Adjust parameters to maximize likelihood *t given w*

- **Example**

Input: articles from JSTOR's Data for Research website (dfr.jstor.org) with certain key words

Output: tree map showing top words

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

THE END