Data Preparation

Paul Rodriguez

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

On the Importance of Data Prep

- “Garbage in, garbage out”

- Sometimes takes 60-80% of the whole data mining effort

Working definition

- **Data Preparation:**
  - Cleaning
  - Filtering
  - Transforming
  - Organizing the data matrix (aka ‘data wrangling’ or ‘data munging’)

In a nutshell, preparing data for modeling

Cleaning Noise

- Entity Resolution and Record Linkage

e.g. Are these equal?

West Main Street ***W Main St***

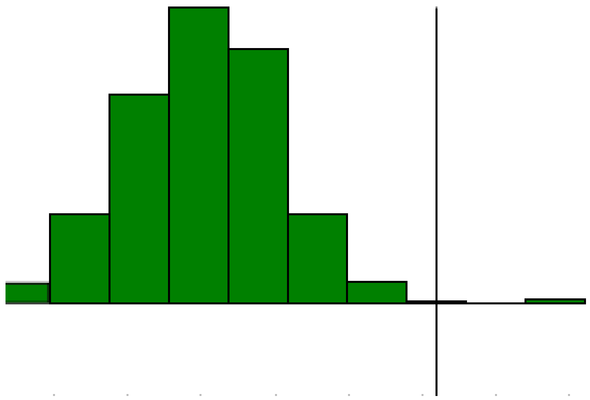
Strategy:

use dictionaries and search possible matches

Statistical Noise:

- Outliers

e.g. remove them,

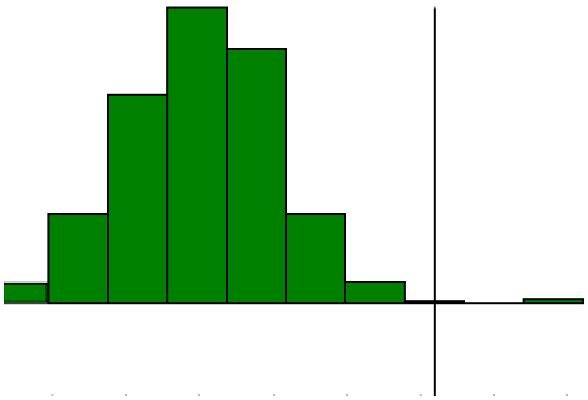


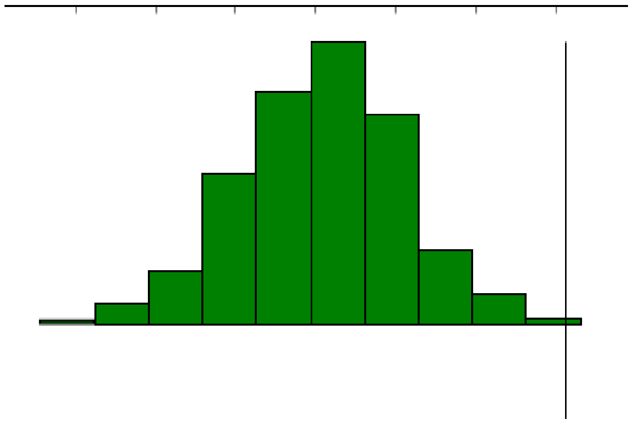
mean + 3*std-devm

Statistical Noise:

- Outliers

e.g. remove them, but cutoff is arbitrary





mean + 3*std-devm

Missing Data

- **Important to review statistics of a missing variable**
  - Are entries missing completely at random?
  - Are entries missing depending on some other variable?
  - What are the counts and combinations of missing entries among variables?
  - Is there a relation between missing cases and outcome variable?

Missing Data

- **Not applicable**

  e.g. spouse name depends on marital status

- **Not Available**

  unknown

  not entered

Missing Data

- Do missing cases depend on some other variable?

e.g. ‘CEOs’ don’t like to list their salary

Strategy: *get most common job titles*

for missing salaries

Quick Approaches

- Delete instances

and/or

- Delete attributes with high missingness

Quick Approaches

- Leave as ‘NULL’ category
  - Some algorithms implementation handle NULL (ie Decision Trees)

Simple Imputation

- Use the attribute mean (by class)

Complicated Imputation

- Use a model (based on other attributes) to infer missing value

Complicated Imputation

- **Use a model (based on other attributes) to infer missing value**

Best strategy depends on

time vs accuracy tradeoffs

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

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

  Note: mice package is probably more well known, and has similar options, but MCMC is slower than EM

time variable

crosssection

ignore 'id' variables

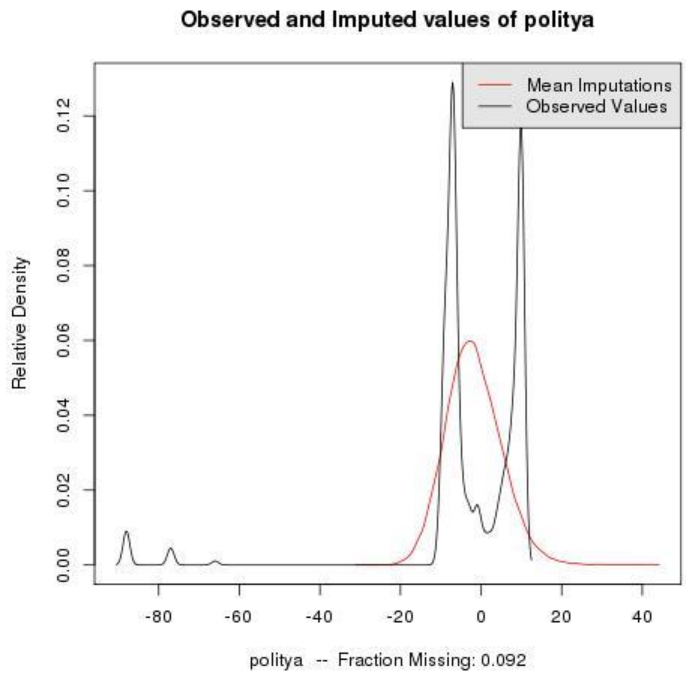
```
# run the imputation
library('amelia')
 a.out <- amelia(data, ts = "year", cs = "dyadid",
          idvars = c("dyadidyr", "cntryera", "statea", "stateb"),
          intercs=FALSE, p2s = 2, m=10, parallel = "multicore")
```

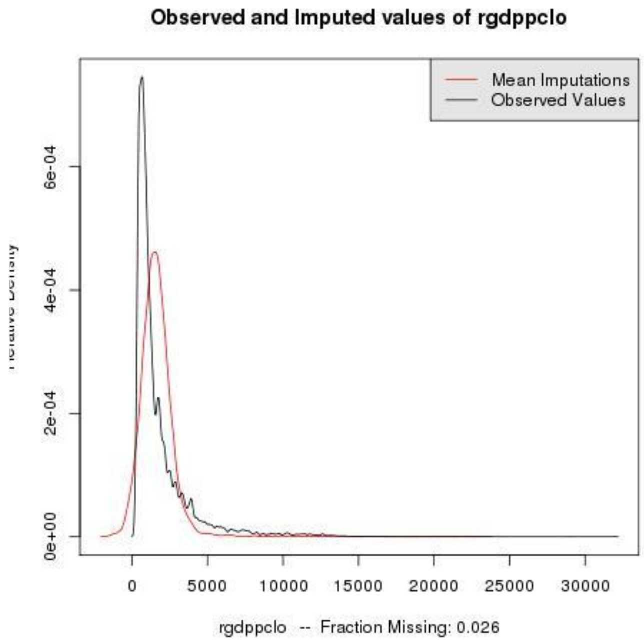
interactions

parallel options

R and missing data

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

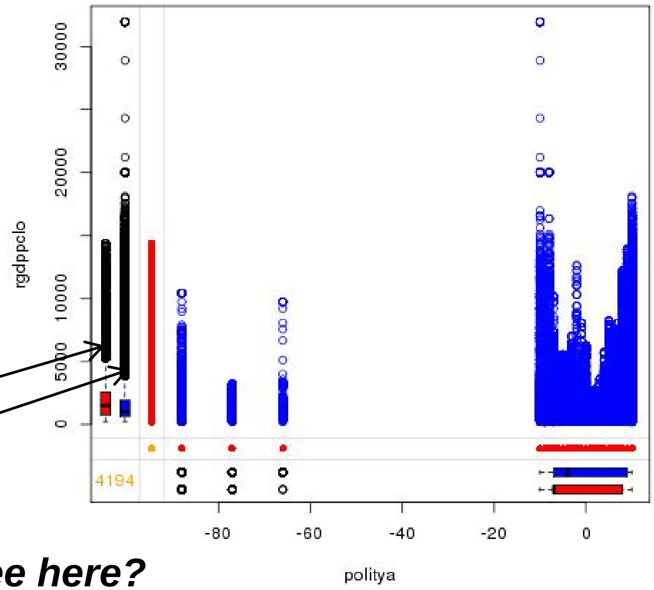




R and missing data

```
# Useful library for printing margin plots, to compare histograms
#                         conditional on missing/non-missing data
library('VIM')
marginplot(gart2use[,c('politya', 'rgdppclo')],
  col=c('blue','red','orange')
```

dist of rgdppclo when
politya missing
politya present



What would you want to see here?

Variable Transformations

- **Engineer new features**

- **Combine attributes**
  e.g. rates and ratios

- **Normalize or Scale data**

- **D**iscretize data
  (perhaps more intuitive to deal with binned data)

Feature Engineering is Variable Enhancement

- **Use Domain and world knowledge**
- **Examples:**

  given date and location of doctor visits

  - deduce a new variable for Number-of-1st-time-visits
  - deduce a new variable for Number-of-visits-over-25-miles
  - deduce a new variable for Amount-of-time-between-visits

Re-scaling

- **Mean center** $$x_{new} = x - \mathrm{mean}(x)$$
- **z-score** $$score = \frac{x - \mathrm{mean}(x)}{\mathrm{std}(x)}$$
- **Scale to [0…1]** $$x_{new} = \frac{x - \min(x)}{\max(x) - \min(x)}$$
- **log scaling** $$x_{new} = \log(x)$$

Variable selection

- **Heuristic methods:**

  remove variables with low correlations to outcome

  (other criteria: information gain, sensitivity, etc…)

- step wise: add 1 variable at a time and test algorithm on samples

Variable selection

- **Some algorithms are robust to extra noise variables**

- E.g. Least Angle Regression (L_1 penalty),

  - penalize small effect sizes (zero them out)

- E.g. Random Forest outputs ‘importance’

  - low importance implies non-influence in the model

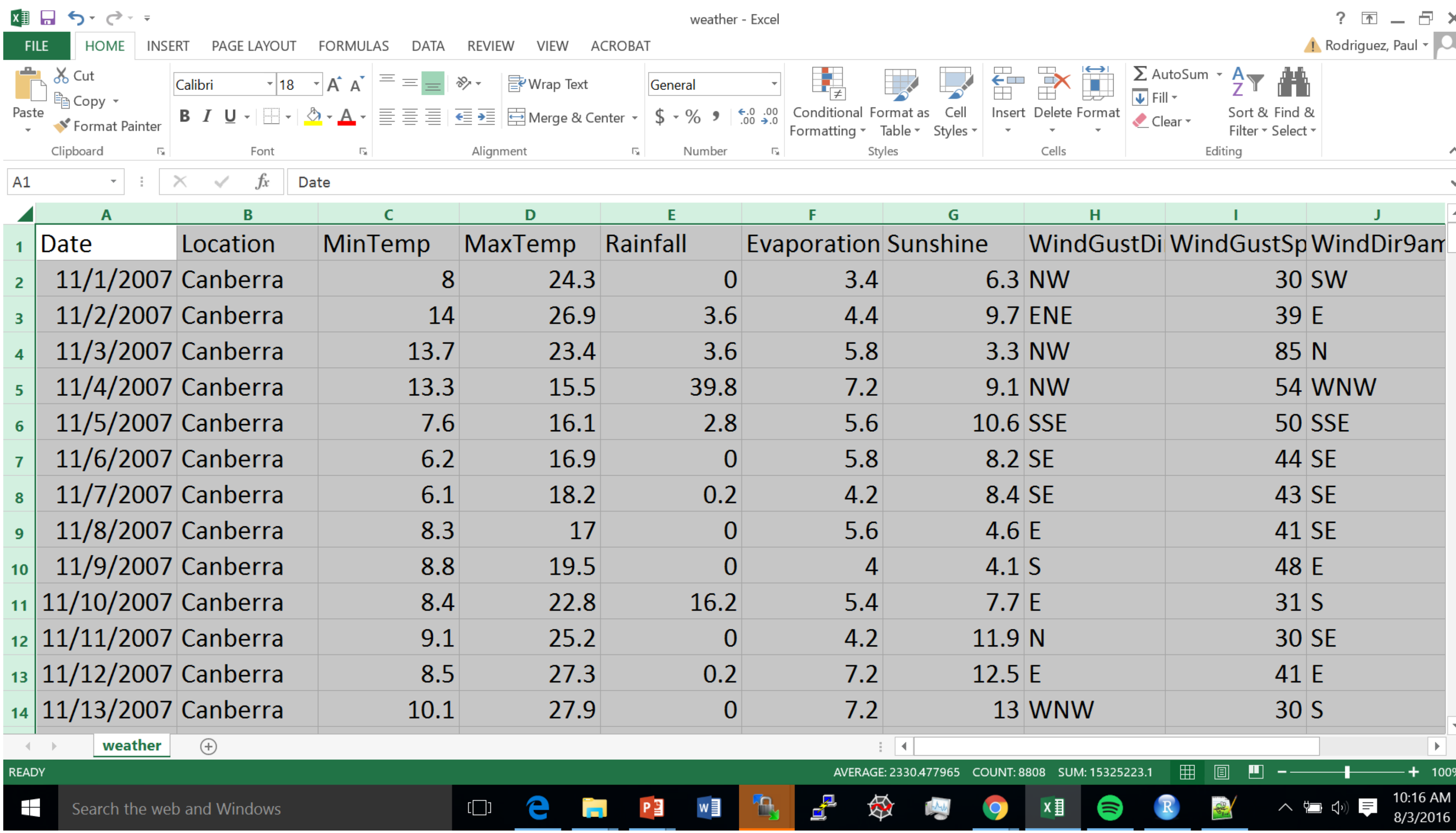
    (other criteria: information gain, sensitivity, etc…)

Summary

- **Preparing data is based on statistical principles,**

- **But also heuristics**

Data Prep Exercise: Weather Data



Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDi	WindGustSp	WindDir9am
11/1/2007	Canberra	8	24.3	0	3.4	6.3	NW	30	SW
11/2/2007	Canberra	14	26.9	3.6	4.4	9.7	ENE	39	E
11/3/2007	Canberra	13.7	23.4	3.6	5.8	3.3	NW	85	N
11/4/2007	Canberra	13.3	15.5	39.8	7.2	9.1	NW	54	WNW
11/5/2007	Canberra	7.6	16.1	2.8	5.6	10.6	SSE	50	SSE
11/6/2007	Canberra	6.2	16.9	0	5.8	8.2	SE	44	SE
11/7/2007	Canberra	6.1	18.2	0.2	4.2	8.4	SE	43	SE
11/8/2007	Canberra	8.3	17	0	5.6	4.6	E	41	SE
11/9/2007	Canberra	8.8	19.5	0	4	4.1	S	48	E
11/10/2007	Canberra	8.4	22.8	16.2	5.4	7.7	E	31	S
11/11/2007	Canberra	9.1	25.2	0	4.2	11.9	N	30	SE
11/12/2007	Canberra	8.5	27.3	0.2	7.2	12.5	E	41	E
11/13/2007	Canberra	10.1	27.9	0	7.2	13	WNW	30	S

Data Prep exercise

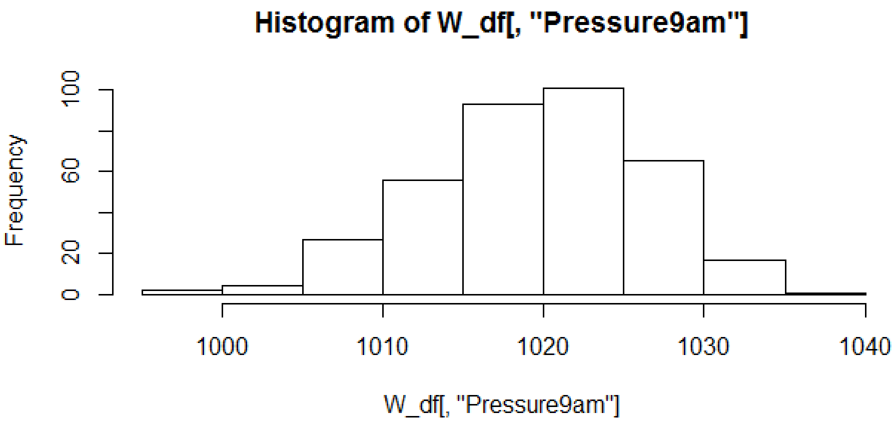
```
W_df = read.table('weather.csv',
                  header=TRUE,
                  sep=",",
                  stringsAsFactors = TRUE)
```

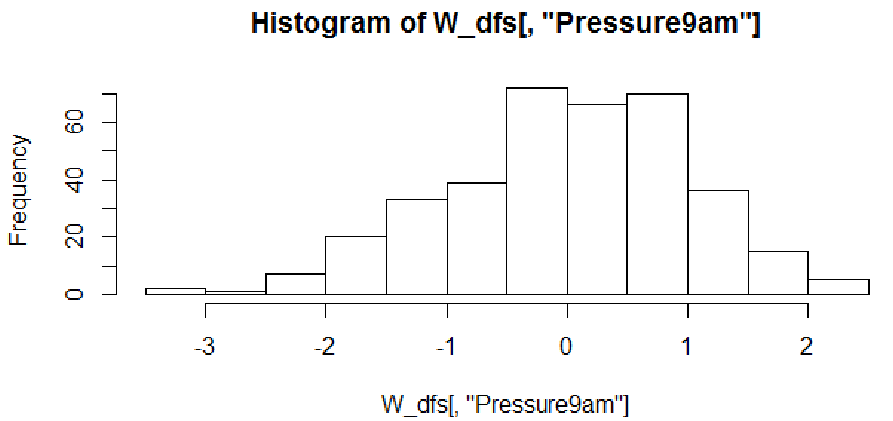
#Try: run the summary command, what do you notice above value ranges?

```
#Let do some normalization
#Make a function
myscale = function(x)
         if (class(x)=='integer' || class(x)=='numeric') {
                    (x-mean(x,na.rm=T)) / sd(x,na.rm=T) }
        else { x}

#get a new dataframe and replace with normalized
#       values
W_dfs = W_df
for (i in num_classes)
                   { W_dfs[,i]=myscale(W_dfs[,i])}
```

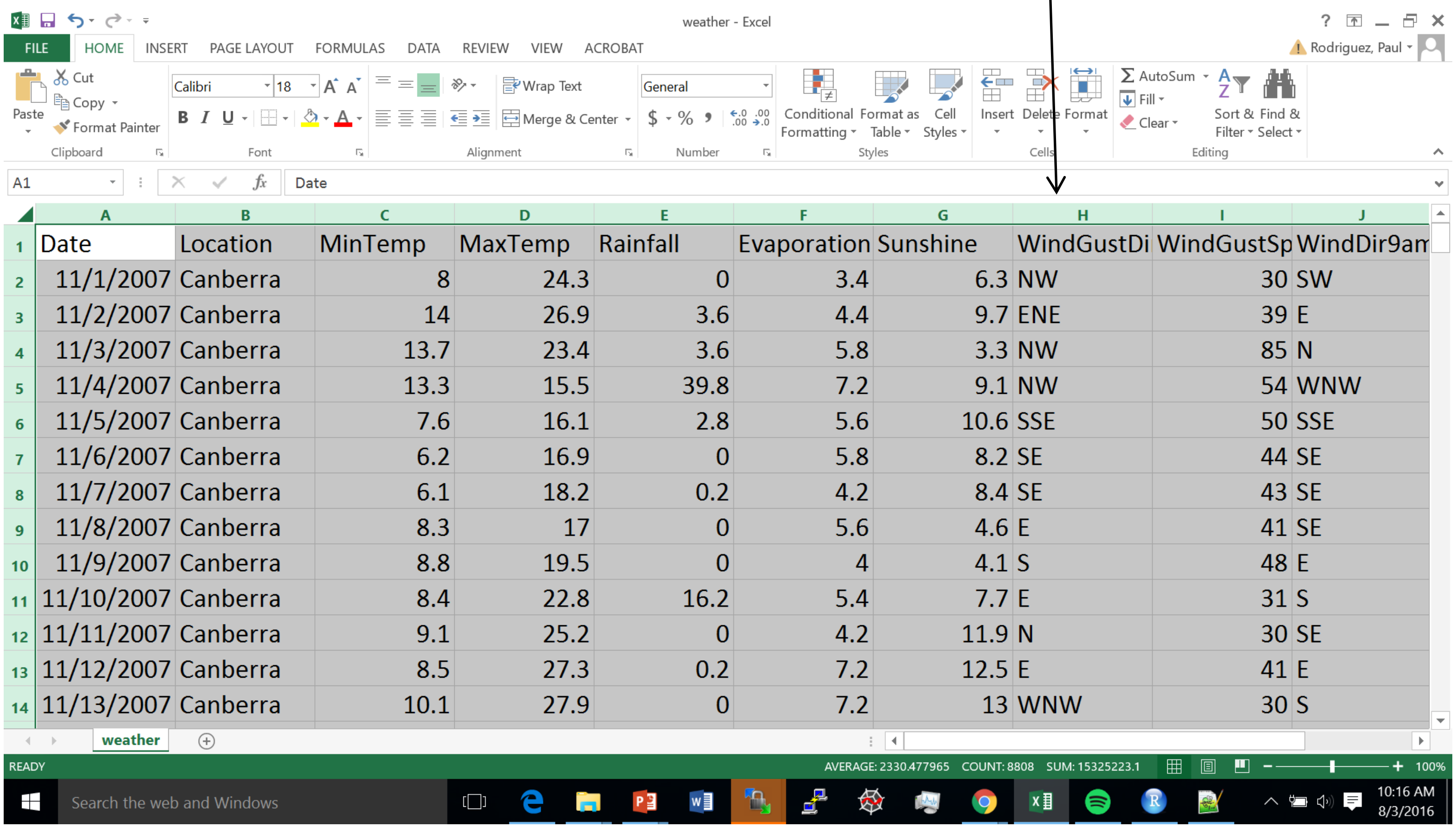
hist(W_df[,'Pressure9am'])





Transforming Weather Data Matrix

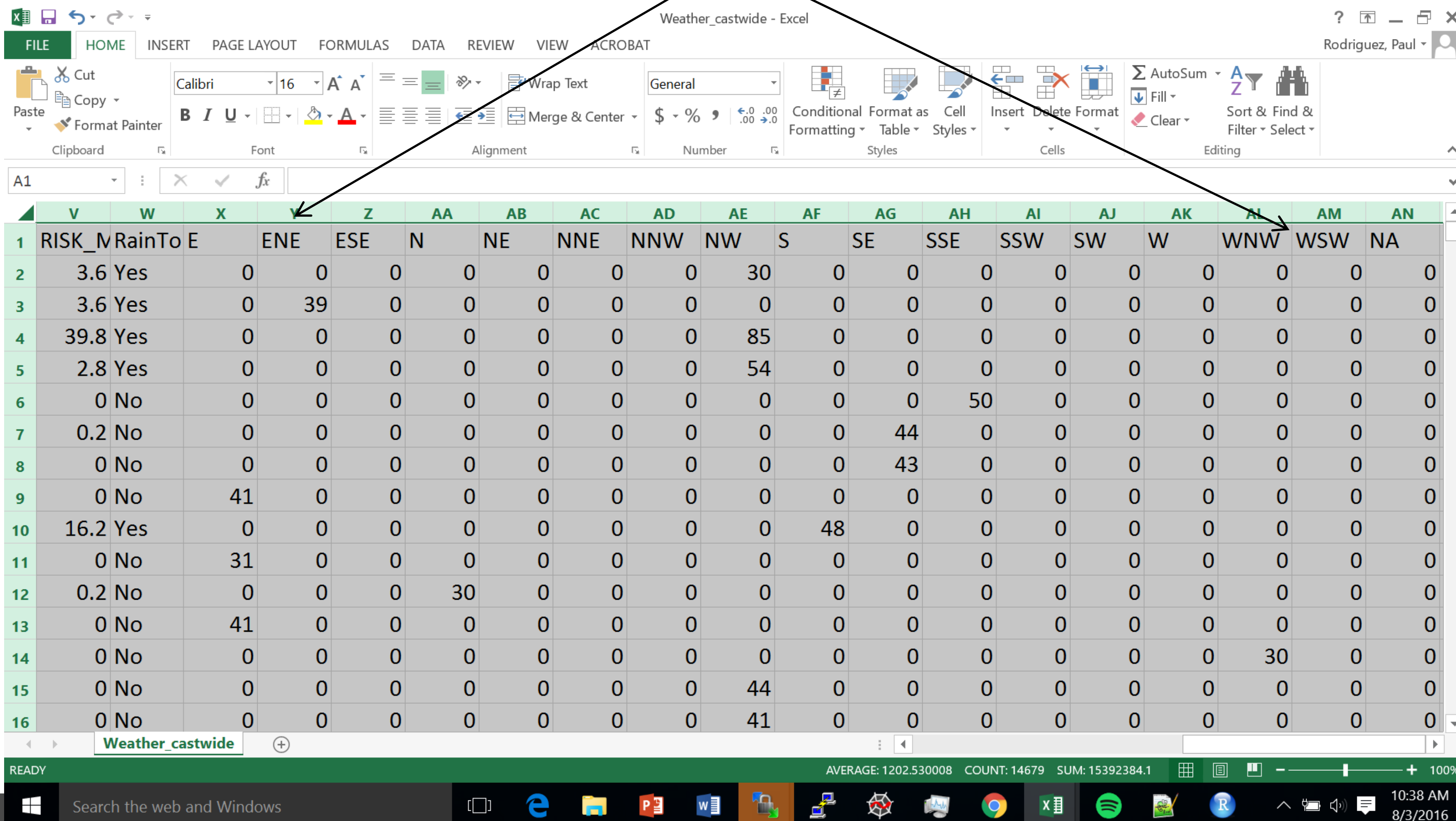
Let's consider WindGustDir as if it's a repeated measurement



	A	B	C	D	E	F	G	H	I	J
1	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDi	WindGustSp	WindDir9am
2	11/1/2007	Canberra	8	24.3	0	3.4	6.3	NW	30	SW
3	11/2/2007	Canberra	14	26.9	3.6	4.4	9.7	ENE	39	E
4	11/3/2007	Canberra	13.7	23.4	3.6	5.8	3.3	NW	85	N
5	11/4/2007	Canberra	13.3	15.5	39.8	7.2	9.1	NW	54	WNW
6	11/5/2007	Canberra	7.6	16.1	2.8	5.6	10.6	SSE	50	SSE
7	11/6/2007	Canberra	6.2	16.9	0	5.8	8.2	SE	44	SE
8	11/7/2007	Canberra	6.1	18.2	0.2	4.2	8.4	SE	43	SE
9	11/8/2007	Canberra	8.3	17	0	5.6	4.6	E	41	SE
10	11/9/2007	Canberra	8.8	19.5	0	4	4.1	S	48	E
11	11/10/2007	Canberra	8.4	22.8	16.2	5.4	7.7	E	31	S
12	11/11/2007	Canberra	9.1	25.2	0	4.2	11.9	N	30	SE
13	11/12/2007	Canberra	8.5	27.3	0.2	7.2	12.5	E	41	E
14	11/13/2007	Canberra	10.1	27.9	0	7.2	13	WNW	30	S

Transforming Weather Data Matrix

Now: WindGustDir values each have their own column



	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
1	RISK_M	RainTo	E	ENE	ESE	N	NE	NNE	NNW	NW	S	SE	SSE	SSW	SW	W	WNW	WSW	NA
2	3.6	Yes	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
3	3.6	Yes	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	39.8	Yes	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0
5	2.8	Yes	0	0	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0
6	0	No	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0
7	0.2	No	0	0	0	0	0	0	0	0	0	44	0	0	0	0	0	0	0
8	0	No	0	0	0	0	0	0	0	0	0	43	0	0	0	0	0	0	0
9	0	No	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	16.2	Yes	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0
11	0	No	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0.2	No	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	No	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	No	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0
15	0	No	0	0	0	0	0	0	0	44	0	0	0	0	0	0	0	0	0
16	0	No	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0

```
install.packages('reshape2')
library(reshape2)

# long-to-wide: 'cast' repeated measure into wide table
W_cast   =dcast(W_df,
      formula=Date+Location+ ...~ WindGustDir,
      fill=0,
      value.var="WindGustSpeed")
```

WindGustDir distinguished the repeated measures

WindGustSpeed has the actual values

Extra to try:

```
# wide to long: ie 'melt' repeated measure into long
#          table
W_melt   =melt(W_cast,
         na.rm=TRUE,
         measure.vars=c(23:39),
         variable.name="WindGustDir_cast")
```

The repeated measures are now in columns 23 to 30

pause

Reading Material

- **Data Preparation for Data Mining by Dorian Pyle**
  - http://www.ebook3000.com/Data-Preparation-for-Data-Mining_88909.html
- **Data mining – Practical Machine learning tools and techniques by Witten & Frank**
  - http://books.google.com

Many Variables

- **More variables => more information, but also more noise and more ways of interactions**

- **2 ways to handle many variables**
  - Variable Selection
  - Dimension reduction methods

Variable selection vs Dimensionality Reduction

- **Prior to algorithm, depends on data**
  - For large P, with noise particular to variables, try variable selection
  - For large P, diffuse noise, try dimension reduction by Matrix Factorization

Matrix Factorization:

Given a numeric matrix, can we reduce the number of columns?

conversely

Can we find interesting subspaces?

- **Matrix Factorization:**

Given a numeric matrix, can we reduce the number of columns?

  - Yes, if features are constant or redundant

- **Matrix Factorization:**

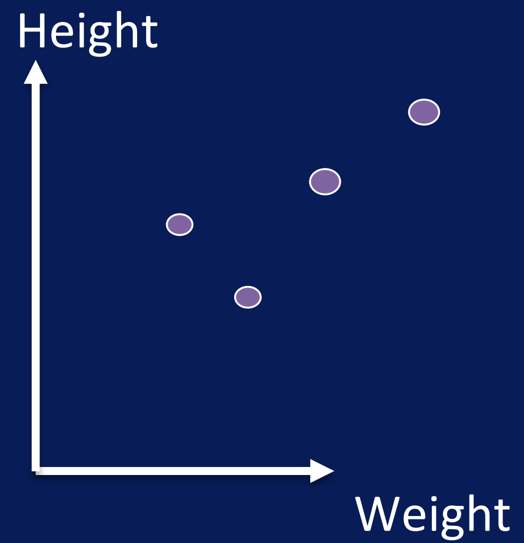
Given a numeric matrix, can we reduce the number of columns?

- Yes, if features are constant or redundant
- Yes, if features only contribute noise (conversely, want features that contribute to variations of the data)

Example: 2D data

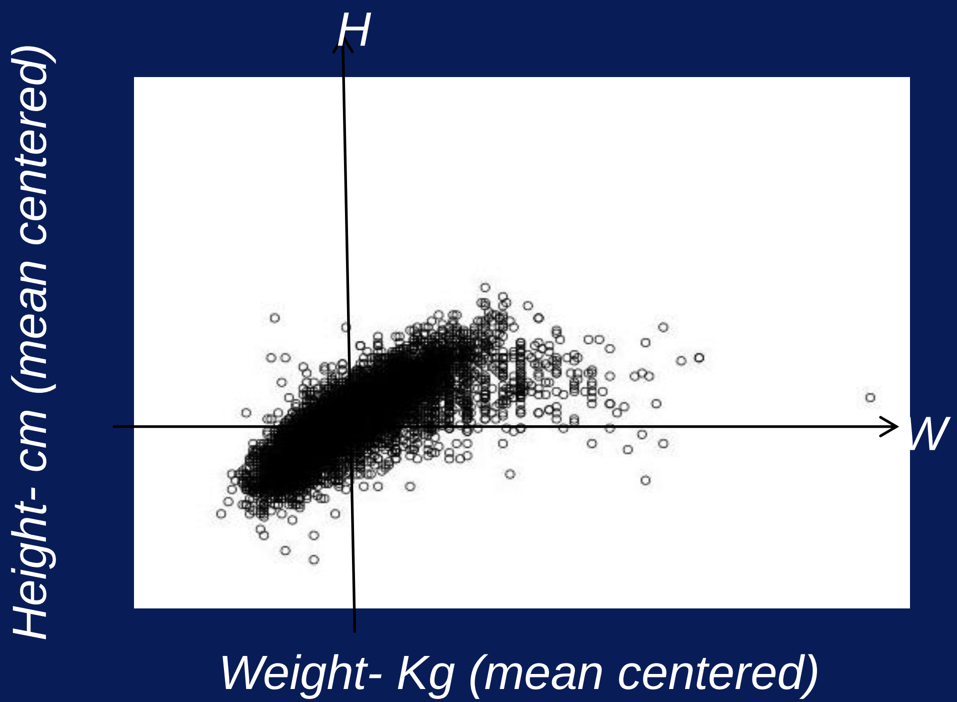
	Weight	Height
S1	50	179
S2	66	175
S3	74	180
S4	94	192

PLOT



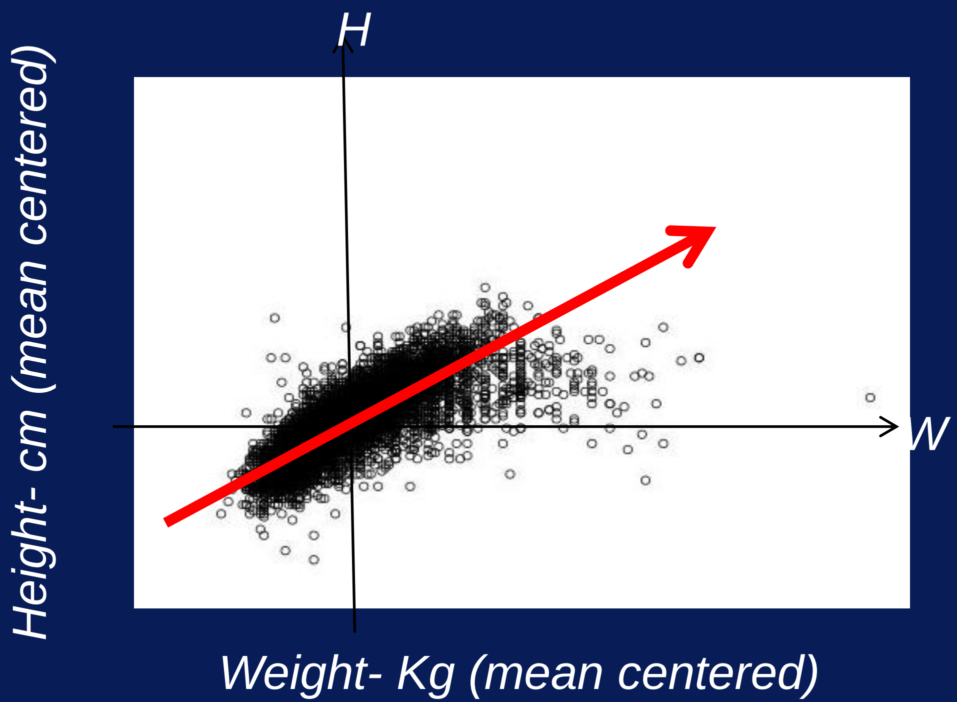
this is the input space

Example: Athletes' Height by Weight

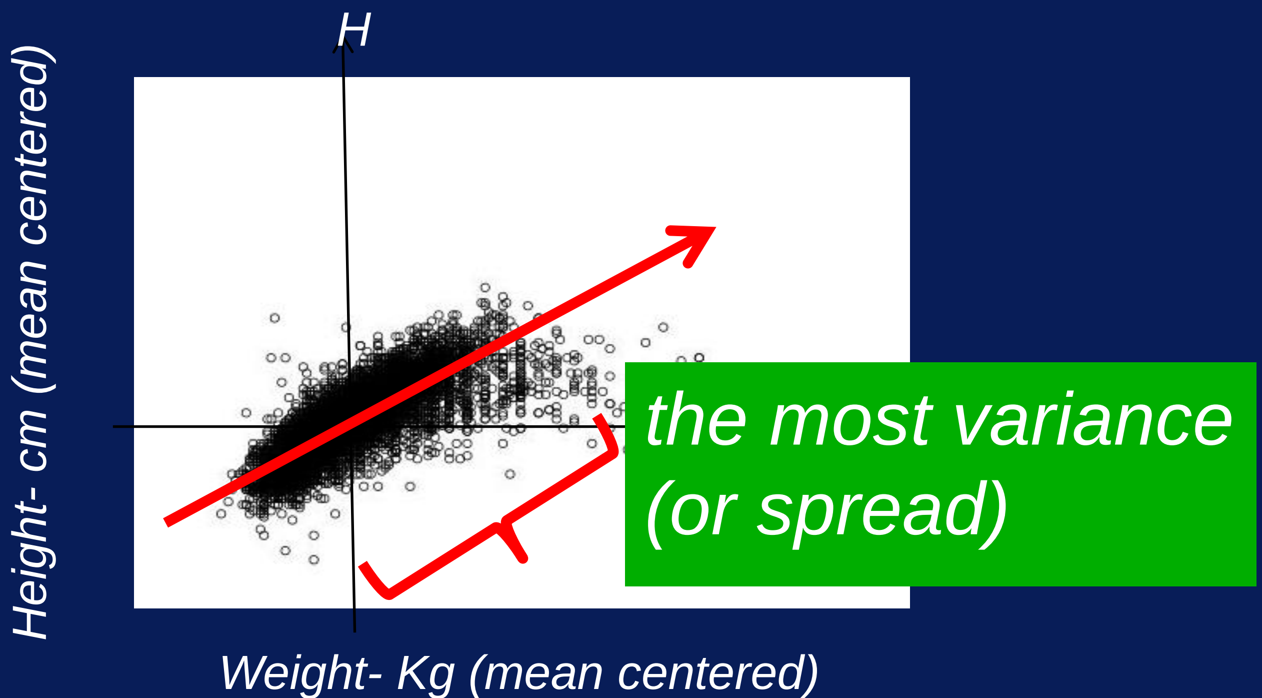


Find a line that aligns with the data.

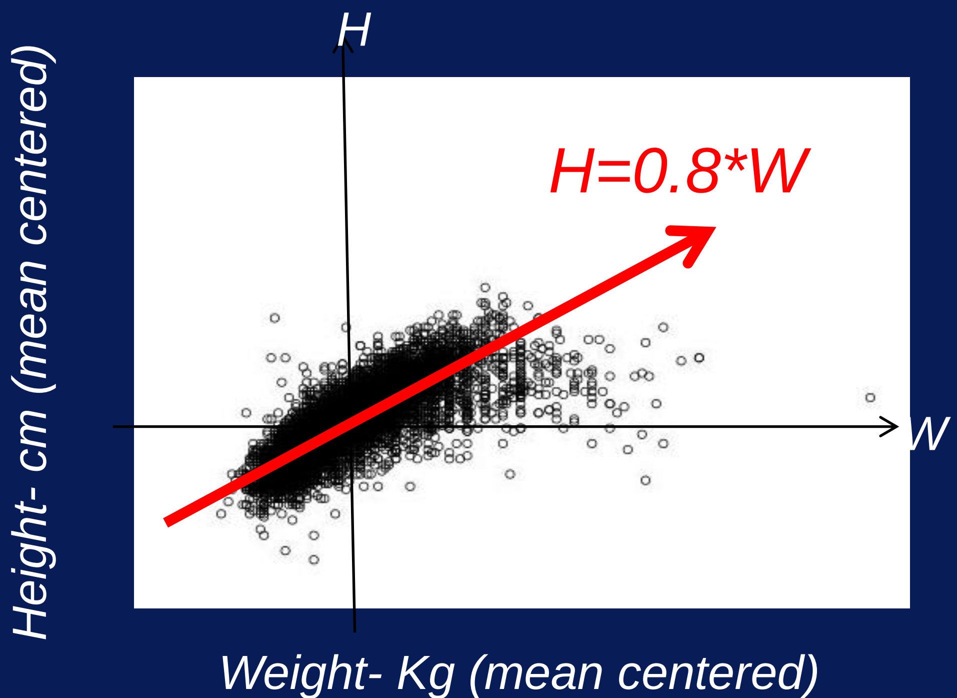
Example: Athletes' Height by Weight



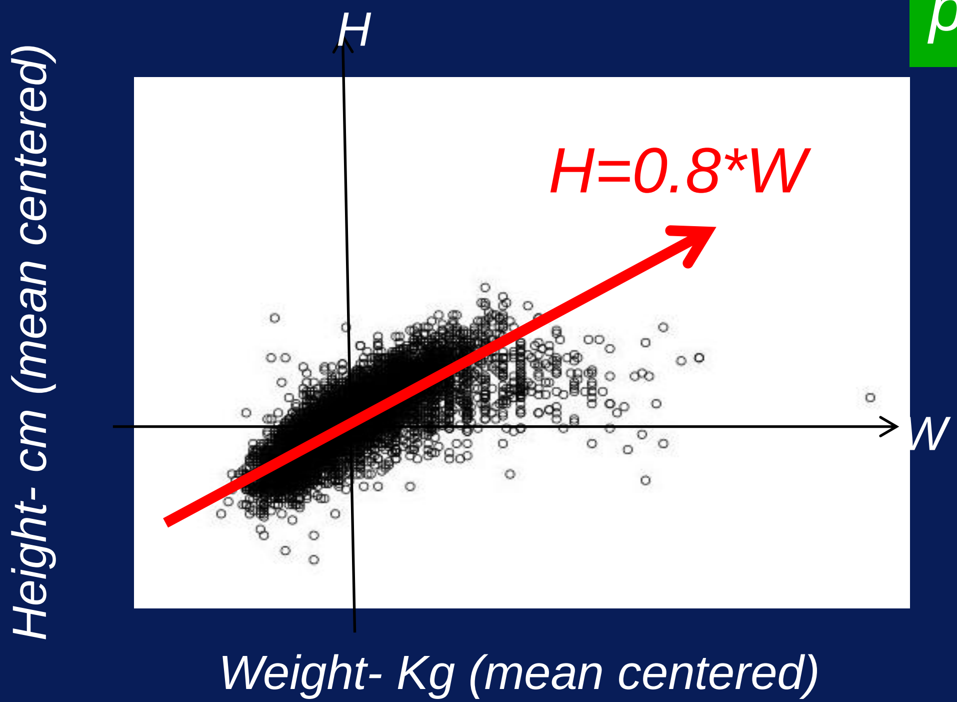
Find a line that aligns with the data.



Find a line that aligns with the data.



Find a line that aligns with the data.

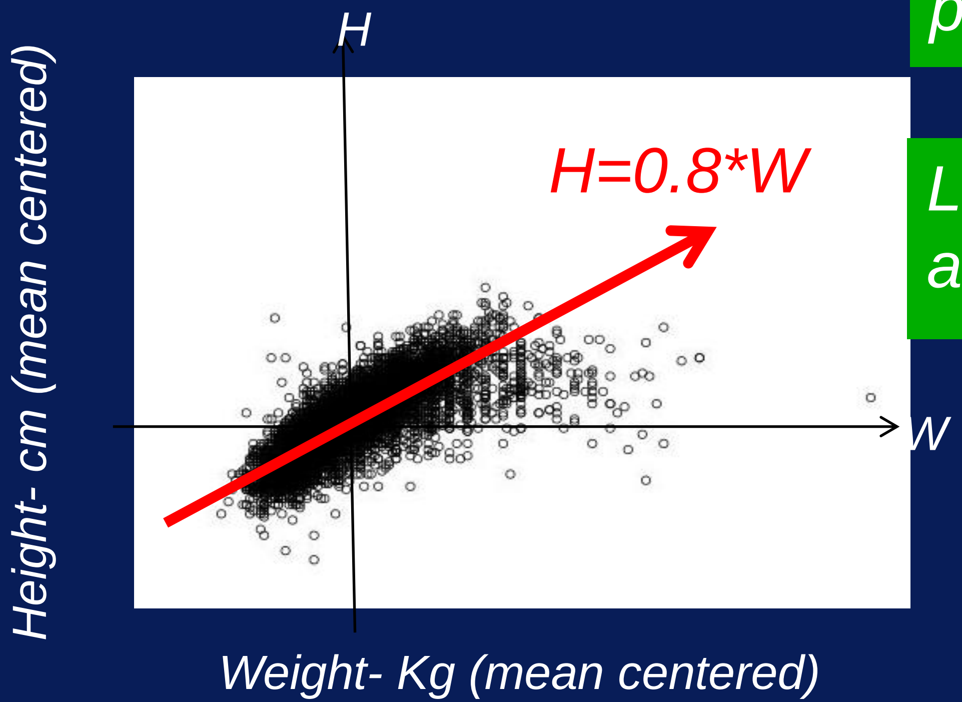


Note that W=1,H=0.8 is a point on the line, for example.

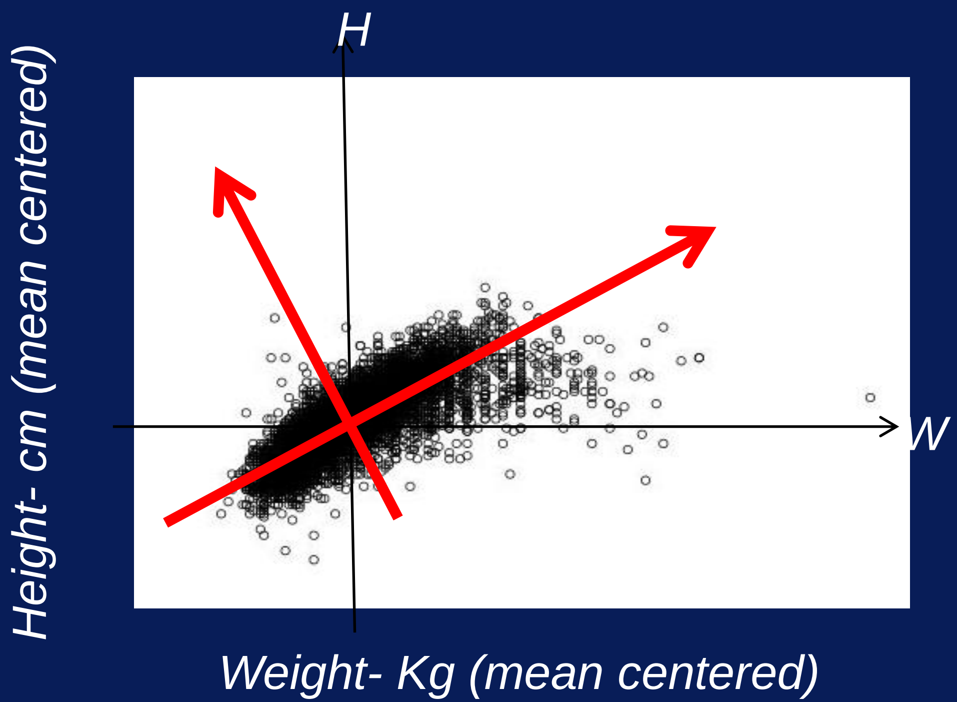
Find a line that aligns with the data.

Note that W=1,H=0.8 is a point on the line, for example.

Let [1 0.8] represent the line, as a combination of W & H.

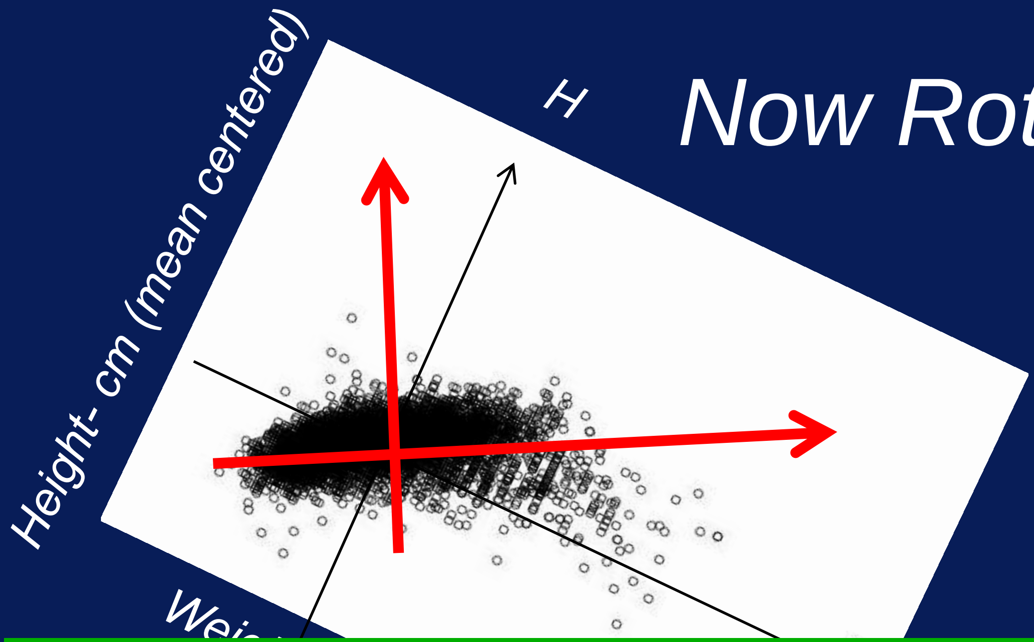


Find a line that aligns with the data.

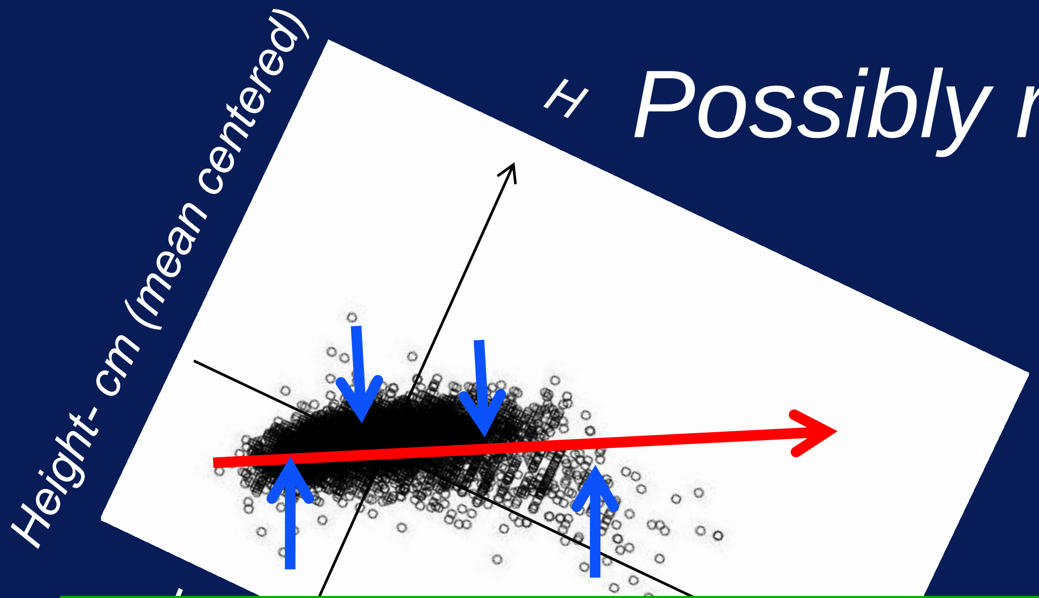


The next direction of most variance.

Now Rotate Axis



New axis (AKA features) defined as combinations of old features



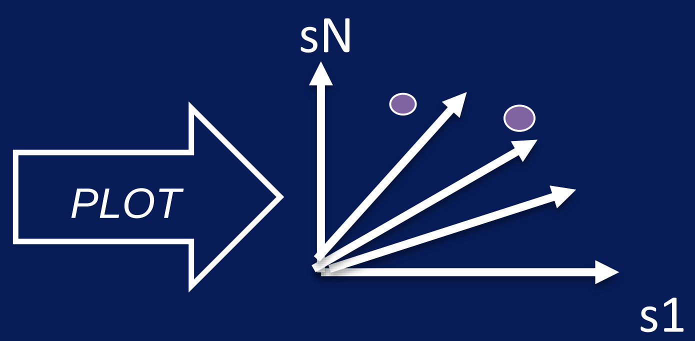
Possibly reduce dimensions

Project all points to one axis

(defined by the [1 0.8] 2D vector)

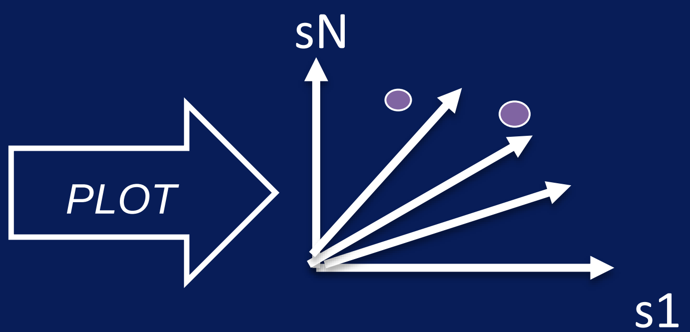
2D data transposed to 2 points in high dimensional space

	s1	s2	s3	…
Height	179	175	180	…
Weight	50	66	74	…



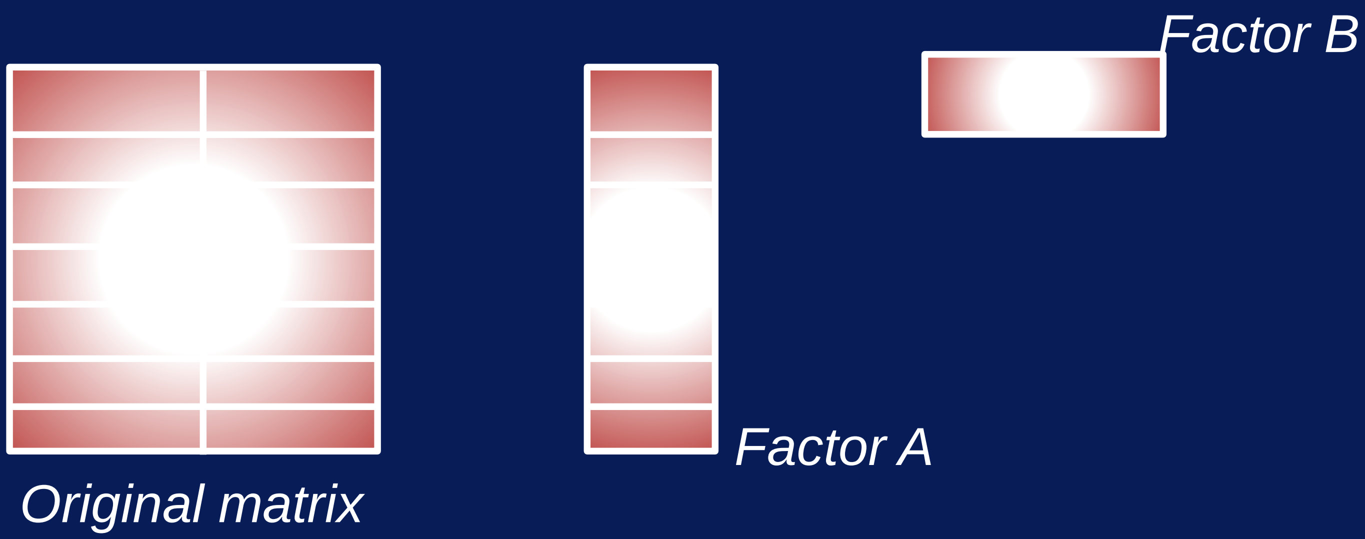
2D data transposed to 2 points in high dimensional space

	s1	s2	s3	…
Height	179	175	180	…
Weight	50	66	74	…

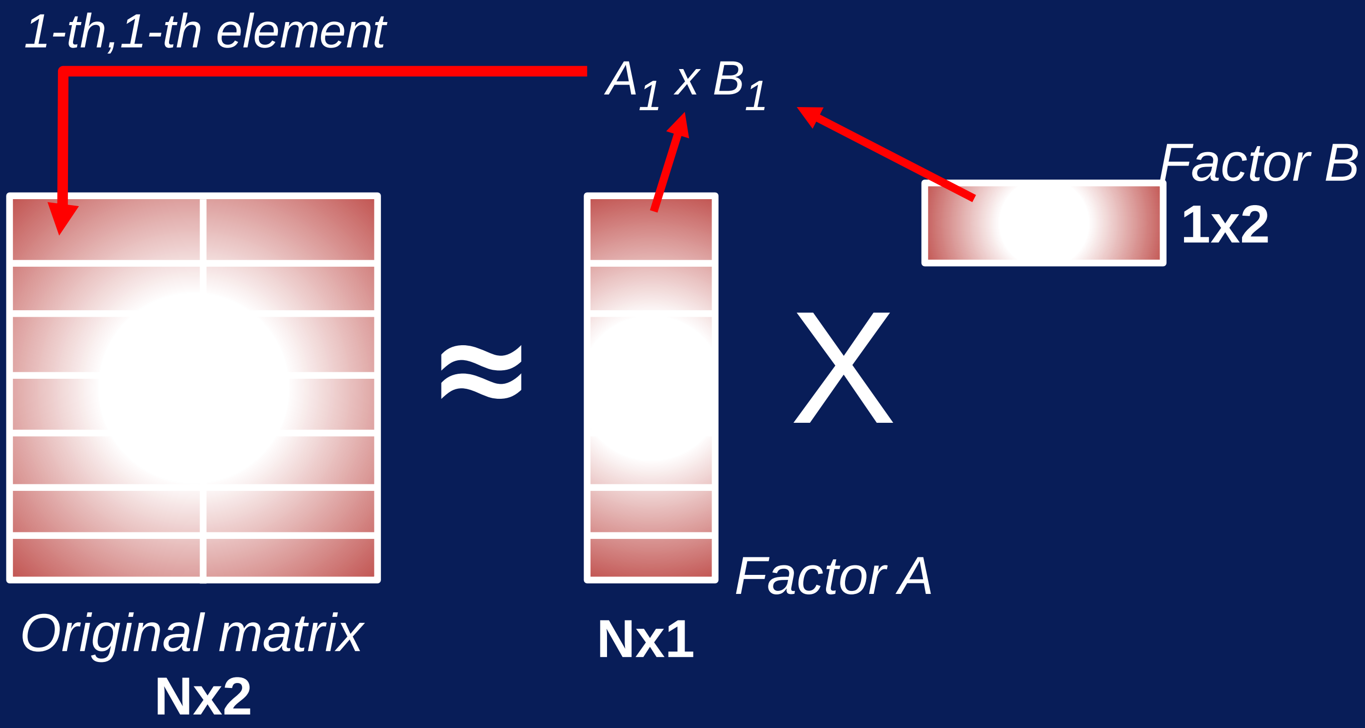


Same process as before, but now factors are N-dimensional vectors

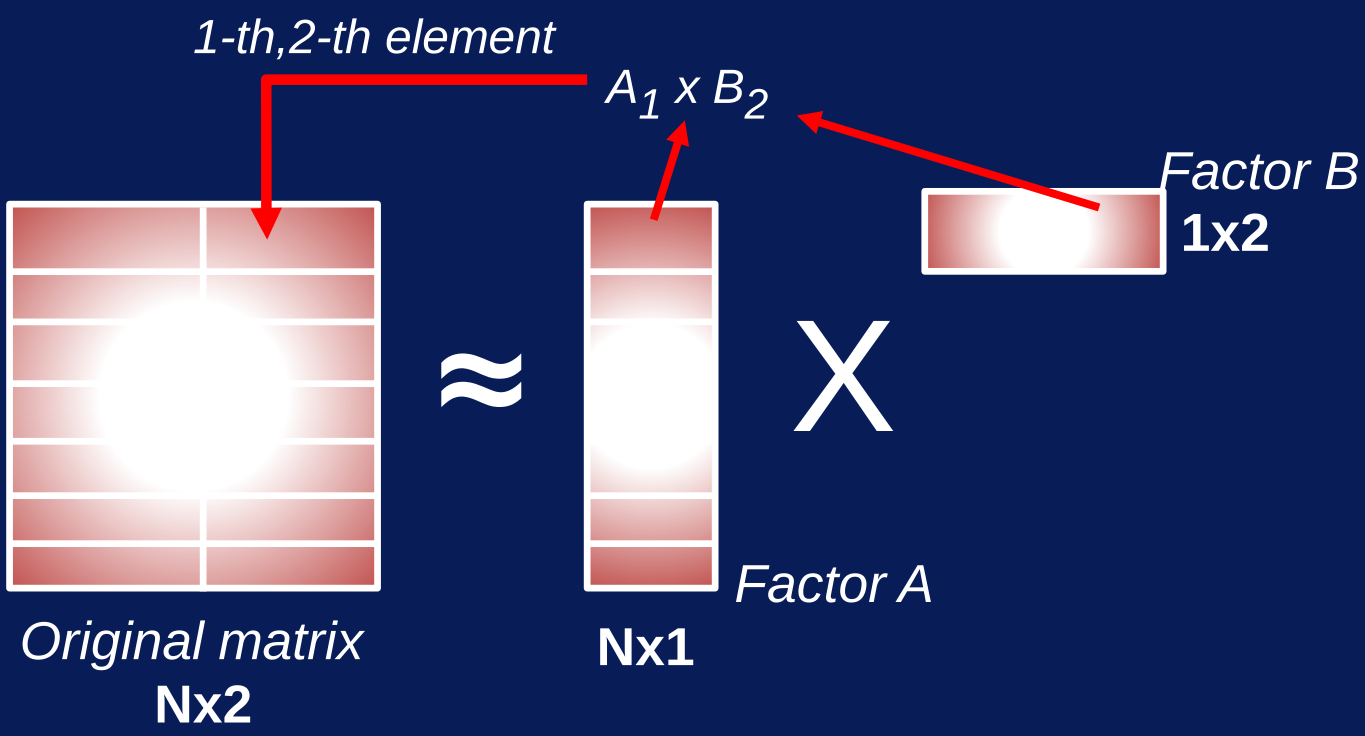
Any Matrix can be approximated by outer product of factors



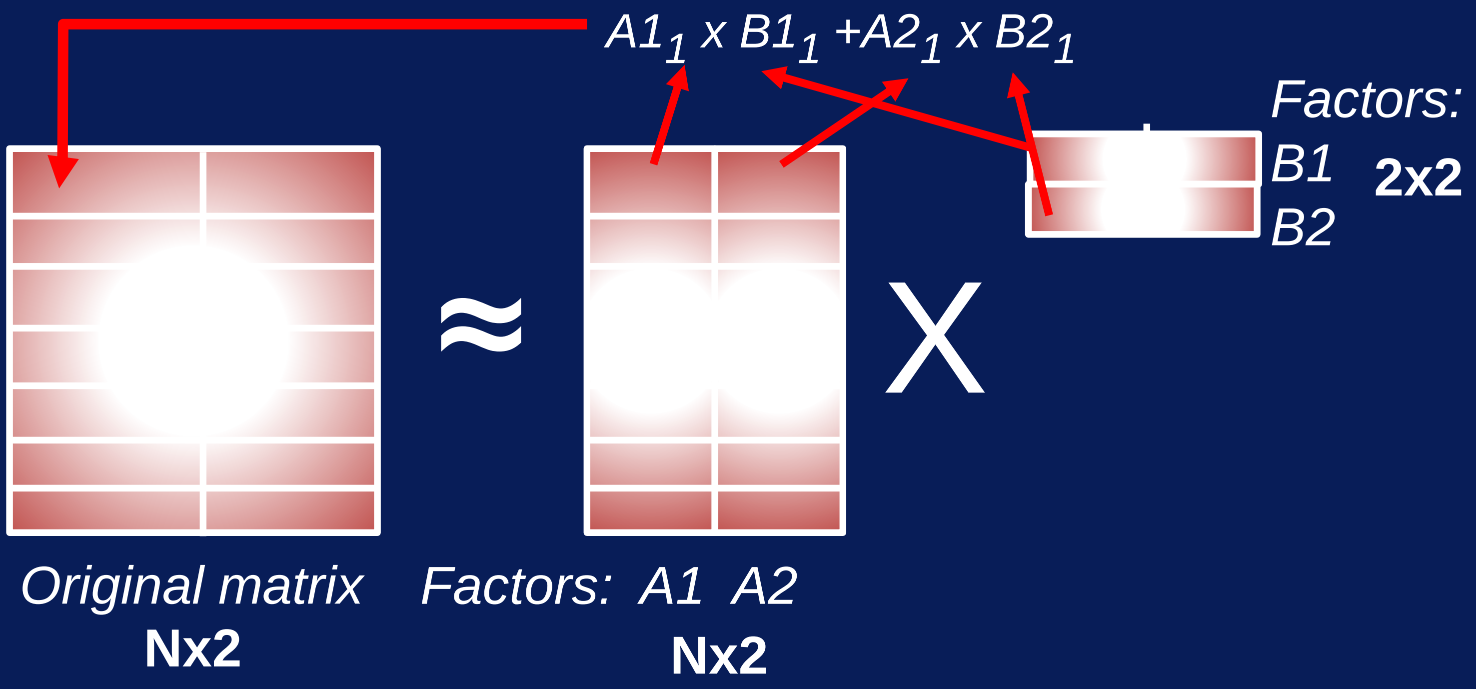
Any Matrix can be approximated by outer product of factors



Any Matrix can be approximated by outer product of factors



More factors gives better approximation



- Best Known Factorization Algorithms:
    - SVD (singular value decomposition)
    - PCA (principle component analysis)

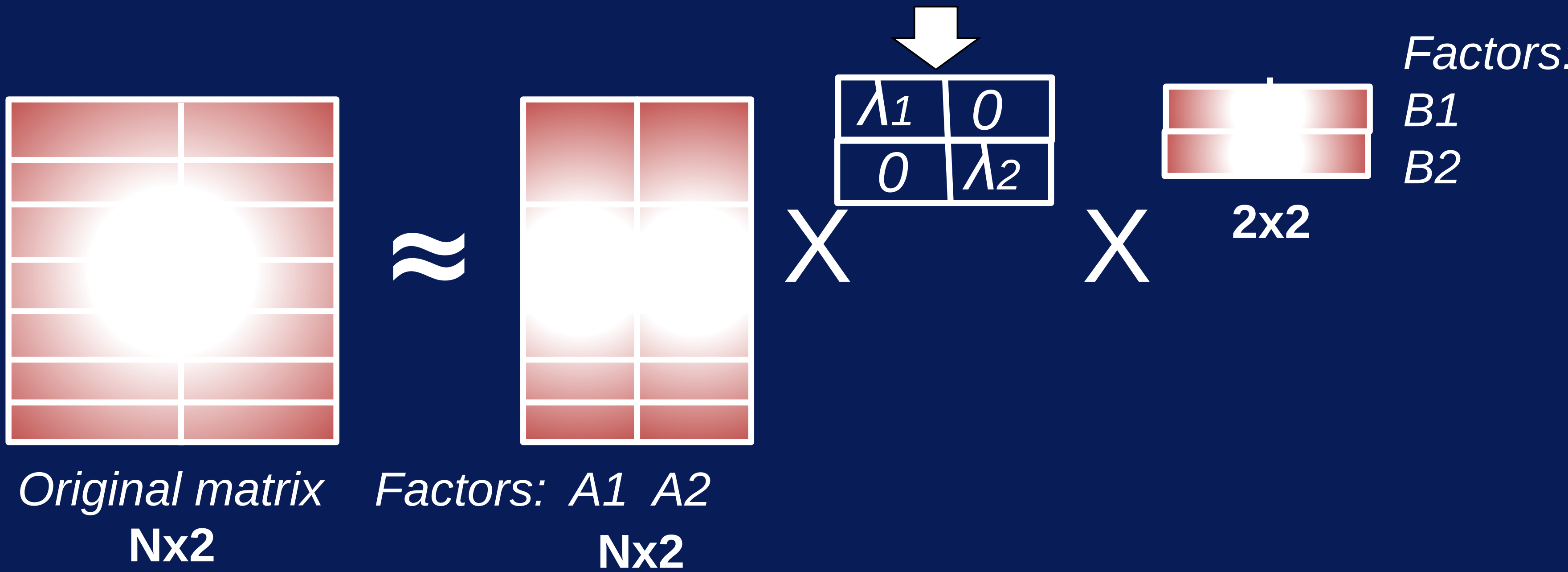
- Best Known Factorization Algorithms:

  SVD (singular value decomposition)

  PCA (principle component analysis)

Find orthogonal factors and
scale them down (i.e. normalize)

SVD: factors and 'singular' scale values



- More generally:

Factorization Algorithms may vary depending on criterion for how factors ‘align’ with data.

- More generally:

Factorization Algorithms may vary depending on criterion for how factors ‘align’ with data.

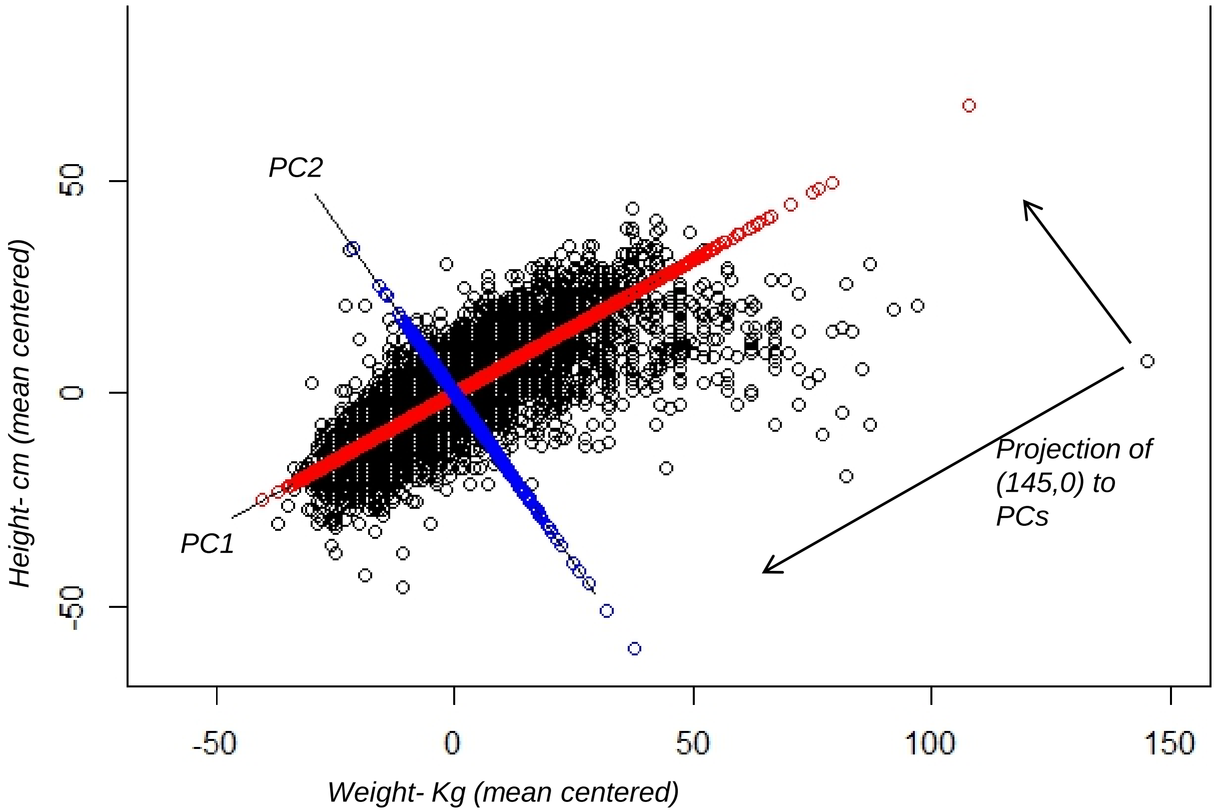
- Number of factors to use depends on tradeoff of good approximation vs good dimensional reduction

Can use cross validation or heuristics to choose.

Dimensionality Reduction via Principle Components

- **PCA:**
  - Find set of *k* vectors (aka factors) that describe data in alternative way
  - First component is the vector that maximizes the variance of data projected onto that vector
  - *K*-th component is orthogonal to all *k*-1 previous components

PCA conserves and reorders variance



Total Variance Conserved:

Var in Weight
\+
Var in Height
=
Var in PC1 +
Var in PC2

In general:

Var in PC1>
Var in PC2>
Var in PC3…

Principle Components

- **Can choose *k* heuristically as approximation improves, or choose *k* so that 95% of data variance accounted**
- **aka Singular Value Decomposition**

  PCA on square matrices only

  SVD gives same vectors on square matrices
- **Works for numeric data only**
- **For higher dimensional data, use factors to visualize 2 factors at a time**

Later, we'll compare SVD components with Clustering

```
#W_num is only numeric or integer fields of Weather data
> Wsvd=svd(W_num)

> str(Wsvd)
List of 3
 $ d: num [1:9] 27442.7 231.2 96.4 68.2 44.5 ...
 $ u: num [1:363, 1:9] -0.0524 -0.0521 -0.052 -0.0519 -0.0525 ...
 $ v: num [1:9, 1:9] -0.005042 -0.014276 -0.000969 -0.00314 -0.005491 ...
```

- **end**