

# Data Preparation

Paul Rodriguez



# Overview

- **Highlights of data prep**
- **Variable Selection and Reduction**

# The Importance of Data Prep

- **“Garbage in, garbage out”**
- **Sometimes takes 60-80% of the whole data mining effort**

# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data

# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data
  - Filtering – select subsets or summaries

# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data
  - Filtering – select subsets or summaries
  - Transforming – normalize or enhance

# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data
  - Filtering – select subsets or summaries
  - Transforming – normalize or enhance
  - Organizing data - aka ‘data wrangling’ or ‘data munging’

# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data
  - Filtering – select subsets or summaries
  - Transforming – normalize or enhance
  - Organizing data - aka ‘data wrangling’ or ‘data munging’
  - Variable Selection/Dimension Reduction



# Working definition

- **Data Preparation:**
  - Cleaning – deal with noise, outliers, missing data
  - Filtering – select subsets or summaries
  - Transforming – normalize or enhance
  - Organizing data - aka ‘data wrangling’ or ‘data munging’
  - Variable Selection/Dimension Reduction

***In a nutshell, prepare data for modeling***

# Missing Data (NA vs NULL)

- **Not applicable - NA**

e.g. spouse name depends on marital status

- **Not available - NULL**

unknown

not entered

- **In R:**

NA is logically missing (ie the value won't be available): `is.na()`

NULL is object not yet defined (ie the object is not available): `is.null()`

# Missing Data

- **What are frequency counts of missing variables?**

Are entries missing completely at random or contingent on some other variable?

# Quick Approaches

- Delete instances  
and/or
- Delete attributes with high missing-ness

# Simple Imputation

- Use the attribute mean
- Use the attribute mean for each class label

# 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

- ‘Amelia’ package example
  - 50 attributes from UN voting data
  - 1K-100K entries missing per col for about 20 cols
  - 300K rows ~ 1 hour on compute node (not run on the user’s PC)

```
# 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")
```

*time variable for time  
series models*

*cross section to select  
temporal periods*

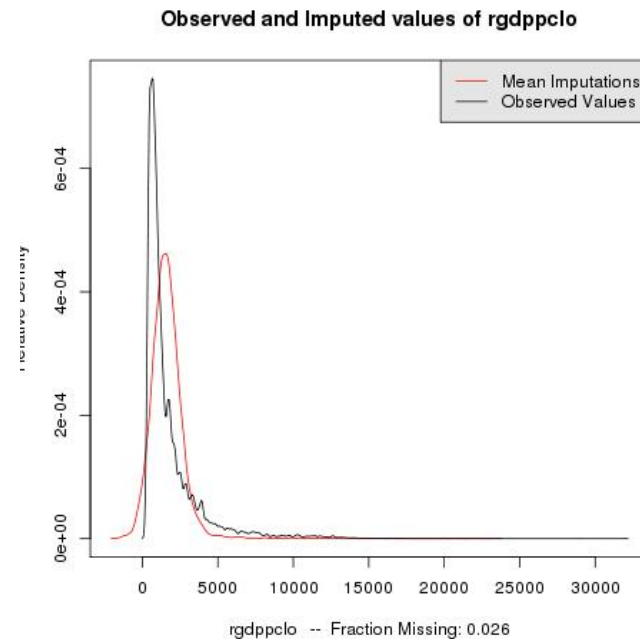
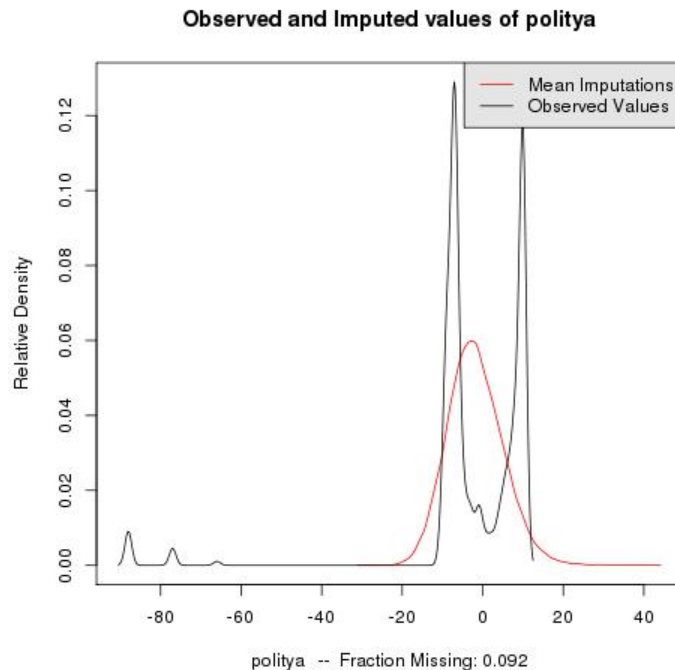
*ignore 'id' variables*

*interactions*

*parallel options*

# R and missing data

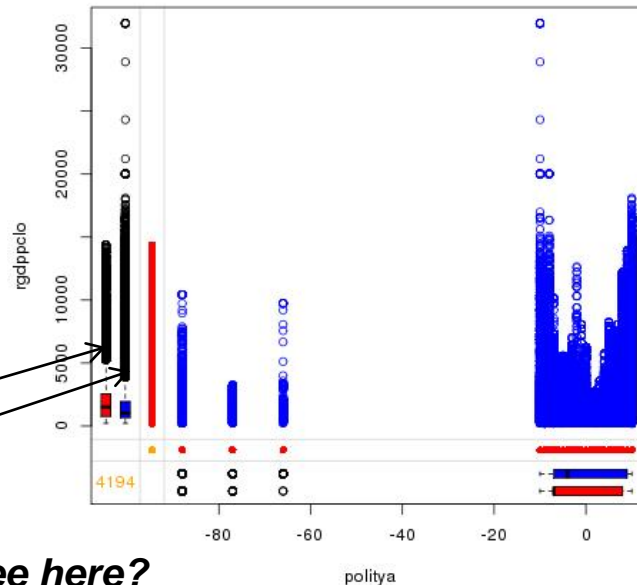
#QA on missing data by comparing density of imputed & original data  
`compare.density(a.out, var="politya")`  
`compare.density(a.out, var='rgdpcontg')`



# 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**
- **Discretize data**  
(perhaps more intuitive to deal with binned data)

# Feature Engineering is Variable Enhancement

- Use Domain and world knowledge
- Example: given date and location of doctor visits
  - a new variable for Number-of-1<sup>st</sup>-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**  $x_{new} = x - \text{mean}(x)$
- **z-score**  $score = \frac{x - \text{mean}(x)}{\text{std}(x)}$
- **Scale to [0...1]**  $x_{new} = \frac{x - \min(x)}{\max(x) - \min(x)}$
- **log scaling**  $x_{new} = \log(x)$

# Generally

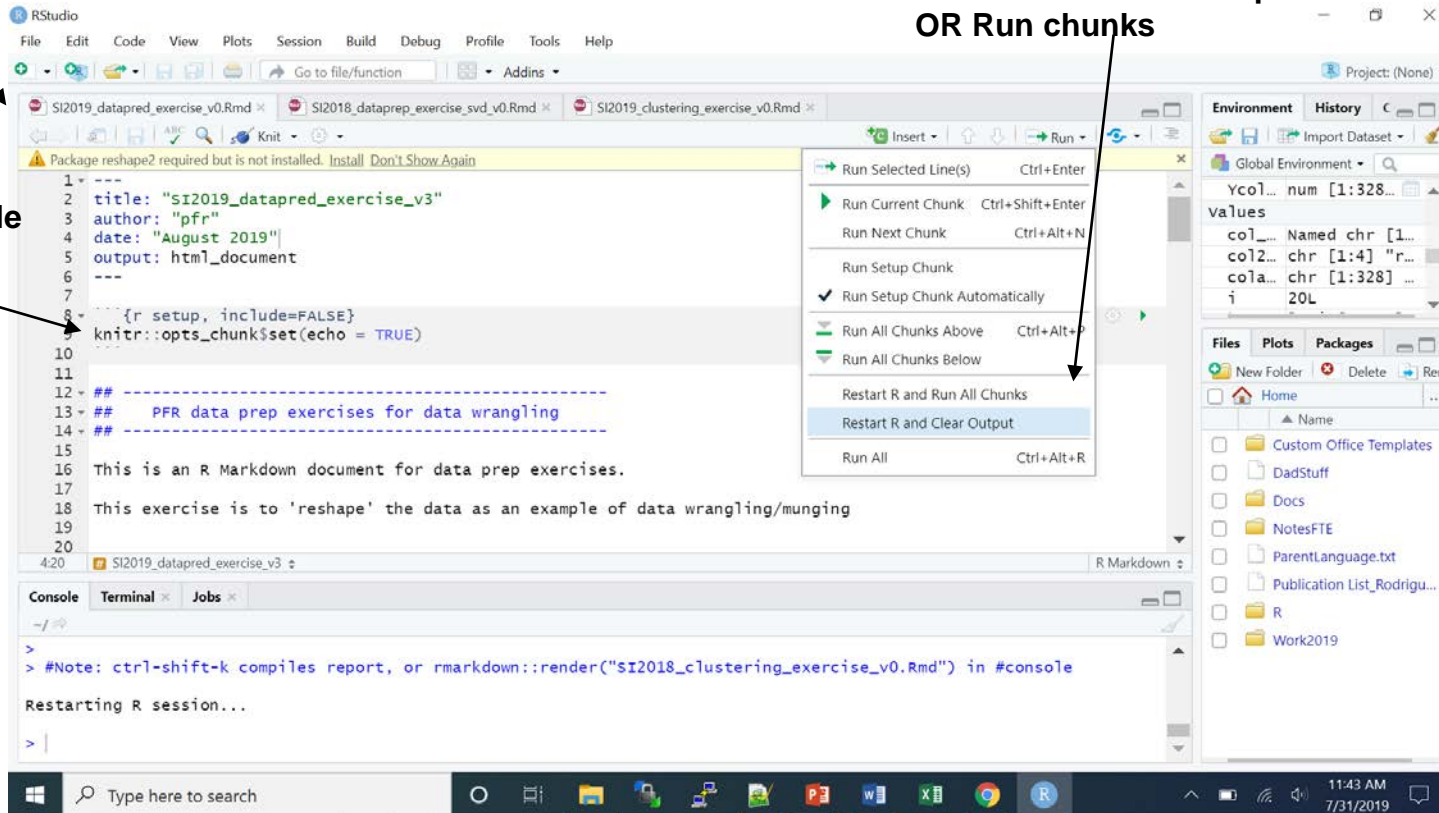
- **Preparing data is based on statistical principles,**
- **But also heuristics**

# Data Wrangling Exercise

Open File -> to  
load R mark-  
down exercises  
("\_.Rmd")

R code  
chunks  
inside triple  
backticks

Select Run drop down->  
restart R and clear output  
OR Run chunks





Read file into  
dataframe and  
show the first  
few (head) rows

Run code chunk  
above

Run current  
chunk

The screenshot shows the RStudio interface with a code editor, environment pane, and console. The code in the editor reads a CSV file and displays the first six rows of the resulting data frame.

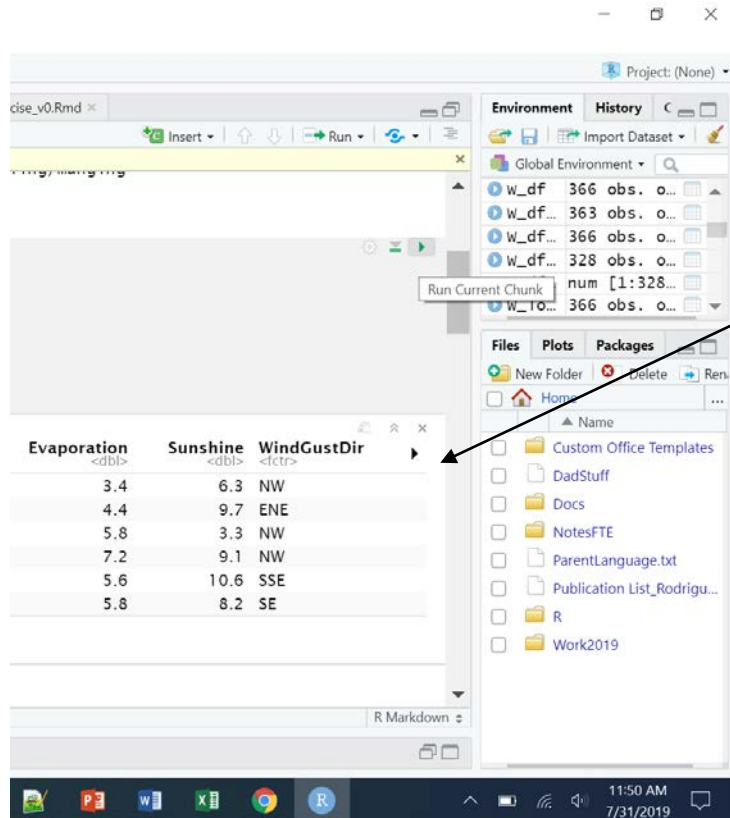
```
19  
20  
21- ##load data  
22- {r load data}  
23- #use setwd("c:/your-path-to-your-project/data/")  
24  
25- w_df = read.table('weather_orig.csv',  
26-                   header=TRUE, sep="," ,  
27-                   stringsAsFactors = TRUE) #try TRUE  
28- head(w_df)  
29-  
30-  
31  
32- ##reshape data  
23:49 [1] Chunk 2: load data
```

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir
1 2007-11-01	Canberra	8.0	24.3	0.0	3.4	6.3	NW
2 2007-11-02	Canberra	14.0	26.9	3.6	4.4	9.7	ENE
3 2007-11-03	Canberra	13.7	23.4	3.6	5.8	3.3	NW
4 2007-11-04	Canberra	13.3	15.5	39.8	7.2	9.1	NW
5 2007-11-05	Canberra	7.6	16.1	2.8	5.6	10.6	SSE
6 2007-11-06	Canberra	6.2	16.9	0.0	5.8	8.2	SE

6 rows | 1-9 of 24 columns

Notice  
“WindGustDir”  
describes a  
category.  
Scroll to next  
column to see  
its value.

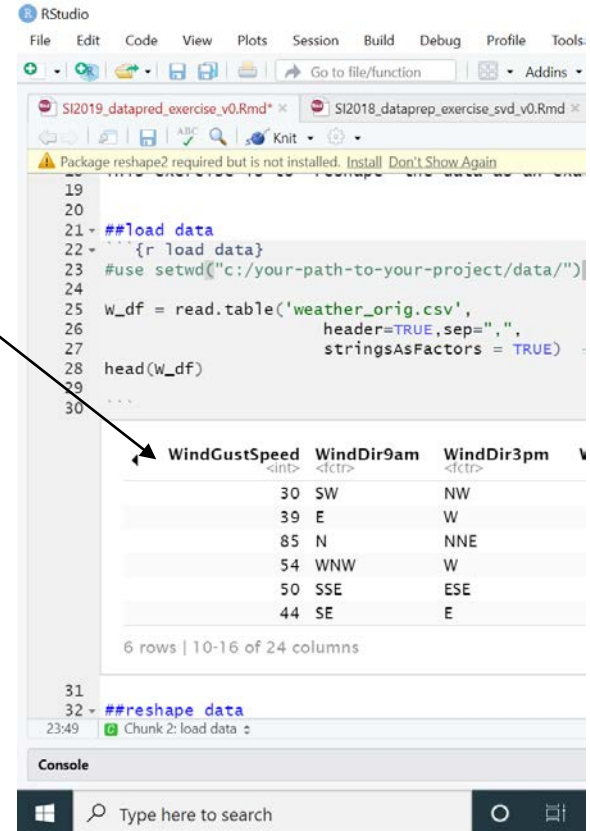
***Let's consider WindGustDir & WindGustSpeed as a repeated measurement. Assume we want code these as binary factors for a linear model, i.e. we want to put each direction in its own column.***



The RStudio Environment pane shows a data frame with the following columns: Evaporation (dbl), Sunshine (dbl), and WindGustDir (fctr). The data is as follows:

Evaporation	Sunshine	WindGustDir
3.4	6.3	NW
4.4	9.7	ENE
5.8	3.3	NW
7.2	9.1	NW
5.6	10.6	SSE
5.8	8.2	SE

Scroll to next column to see its value.

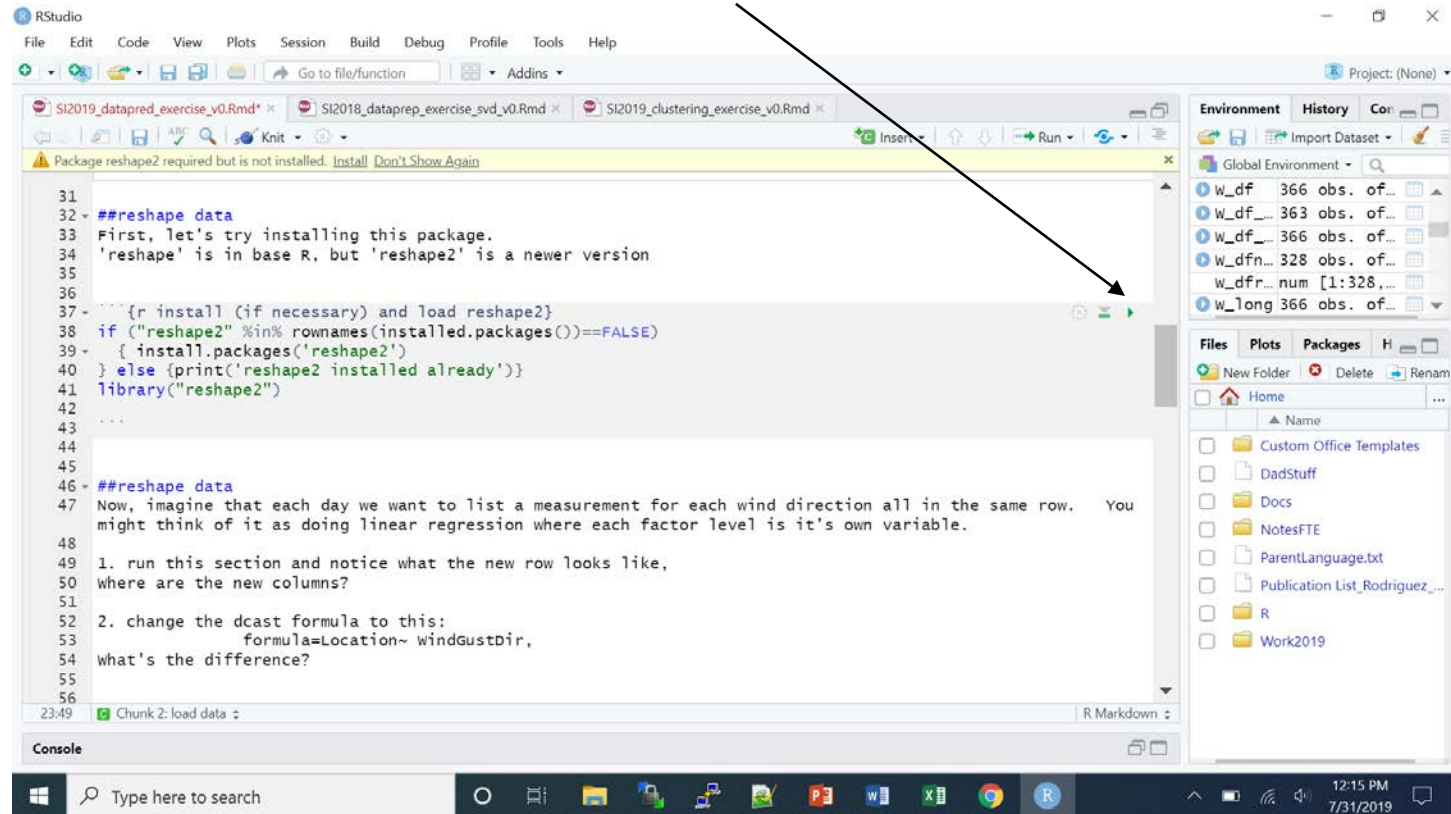


```
19  
20  
21- ##load data  
22- {r load data}  
23- #use setwd("c:/your-path-to-your-project/data/")  
24  
25- W_df = read.table('weather_orig.csv',  
26-                   header=TRUE, sep="," ,  
27-                   stringsAsFactors = TRUE)  
28- head(W_df)  
29  
30
```

The RStudio Code pane shows the following R code to load and view data. The output shows the first 6 rows of the data frame, with columns WindGustSpeed, WindDir9am, and WindDir3pm.

WindGustSpeed	WindDir9am	WindDir3pm
30	SW	NW
39	E	W
85	N	NNE
54	WNW	W
50	SSE	ESE
44	SE	E

Run this chunk to install  
“reshape2” package (if it is not  
already installed) and load it into  
this R session



# Long to Wide transform

	A	B	C	D	E	F	G	H	I	J
1	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDi	WindGustSp	WindDir9arr
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

*date, location and the rest identify the row*

*WindGustDir entries are labels for the repeated measures*

```
59
60 # long to wide: ie 'cast' repeated measure into wide table
61 w_long = dcast(w_df,
62               formula = Date + Location + ... ~ WindGustDir,
63                   #date, location and the rest are not repeated
64                   #WindGustDir entries are labels for the repeated measures
65                   fill = 0, #this could be 0 or NA, for example.
66                   value.var = "windGustSpeed")
67                   #this variable has the repeated measurement values
68
```

*this could be 0 or NA*

*indicate variable that has the repeated measurement values*

# Transformed Data Matrix

***Now: WindGustDir values each have their own column. These are all new columns AFTER the last column of original data frame. Only one value in each row is non-zero.***

	RainToday <fctr>	RISK_MM <dbl>	RainTomorrow <fctr>	E <dbl>	ENE <dbl>	ESE <dbl>	N <dbl>	NE <dbl>	NNE <dbl>	NNW <dbl>
	No	3.6	Yes	0	0	0	0	0	0	0
	Yes	3.6	Yes	0	39	0	0	0	0	0
	Yes	39.8	Yes	0	0	0	0	0	0	0
	Yes	2.8	Yes	0	0	0	0	0	0	0
	Yes	0.0	No	0	0	0	0	0	0	0
	No	0.2	No	0	0	0	0	0	0	0

## 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](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  $\Rightarrow$  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 and number of variables ( $P$ )**
  - For large  $P$ , with noise particular to variables, try variable selection
  - For large  $P$ , diffuse noise, try dimension reduction by Matrix Factorization

# 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 small error effect in the model  
when removed or permuted

# Matrix Factorization:

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

# 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

# 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

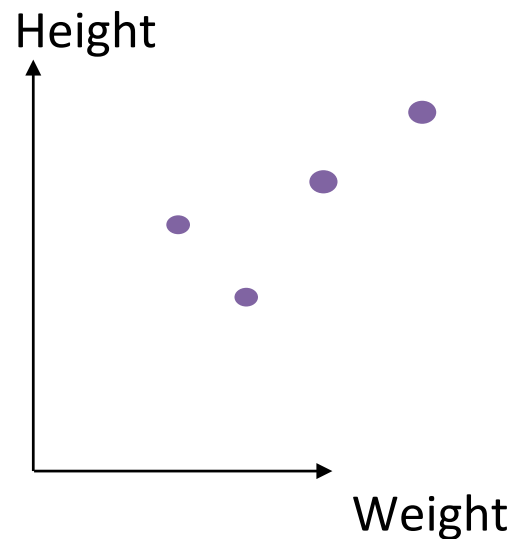
Conversely, want features that contribute to variations of the data

# Think graphically!

## An example in 2D:

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

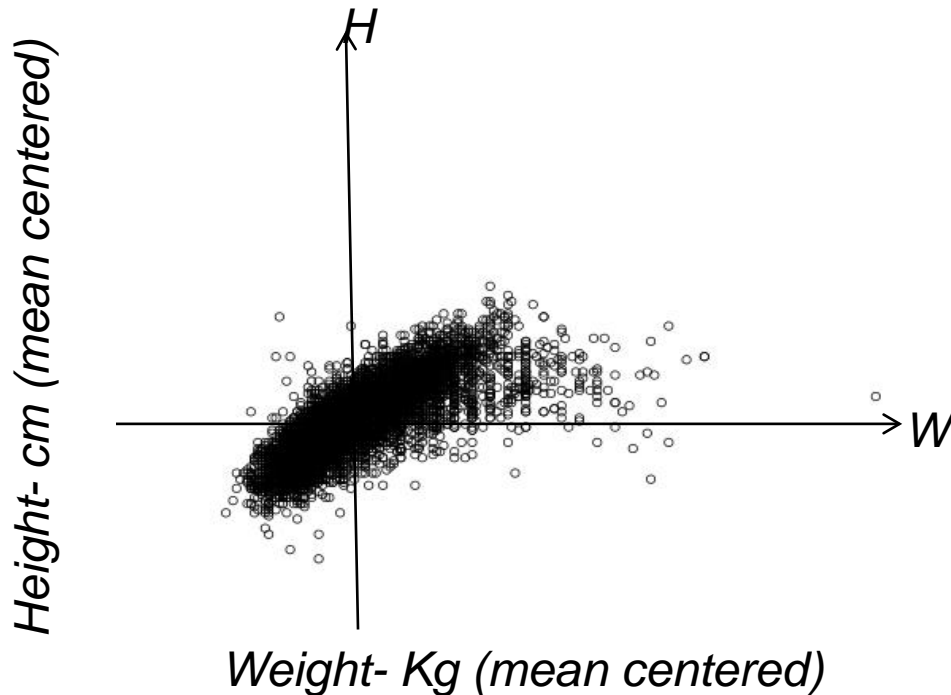
PLOT



***this is the  
input space,  
each row is  
a coordinate  
point***

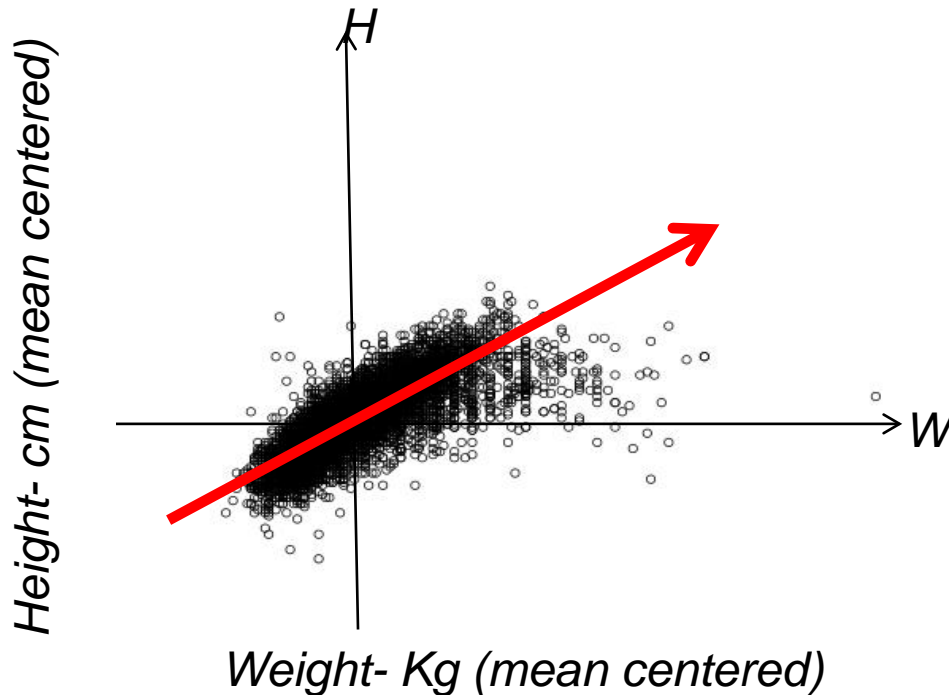


## Example: Athletes' Height by Weight



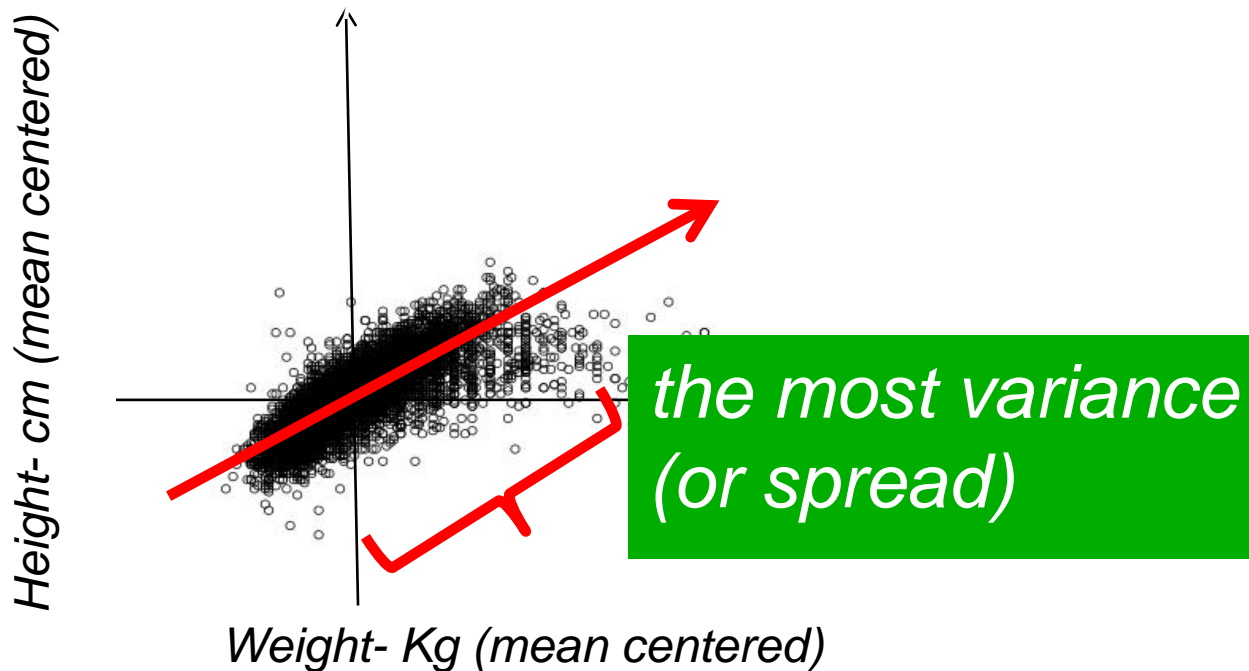
*Find a line that aligns with the data.*

## Example: Athletes' Height by Weight



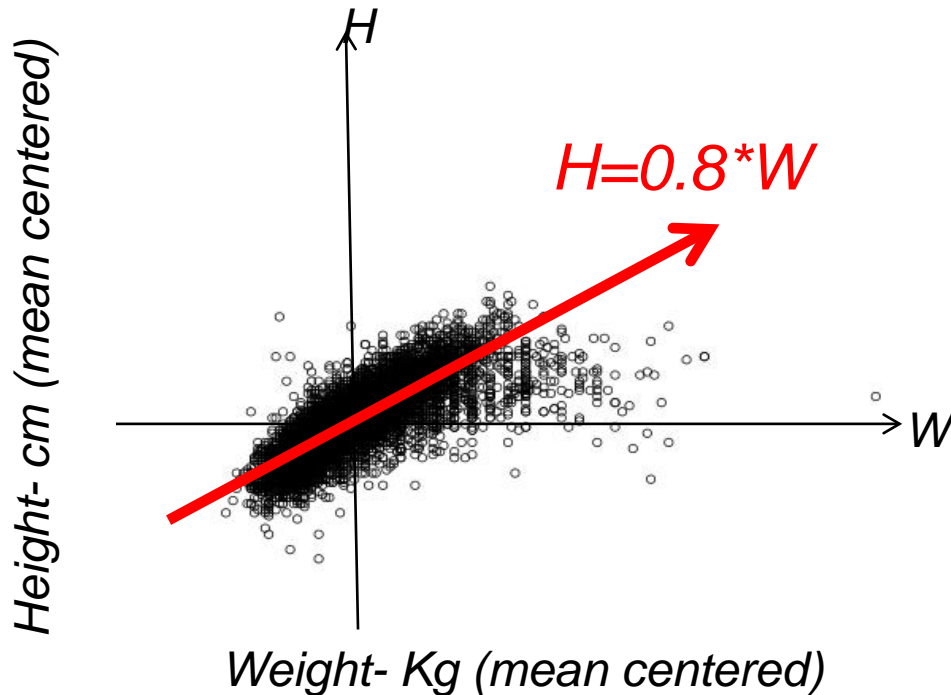
*Find a line that aligns with the data.*

## Example: Athletes' Height by Weight

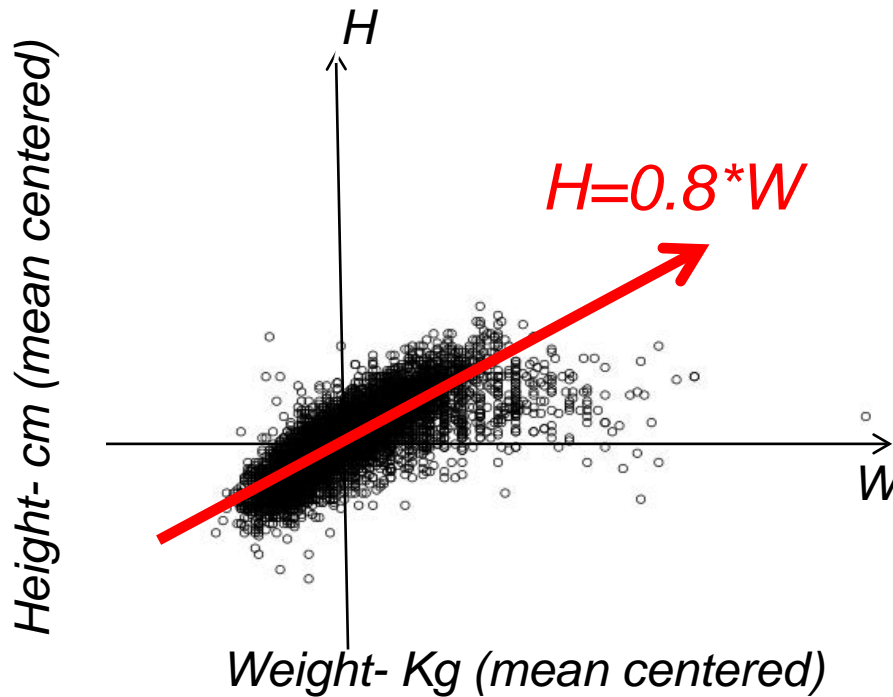


*Find a line that aligns with the data.*

## Example: Athletes' Height by Weight



*Find a line that aligns with the data.*



*Note that (0,0) and (1,0.8) are points on the line that are combinations of  $H, W$*

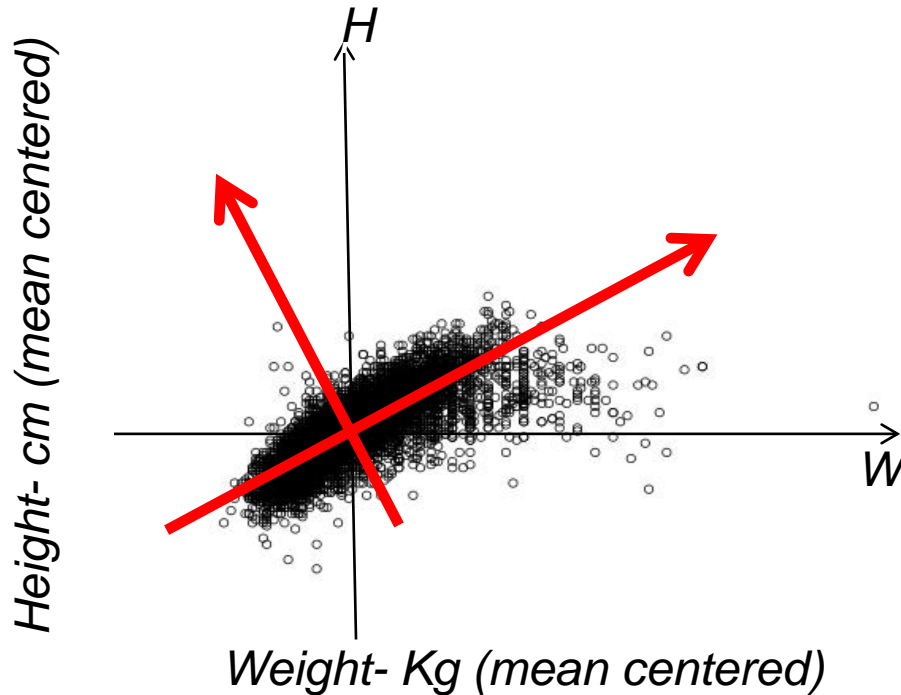
- OR -

$\begin{bmatrix} 1 \\ 0.8 \end{bmatrix}$

$\begin{bmatrix} 1 \\ 0.8 \end{bmatrix}$

*describes a vector in  $H \times W$  space*

*Find a line that aligns with the data.*



$\begin{bmatrix} -0.8 \\ 1.0 \end{bmatrix}$

$\begin{bmatrix} 1.0 \\ 0.8 \end{bmatrix}$

*describes a 2nd vector,  
orthogonal to 1<sup>st</sup> one*

- AND -

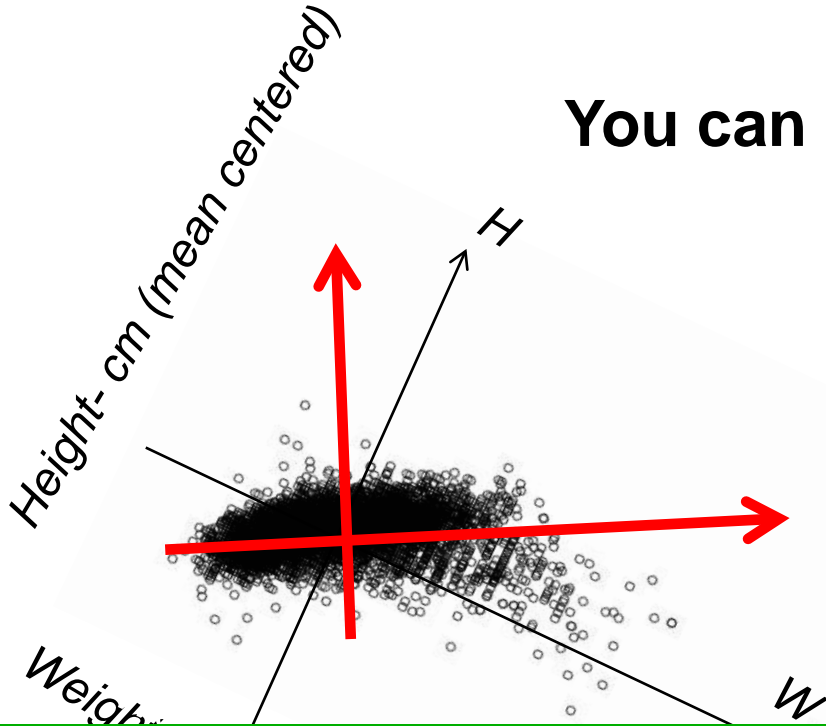
*Both vectors are columns in  
a matrix:*

$\begin{bmatrix} 1 & -0.8 \\ 0.8 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & -0.8 \\ 0.8 & 1 \end{bmatrix}$

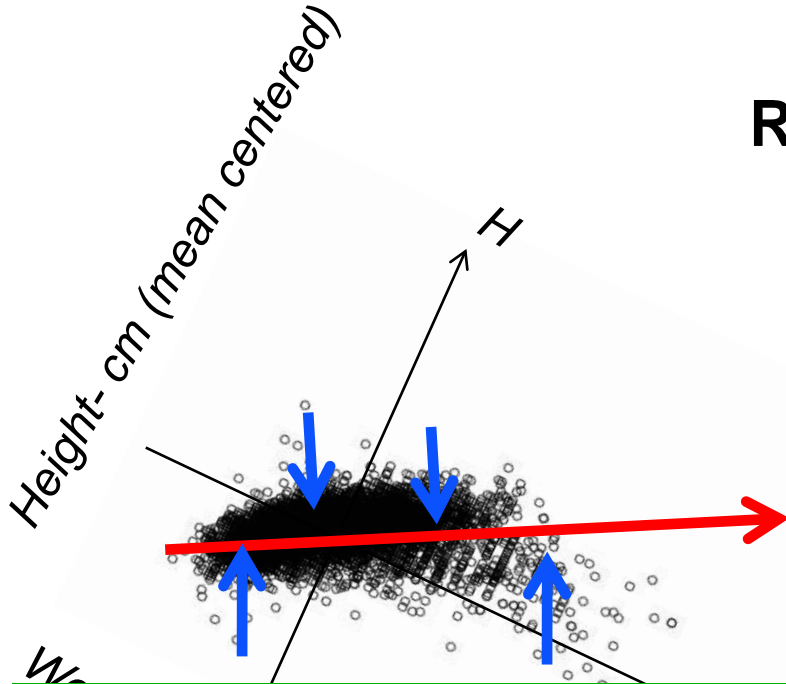
*The next direction of most variance.*

**You can rotate the axes**



*New axes (i.e., new features or latent factors) are combinations of old axis (i.e., old features or observed factors)*

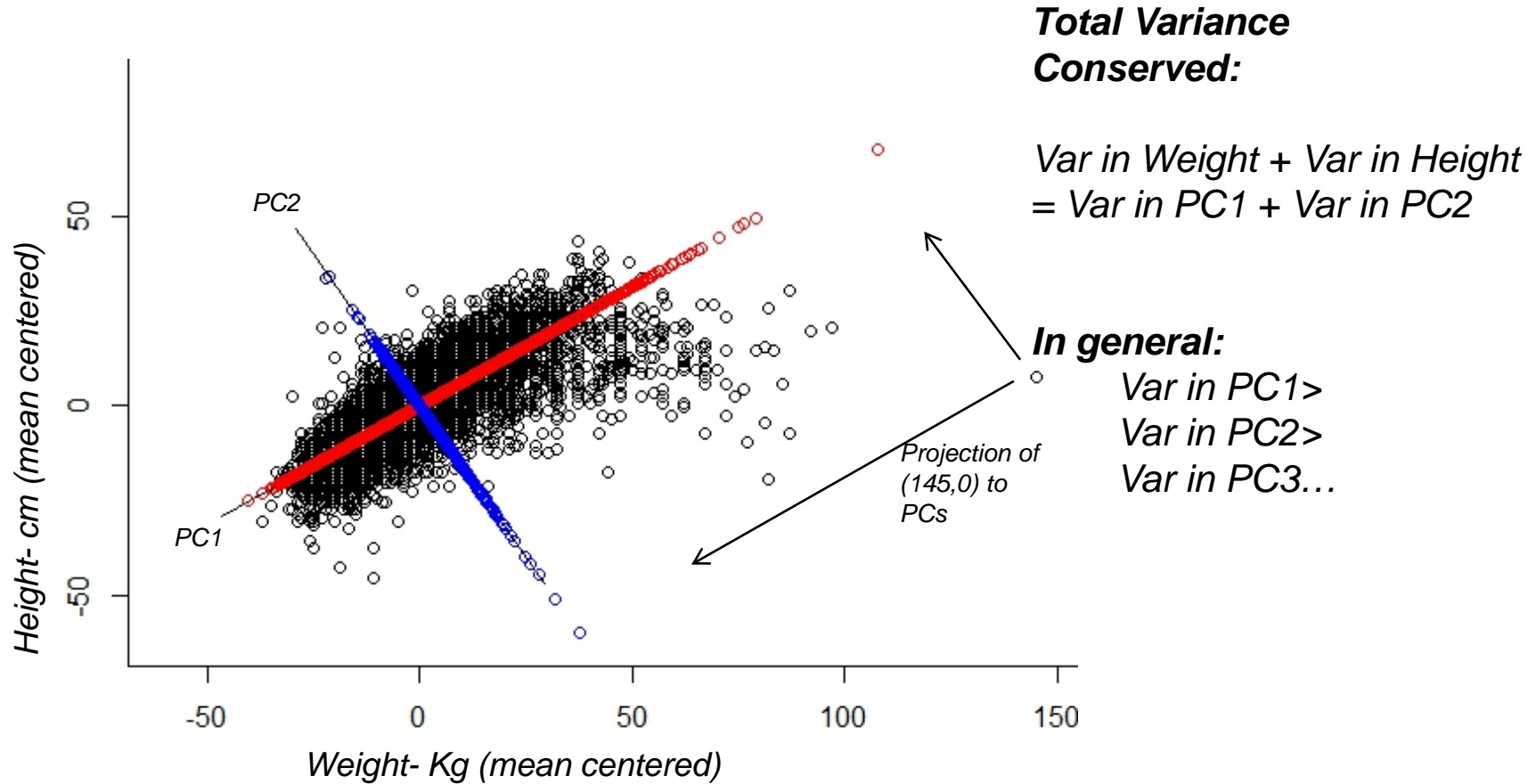
**Rotate axis**



*Project all points to first axis  
It keeps much of the variance*

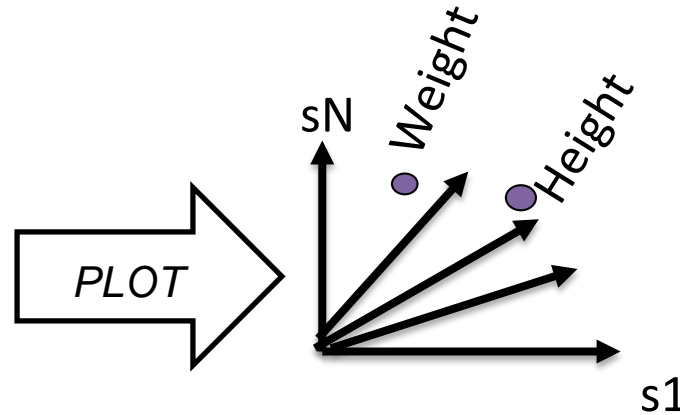


# Note: factorization conserves and reorders variance



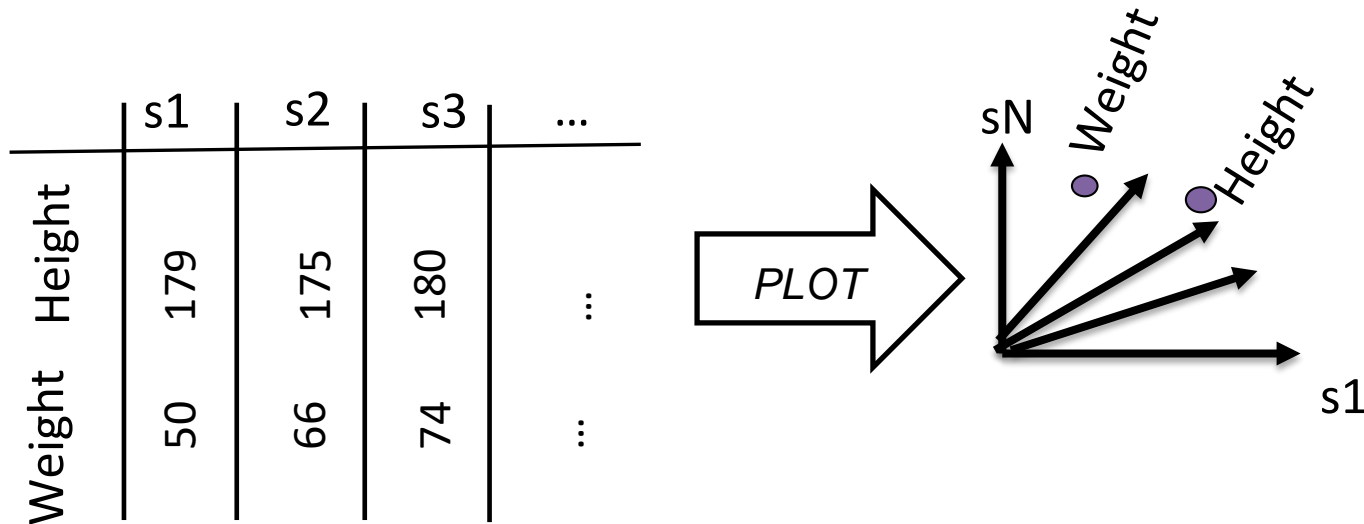
# 2D data transposed to 2 points in high dimensional space

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



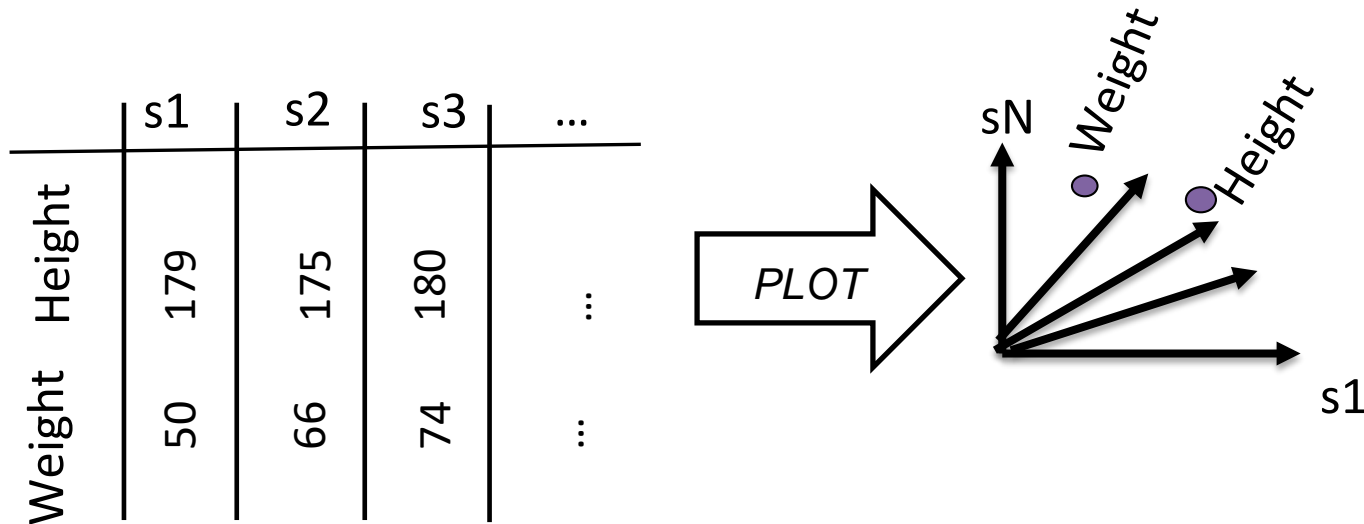
*We can also work in the transposed space (aka 'output' space or 'column' space)*

# 2D data transposed to 2 points in high dimensional space



*How do we find factors?*

# 2D data transposed to 2 points in high dimensional space



*How do we find factors?*

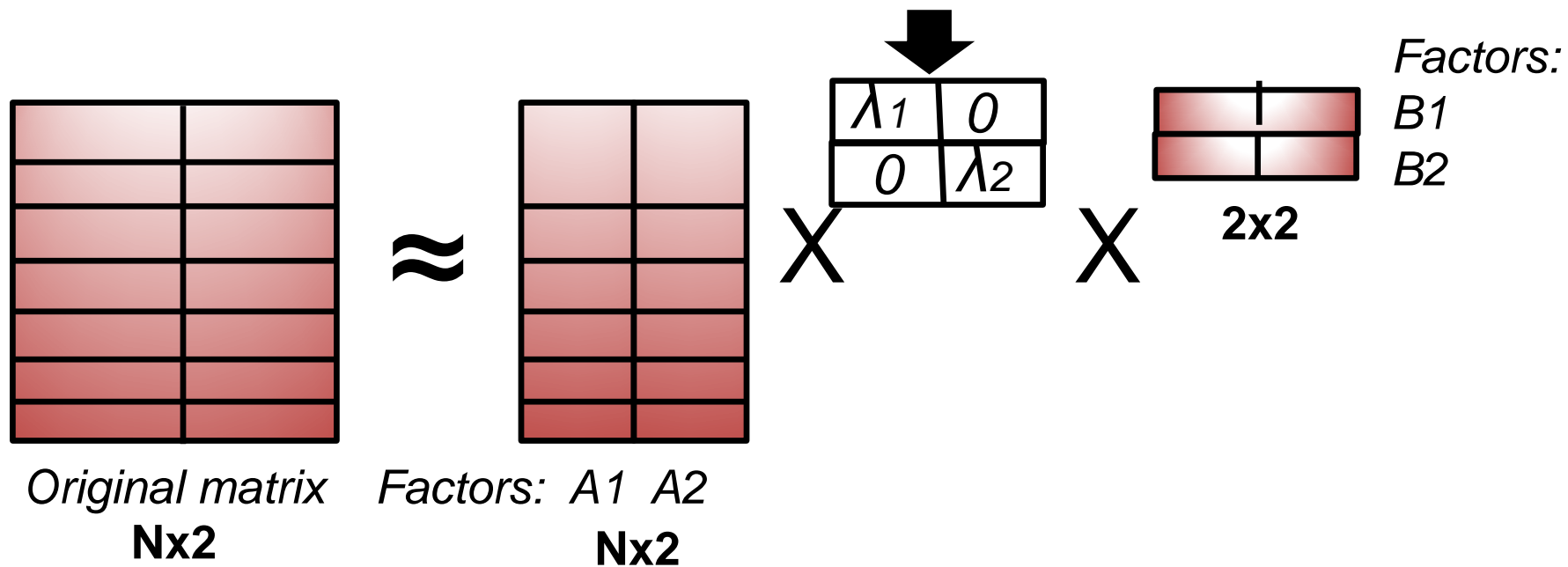
*Same process as before, but now factors are combinations of the  $s1...sN$  axes*

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

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

*SVD more generally works on non square matrices*

SVD produces factors (ie column vectors) and 'singular' values that are multiplied to recreate original matrix



- 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.*

## Summary: Principle Components

- Can choose  $k$  heuristically as approximation improves, or choose  $k$  so that high percent (ie 80-95%) of data variance accounted for
- 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

# Pause

# SVD Exercise

- Overview

Run on numeric fields of weather data

Run SVD and select smaller number of dimensions

Run linear model with original and reduced data

Load SVD R  
mark-down  
exercise

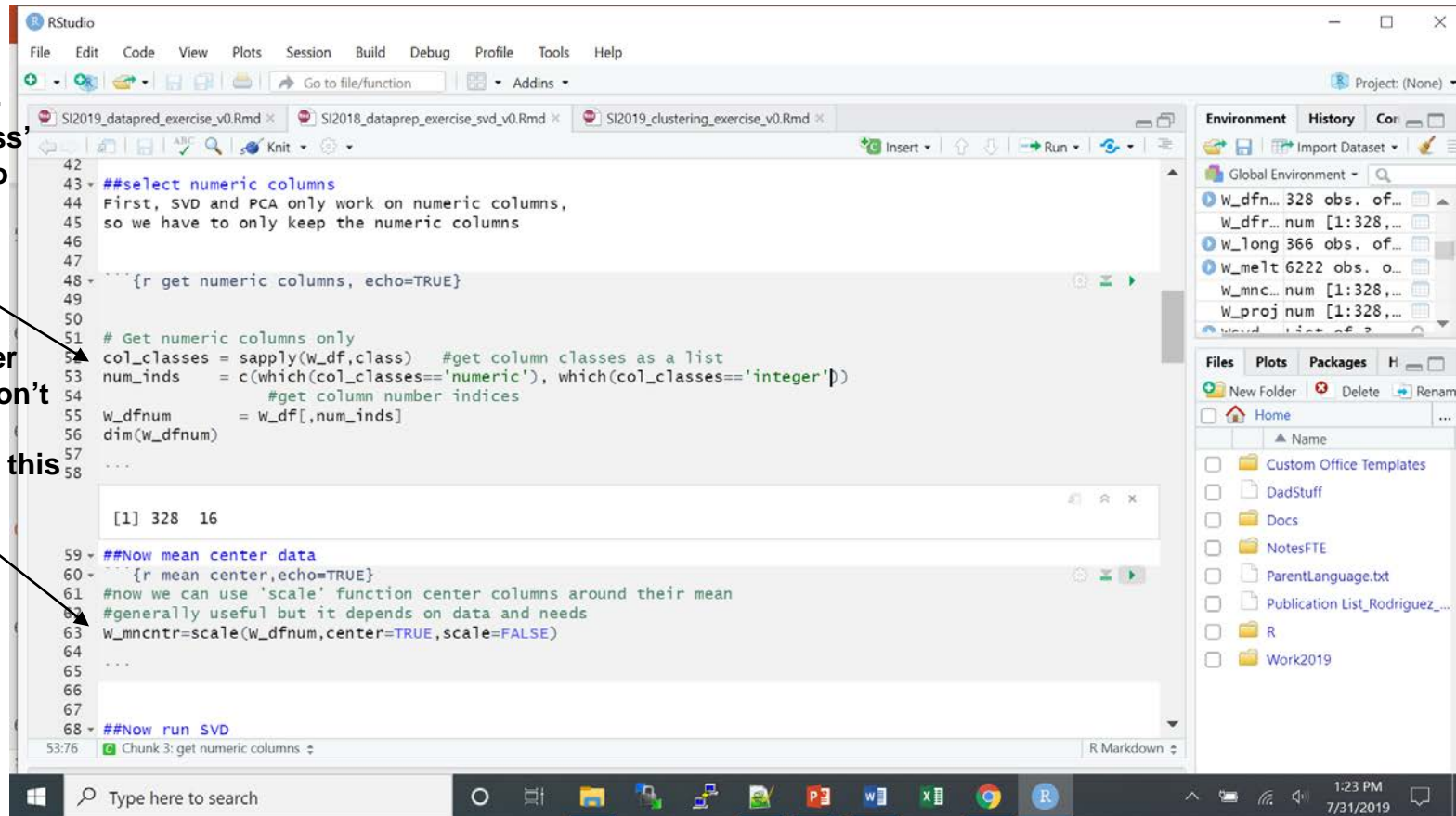
Find “load data” chunk.  
Run chunks above and  
current chunk

Take  
complete  
rows (ie no  
missing  
values), and  
take out the  
target  
variable

```
21- ##load data
22- {r load data}
23- #setwd("~/work2016/comet/SI2017/data")
24-
25- W_df_orig = read.table('weather_orig.csv',
26-                       header=TRUE, sep=",",
27-                       stringsAsFactors = TRUE) #try TRUE
28-
29- #Keep rows that are NOT missing data
30- keep_ind = complete.cases(W_df_orig)
31- W_df      = W_df_orig[keep_ind,]
32-
33- Y=as.numeric(W_df[, 'RainTomorrow']) #save thsi for later
34-
35- # subset with select is good to remove columns
36- W_df = subset(W_df, select=-c(RISK_MM))
37-
38- dim(W_df)
39- ...
40-
41- [1] 328 23
42-
43- ##select numeric columns
44- First, SVD and PCA only work on numeric columns,
45- so we have to only keep the numeric columns
46-
47-
1:1 SI2018_dataprep_exercise_svd_v1 R Markdown
```

Take  
numeric  
variables -  
apply 'class'  
function to  
each  
column

Mean center  
data, but don't  
scale the  
variance in this  
case



```
42  
43 ##select numeric columns  
44 First, SVD and PCA only work on numeric columns,  
45 so we have to only keep the numeric columns  
46  
47  
48 {r get numeric columns, echo=TRUE}  
49  
50  
51 # Get numeric columns only  
52 col_classes = sapply(w_df,class) #get column classes as a list  
53 num_inds = c(which(col_classes=='numeric'), which(col_classes=='integer'))  
54 #get column number indices  
55 w_dfnum = w_df[,num_inds]  
56 dim(w_dfnum)  
57  
58  
59 ##Now mean center data  
60 {r mean center,echo=TRUE}  
61 #now we can use 'scale' function center columns around their mean  
62 #generally useful but it depends on data and needs  
63 w_mncntr=scale(w_dfnum,center=TRUE,scale=FALSE)  
64  
65  
66  
67  
68 ##Now run SVD  
53:76 Chunk 3: get numeric columns
```

Environment

- Global Environment
- w\_dfn... 328 obs. of...
- w\_dfr... num [1:328,...
- w\_long 366 obs. of...
- w\_melt 6222 obs. o...
- w\_mnc... num [1:328,...
- w\_proj num [1:328,...

Files

- Home
- Custom Office Templates
- DadStuff
- Docs
- NotesFTE
- ParentLanguage.txt
- Publication List\_Rodriguez...
- R
- Work2019

# 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)
```

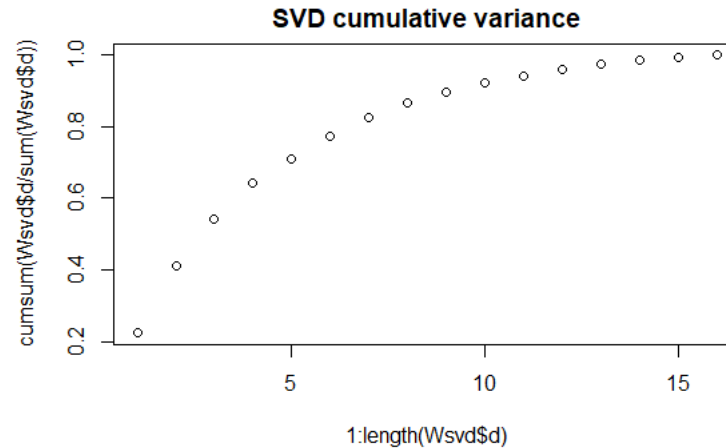
List of 3

```
$ 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 ...
```

# Exercise highlights



**Compare Linear Model results, using  $Y = \text{raintomorrow}$ :**

**look for residual standard error values and degree of freedom,  
look at coefficient estimates**



Take 3 SVD components, rebuild a  
Nx3 reduced dimension data matrix

```
92 #One could take first 3 components as an approximation to original data, for example
93
94 numcomp=3
95 #NOTE the %%% is matrix multiplication
96 w_dfred=wsvd$u[,1:numcomp] %%% diag(wsvd$d[1:numcomp]) %%% wsvd$v[1:numcomp,1:3]
97
98 dim(w_dfred)
99
100
101 ## For fun, run a linear model of Y= rainfall as a function of original data vs reduced data (numerical
102 fields)
103 ##check out the Residual standard error near the bottom of the results ##summaries, why are the degrees of
104 freedom different?
105 ##Compare the coefficient of the first component vs coefficient of the model ## with original variables - why
106 is it so high?
107
108 {r compare linear model on original vs reduced data}
109 Ymc = Y-mean(Y)
110 #result_orig=lm(Ymc~.,data=W_mncntr)
111 result_orig=lm(Ymc~W_mncntr)
112 #linear model takes Y~ matrix or Y~.,data=dataframe object
113 result_red =lm(Ymc~w_dfred)
114
115 summary(result_orig)
116
117 summary(result_red)
118
119 # For fun, run a linear model of Y= rainfall as a function of original data vs reduced data (numerical fields)
```

[1] 328 3

Mean center Y and compare linear  
models with original or reduced data  
matrix

Call:

lm(formula = Ymc ~ W\_mncntr)

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-1.124e-15	1.641e-02	0.000	1.000000
W_mncntrMinTemp	-1.368e-02	1.013e-02	-1.350	0.177844
W_mncntrMaxTemp	1.035e-02	2.010e-02	0.515	0.607120
W_mncntrRainfall	4.269e-03	4.471e-03	0.955	0.340442
W_mncntrEvaporation	2.690e-02	1.010e-02	2.663	0.008137 **
W_mncntrSunshine	-3.446e-02	9.898e-03	-3.482	0.000570 ***
W_mncntrPressure9am	6.569e-02	1.325e-02	4.960	1.16e-06 ***
W_mncntrPressure3pm	-8.047e-02	1.337e-02	-6.021	4.89e-09 ***

Residual standard error: 0.2971 on 311 degrees of freedom

Call:

lm(formula = Ymc ~ W\_dfred)

Coefficients: (13 not defined because of singularities)

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-4.874e-16	1.808e-02	0.000	1.000000
W_dfred1	4.519e+00	1.242e+00	3.638	0.000320 ***
W_dfred2	4.650e+00	1.307e+00	3.559	0.000429 ***
W_dfred3	1.580e+00	4.357e-01	3.627	0.000333 ***
W_dfred4	NA	NA	NA	NA
W_dfred5	NA	NA	NA	NA

Residual standard error: 0.3274 on 324 degrees of freedom

- end