Introduction to Data Analysis

John Myles White

September 8, 2013

Step 1: Get data

Sources:

- Spreadsheets
  - Excel
  - CSV
- Databases
  - MySQL
  - SAS
- System logs

Step 2: Translate unstructured data into tabular data

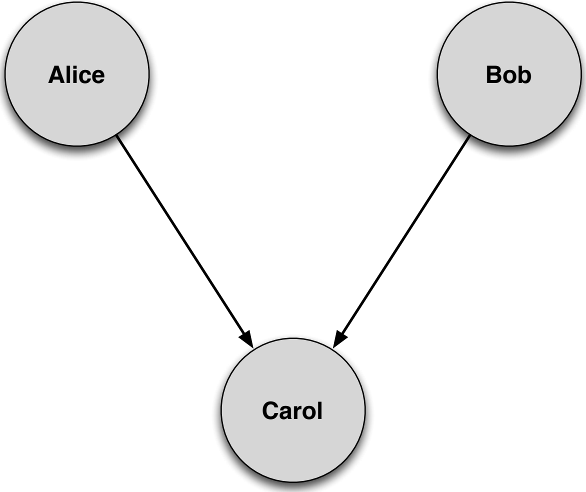
Good data is tabular data

Tabular Data

Year	Category	Value
2001	B	21.47114
2001	C	18.42930
2006	C	32.38368
?	B	59.11655
?	A	59.11655

Lots of data is not provided in tabular form

Social Graphs



Two possible tabular representations:

- Edge list
- Adjacency matrix

Edge List

Out	**In**
Alice	Carol
Bob	Carol

Adjacency Matrix

	Alice	Bob	Carol
Alice	0	0	1
Bob	0	0	1
Carol	0	0	0

Text Corpora

- Document 1: “I am a phrase”
- Document 2: “This is a phrase and that is too”
- Document 3: “This phrase not”

Two possible tabular representations:

- Word counts
- Word occurrences

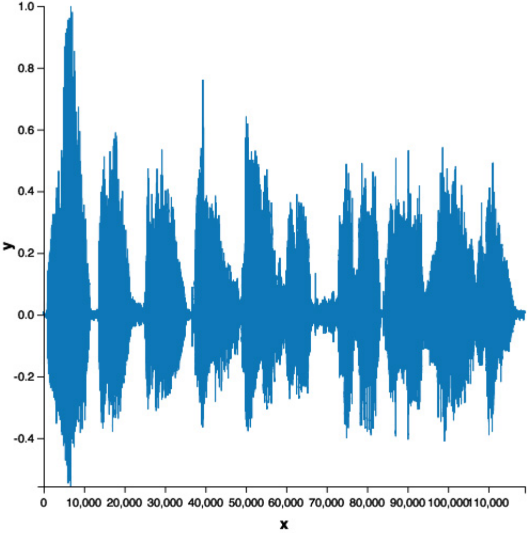
Word counts

I	am	a	phrase	this	is	and	that	too	not
1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	2	1	1	1	0
0	0	0	1	1	0	0	0	0	1

Word occurrences

I	**am**	**a**	**phrase**	**this**	**is**	**and**	**that**	**too**	**not**
1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	0	0	0	0	1

Sounds


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
y
0
10,000
20,000
30,000
40,000
50,000
60,000
70,000
80,000
90,000
100,000
110,000
x

Sounds

```
119000x1 Array{Float64,2}:
 0.00451674
 0.00534074
 0.00589007
 ...
 0.00787378
 0.00787378
 0.00897244
```

Images

```
julia> int(img.data)
400x325 Array{Int64,2}:
 133  133  132  131  133  135  135  132  …   61   62   65   65   67   69   69
 133  134  134  133  133  133  133  132      62   63   65   67   69   69   69
 133  134  135  135  134  132  133  134      62   64   66   69   71   69   69
 133  133  135  136  135  133  133  135      62   63   66   69   70   70   70
 135  134  135  136  137  135  134  135      61   63   66   66   69   71   71
 137  136  136  136  137  136  134  134  …   61   62   65   65   67   71   71
 137  137  137  136  136  136  135  134      62   62   65   66   68   71   71
 136  138  138  136  135  137  137  136      63   63   65   67   69   70   70
 139  138  138  138  137  136  135  135      65   67   67   65   69   69   69
 140  139  138  138  137  136  135  136      64   66   68   67   70   71   71
   ⋮                        ⋮             ⋱             ⋮
 207  209  209  208  209  210  211  210     173  173  172  171  171  170  170
 208  208  209  209  210  210  209  208     175  175  172  171  170  171  171
 209  209  208  208  209  209  209  209     172  171  171  170  169  170  170
 210  209  208  207  208  209  210  210     172  171  171  170  170  170  170
 209  208  208  207  208  209  209  210  …  172  171  170  170  169  169  169
 208  208  208  208  208  208  208  208     170  169  168  169  169  168  168
 206  207  208  209  208  208  207  207     170  170  169  169  168  167  167
 206  207  208  208  207  207  206  206     169  168  166  166  164  162  162
 207  207  208  207  206  206  206  206     164  162  161  160  157  154  154

julia> 
```

Step 3: Do something with your tabular data

Two classes of operations

- Reduce the size of tables
- Fill in tables

Data reductions

- Summary statistics
- Dimensionality reduction
- Clustering

Summary Statistics

AAPL	INTC	GOOG
616.76	23.94	757.00
623.00	21.73	743.10
601.34	19.16	713.32
599.34	20.85	723.24

Mean AAPL	Mean INTC	Mean GOOG
610.11	21.42	734.17

Dimensionality Reduction

AAPL	INTC	GOOG
616.76	23.94	757.00
623.00	21.73	743.10
601.34	19.16	713.32
599.34	20.85	723.24

DJI
13603.14
13614.14
13632.14
13589.14

Clustering

AAPL	INTC	GOOG
616.76	23.94	757.00
623.00	21.73	743.10
601.34	19.16	713.32
599.34	20.85	723.24

Market
Bull
Bull
Bear
Bear

Filling in data

- Regression
- Classification

Regression

Inputs　　　　Outputs

Year	Timbre1	Timbre2
2001	49.94357	21.47114
2001	48.73215	18.42930
2006	44.16614	32.38368
2011	51.85726	?

Classification

Outputs

Inputs

IsSpam	MentionsViagra	MentionsNigeria
Yes	No	Yes
?	Yes	Yes
No	No	No
Yes	Yes	No

Let's dig in to details now

Summary statistics

Two main categories:

- Summarize the "typical" value
- Summarize variability around typical values

Summary Statistics

AAPL	INTC	GOOG
616.76	23.94	757.00
623.00	21.73	743.10
601.34	19.16	713.32
599.34	20.85	723.24

Mean AAPL	Mean INTC	Mean GOOG
610.11	21.42	734.17

Typical value summaries

- Mean
- Median
- Mode
- Midrange

The mean of a vector is the average

```r
x <- c(9, 9, 10, 11, 9, 10, 11)

mean(x) # => 9.857143

sum(x) / length(x) # => 9.857143
```

The median of a vector is the center of the sorted values

```
median(x) # => 10

sort(x)[ceiling(length(x) / 2)] # => 10
```

The mode of a vector is the most frequently occurring value

```
mode <- function(x)
{
    T <- table(x)
    return(names(T)[which(T == max(T))])
}
mode(x)
```

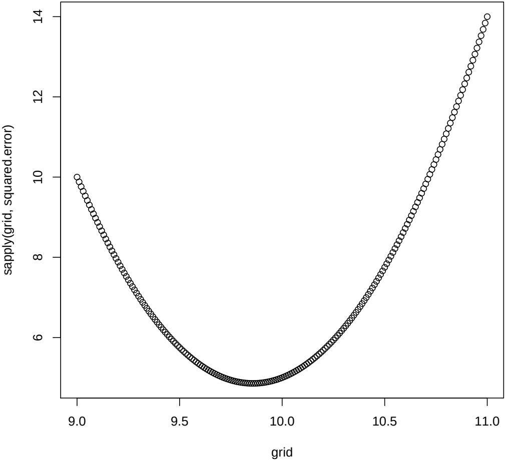
The midrange of a vector lies halfway between its bounds

```
midrange <- function (x)
{
    return(min(x) + (max(x) - min(x)) / 2)
}
midrange(x)
```

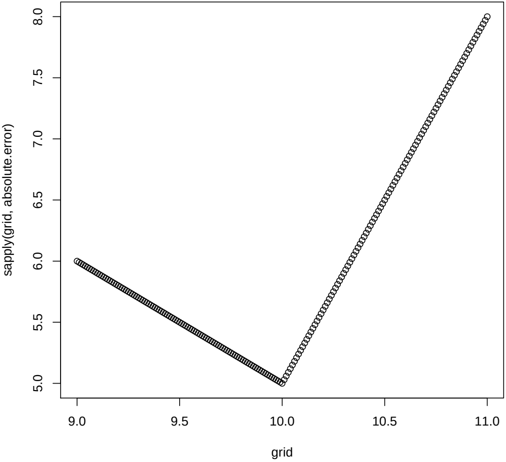
Why are these four summaries special?

They minimize different errors when approximating data

```
squared.error <- function(s)
{
    return(sum(abs(x - s)^2))
}
grid <- seq(min(x), max(x), by = 0.01)
plot(grid, sapply(grid, squared.error))
```

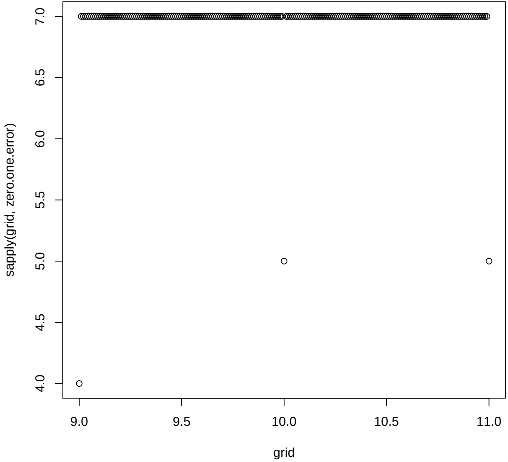

sapply(grid, squared.error)
grid
6
8
10
12
14
9.0
9.5
10.0
10.5
11.0

```
absolute.error <- function(s)
{
    return(sum(abs(x - s)))
}
grid <- seq(min(x), max(x), by = 0.01)
plot(grid, sapply(grid, absolute.error))
```



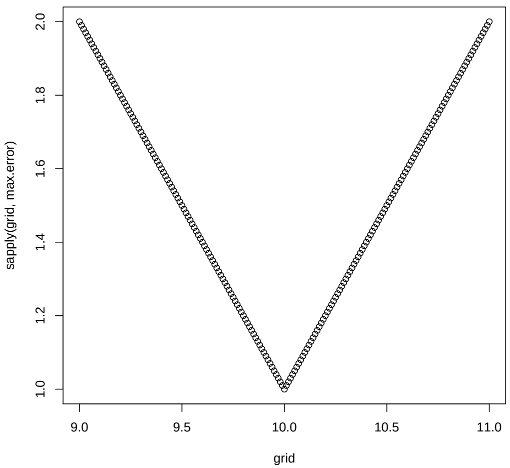
sapply(grid, absolute.error)
8.0
7.5
7.0
6.5
6.0
5.5
5.0
9.0
9.5
10.0
10.5
11.0
grid

```
zero.one.error <- function(s)
{
    return(sum(abs(x != s)))
}
grid <- seq(min(x), max(x), by = 0.01)
plot(grid, sapply(grid, zero.one.error))
```



sapply(grid, zero.one.error)
7.0
6.5
6.0
5.5
5.0
4.5
4.0
9.0
9.5
10.0
10.5
11.0
grid

```
max.error <- function(s)
{
    return(max(abs(x - s)))
}
grid <- seq(min(x), max(x), by = 0.01)
plot(grid, sapply(grid, max.error))
```



sapply(grid, max.error)
2.0
1.8
1.6
1.4
1.2
1.0
9.0
9.5
10.0
10.5
11.0
grid

Summary statistics summarized

- Mean: Minimizes squared error
- Median: Minimizes absolute error
- Mode: Minimizes zero-one error
- Midrange: Minimizes max error

Homework Problems

- Prove the claims made about summary statistics are correct
  - Nested problem: relearn any rusty math
- Make those summary statistics work on huge data sets
  - Compute the mean without storing vector in memory
  - Compute the median without storing vector in memory
  - Compute mode without maintaing list of unique items

Variability summaries

- Range: Min-Max
- Standard Deviation / Variance
- Median Absolute Deviation
- Inter-Quantile Range (IQR)

```
min(x) # => 9
max(x) # => 11
range(x) #=> c(9, 11)
```

```
sd(x) # => 0.8997354
var(x) # => 0.8095238
```

```
my.var <- function(x)
{
    return(sum(abs(x - mean(x))^2) / (length(x) - 1))
}
```

Why divide by length(x) - 1?

This gives correct result when averaging across data sets

```
mad(x) #=> 1.4826
```

```
my.mad <- function(x)
{
    return(1.4826 * median(abs(x - median(x))))
}
```

1.4826 makes estimate behave like standard deviation

```
IQR(x) # => 1.5
```

```
my.IQR <- function(x)
{
    return(diff(quantile(x, c(0.75, 0.25))))
}
```

Almost everything in data analysis reduces to these ideas

Break for questions

Dimensionality reduction

Dimensionality Reduction

AAPL	INTC	GOOG
616.76	23.94	757.00
623.00	21.73	743.10
601.34	19.16	713.32
599.34	20.85	723.24

DJI
13603.14
13614.14
13632.14
13589.14

Try to produce a lower-dimensional representation of data

Common forms:

- PCA - Principal Components Analysis
- ICA - Independent Components Analysis
- NMF - Non-Negative Matrix Factorization

PCA computes a multivariate "mean"

```
stocks <- read.csv(file.path('data', 'stocks.csv'))

dates <- stocks[, 1]

pca <- princomp(stocks[, 2:ncol(stocks)])
```

```
market.index <- predict(pca)[, 1]

dji <- read.csv(file.path('data', 'dji.csv'),
                stringsAsFactors = FALSE)[, 1]

comparison <- data.frame(Date = dates,
                         MarketIndex = market.index,
                         DJI = dji)
```

```
library(ggplot2)

ggplot(comparison, aes(x = MarketIndex, y = DJI)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE)
```

```
comparison <- transform(comparison,
                        MarketIndex = -1 * MarketIndex)

ggplot(comparison, aes(x = MarketIndex, y = DJI)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE)
```

```
alt.comparison <- melt(comparison, id.vars = 'Date')

names(alt.comparison) <- c('Date', 'Index', 'Price')

ggplot(alt.comparison,
       aes(x = Date,
           y = Price,
           group = Index,
           color = Index)) +
  geom_point() +
  geom_line()
```

```
comparison <- transform(comparison,
                        MarketIndex = scale(MarketIndex))
comparison <- transform(comparison,
                        DJI = scale(DJI))
```

```
alt.comparison <- melt(comparison, id.vars = 'Date')

names(alt.comparison) <- c('Date', 'Index', 'Price')

ggplot(alt.comparison,
       aes(x = Date, y = Price,
           group = Index, color = Index)) +
  geom_point() +
  geom_line()
```

Clustering

Replace each row with a category label

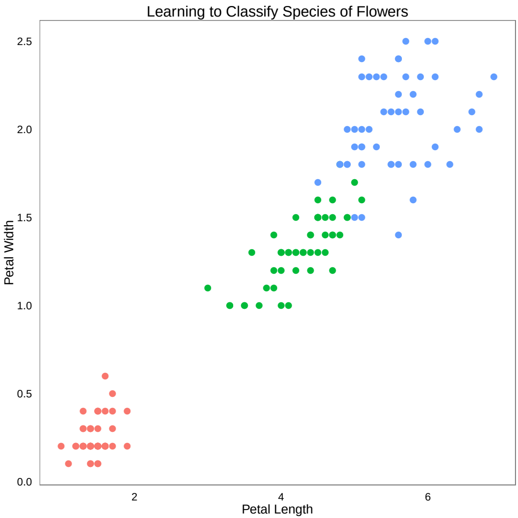
Common forms:

- k-Means
- Hierarchical clustering
- Spectral clustering

```
data(iris)
```

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
ggplot(iris,
       aes(x = Petal.Length,
           y = Petal.Width,
           color = Species)) +
  geom_point()
```


Learning to Classify Species of Flowers
Petal Width
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
Petal Length

```
kmeans(iris[, 1:4], 3)
```

```
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3
[149] 3 2
```

```
library("plyr")
ddply(iris, "Species",
      function (df) {data.frame(X1 = mean(df[, 1]),
                                X2 = mean(df[, 2]),
                                X3 = mean(df[, 3]),
                                X4 = mean(df[, 4]))})
```

	Species	X1	X2	X3	X4
1	setosa	5.006	3.428	1.462	0.246
2	versicolor	5.936	2.770	4.260	1.326
3	virginica	6.588	2.974	5.552	2.026

Break for questions

Switch from summaries to filling in operations

Regression

Inputs

Outputs

Year	Timbre1	Timbre2
2001	49.94357	21.47114
2001	48.73215	18.42930
2006	44.16614	32.38368
2011	51.85726	?

A toy regression problem

Fahrenheit	**Celsius**
212º	100º
32º	0º

Solve in R

```r
df <- data.frame(Fahrenheit = c(212, 32),
                 Celsius = c(100, 0))

lm.fit <- lm(Fahrenheit ~ Celsius, data = df)

summary(lm.fit)

predict(lm.fit, data.frame(Celsius = 40))
```

Results in R

```
> summary(lm.fit)

Call:
lm(formula = Fahrenheit ~ Celsius, data = df)

Residuals:
ALL 2 residuals are 0: no residual degrees of freedom!

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     32.0         NA      NA       NA
Celsius          1.8         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:    NaN
F-statistic:   NaN on 1 and 0 DF,  p-value: NA

>
> predict(lm.fit, data.frame(Celsius = 40))
  1
104
```

Solve variant in R

```r
df <- data.frame(Fahrenheit = c(212, 102, 32),
                 Celsius = c(100, 50, 0))

lm.fit <- lm(Fahrenheit ~ Celsius, data = df)

summary(lm.fit)

predict(lm.fit, data.frame(Celsius = 40))
```

Results in R

```
Call:
lm(formula = Fahrenheit ~ Celsius, data = df)

Residuals:
      1       2       3 
  6.667 -13.333   6.667 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  25.3333    14.9071   1.699   0.3386  
Celsius       1.8000     0.2309   7.794   0.0812 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.33 on 1 degrees of freedom
Multiple R-squared: 0.9838,	Adjusted R-squared: 0.9676 
F-statistic: 60.75 on 1 and 1 DF,  p-value: 0.08123

> 
> predict(lm.fit, data.frame(Celsius = 40))
       1 
97.33333 
```

Classification

Outputs

Inputs

IsSpam	MentionsViagra	MentionsNigeria
Yes	No	Yes
?	Yes	Yes
No	No	No
Yes	Yes	No

A toy classification problem

IsSpam	MentionsViagra	MentionsNigeria
Yes	No	Yes
Yes	Yes	Yes
No	No	No
Yes	Yes	No

Convert categories to numbers

IsSpam	MentionsViagra	MentionsNigeria
1	0	1
1	1	1
0	0	0
1	1	0

Solve in R

```
df <- data.frame(IsSpam = c(1, 1, 0, 1),
                 MentionsViagra = c(0, 1, 0, 1),
                 MentionsNigeria = c(1, 1, 0, 0))

logistic.fit <- glm(IsSpam ~ MentionsViagra + MentionsNigeria,
                    data = df,
                    family = binomial(link = "logit"))

summary(logistic.fit)
```

Results in R

```
glm(formula = IsSpam ~ MentionsViagra + MentionsNigeria, family = binomial(l
= "logit"),
    data = df)

Deviance Residuals:
         1           2           3           4
 1.197e-05   2.110e-08  -1.197e-05   1.197e-05

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)       -23.36   71664.47       0        1
MentionsViagra     46.72  101348.81       0        1
MentionsNigeria    46.72  101348.81       0        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4.4987e+00  on 3  degrees of freedom
Residual deviance: 4.2978e-10  on 1  degrees of freedom
AIC: 6

Number of Fisher Scoring iterations: 23
```

Solve variant in R

```r
df <- data.frame(IsSpam = c(1, 1, 0, 1, 0),
                 MentionsViagra = c(0, 1, 0, 1, 1),
                 MentionsNigeria = c(1, 1, 0, 0, 1))

logistic.fit <- glm(IsSpam ~ MentionsViagra + MentionsNigeria,
                    data = df,
                    family = binomial(link = "logit"))

summary(logistic.fit)
```

Results in R

```
glm(formula = IsSpam ~ MentionsViagra + MentionsNigeria, family = binomial(link
= "logit"),
    data = df)

Deviance Residuals:
      1        2        3        4        5
 1.0519   0.8234  -1.0519   1.0519  -1.5789

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -0.3025     1.7188  -0.176    0.860
MentionsViagra    0.6050     1.9060   0.317    0.751
MentionsNigeria   0.6050     1.9060   0.317    0.751

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.7301  on 4  degrees of freedom
Residual deviance: 6.4907  on 2  degrees of freedom
AIC: 12.491

Number of Fisher Scoring iterations: 4
```

Richer case study

- Predict web site popularity
- Predict book sales for O'Reilly's books

Web Data

Rank	Site	Category
1	facebook.com	Social
2	youtube.com	Online
3	yahoo.com	Web
4	live.com	Search

Web Data

UniqueVisitors	Reach	PageViews
880000000	47.2	9.1e+11
800000000	42.7	1.0e+11
660000000	35.3	7.7e+10
550000000	29.3	3.6e+10

Web Data

HasAdvertising	InEnglish	TLD
Yes	Yes	.com
Yes	Yes	.com
Yes	Yes	.com
Yes	Yes	.com

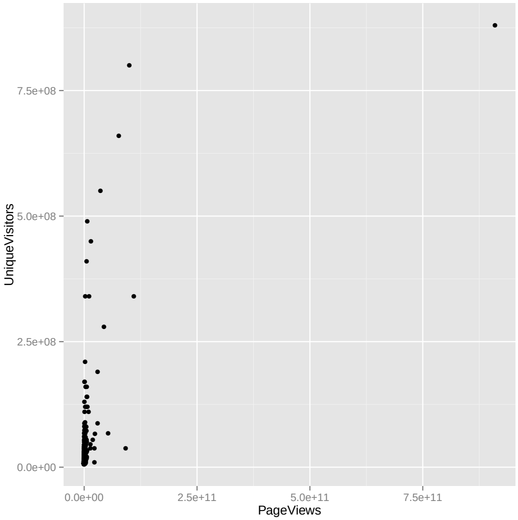
Load and Check Web Data

```
top.1000.sites <- read.csv(file.path('data', 'top_1000_site
                           sep = '\t',
                           stringsAsFactors = FALSE)
head(top.1000.sites, n = 4)
```

Page Views vs Visitors

```
ggplot(top.1000.sites, aes(x = PageViews, y = UniqueVisitor
  geom_point()
ggsave(file.path("images", "page_views_vs_visitors.pdf"))
```

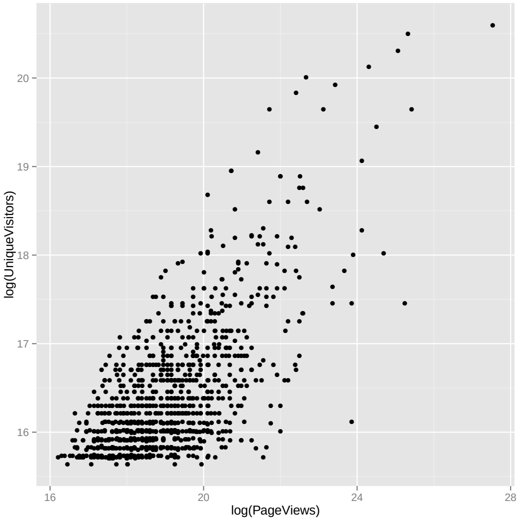
Page Views vs Visitors



Log Page Views vs Log Visitors

```
ggplot(top.1000.sites, aes(x = log(PageViews),
                           y = log(UniqueVisitors))) +
  geom_point()
ggsave(file.path("images", "log_page_views_vs_log_visitors
```

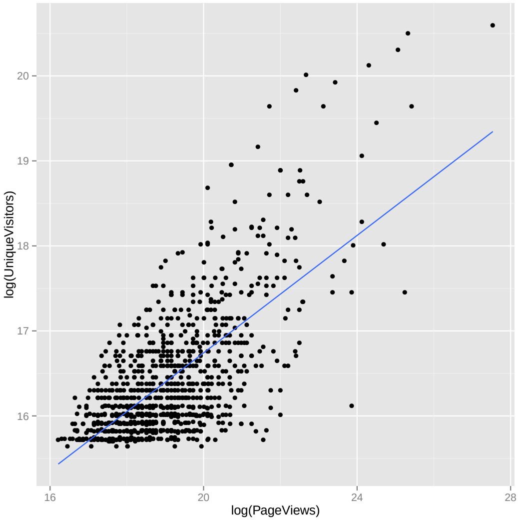
Log Page Views vs Log Visitors



Visual Linear Regression

```
ggplot(top.1000.sites, aes(x = log(PageViews),
                           y = log(UniqueVisitors))) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE)
ggsave(file.path("images",
                 "log_page_views_vs_log_visitors_with_lm.pd
```

Visual Linear Regression



Simple Linear Regression

```
lm.fit <- lm(log(PageViews) ~ log(UniqueVisitors),
             data = top.1000.sites)
summary(lm.fit)
```

Simple Linear Regression

```
Call:
lm(formula = log(PageViews) ~ log(UniqueVisitors), data = t

Residuals:
Min 1Q Median 3Q Max
-2.1825 -0.7986 -0.0741 0.6467 5.1549

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.83441 0.75201 -3.769 0.000173 ***
log(UniqueVisitors) 1.33628 0.04568 29.251 < 2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘

Residual standard error: 1.084 on 998 degrees of freedom
Multiple R-squared: 0.4616, Adjusted R-squared: 0.4611
F-statistic: 855.6 on 1 and 998 DF, p-value: < 2.2e-16
```

Super-Charged Linear Regression

```
lm.fit <- lm(log(PageViews) ~ HasAdvertising +
                              log(UniqueVisitors) +
                              InEnglish,
             data = top.1000.sites)
summary(lm.fit)
```

Super-Charged Linear Regression

```
Call:
lm(formula = log(PageViews) ~ HasAdvertising + log(UniqueVi
InEnglish, data = top.1000.sites)

Residuals:
Min 1Q Median 3Q Max
-2.4283 -0.7685 -0.0632 0.6298 5.4133

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.94502 1.14777 -1.695 0.09046 .
HasAdvertisingYes 0.30595 0.09170 3.336 0.00088 ***
log(UniqueVisitors) 1.26507 0.07053 17.936 < 2e-16 ***
InEnglishNo 0.83468 0.20860 4.001 6.77e-05 ***
InEnglishYes -0.16913 0.20424 -0.828 0.40780
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

Measuring Predictive Power

```
lm.fit <- lm(log(PageViews) ~ HasAdvertising,
             data = top.1000.sites)
summary(lm.fit)$r.squared
[1] 0.01073766

lm.fit <- lm(log(PageViews) ~ log(UniqueVisitors),
             data = top.1000.sites)
summary(lm.fit)$r.squared
[1] 0.4615985

lm.fit <- lm(log(PageViews) ~ InEnglish,
             data = top.1000.sites)
summary(lm.fit)$r.squared
[1] 0.03122206
```

Correlation and Causation

```
x <- 1:10
y <- x^2

cor(x, y)
[1] 0.9745586

coef(lm(scale(y) ~ scale(x)))[2]
[1] 9.745586e-01
```

- Regularization
- Cross-Validation
- Text Regression
- Optimization