

MIS 572:

Introduction to Big Data Analytics

Fundamentals of Data Analytics

Yihuang K. Kang

Why "Fundamentals"?

- Statistics, probability, and programming are the fundamentals of the Data Science. It's always helpful to go back to them and remind what statistical, computational, and model thinking are about.
- In this unit, we will have a quick review on basic statistical analysis methods & data analytics process with help of R, and discuss their roles in the age of Big Data. Those of you who need Statistics concepts may refer to online course [Statistics and Probability at Khan Academy](#).

Statistical Inference

- The world is *complex, stochastic, and dynamic*. It keeps producing *data* and will never stop. Now you have *the data*, whatever the format it is, it may give you some *information* about the world.
- The *Statistical Inference* is a process of deducing *properties* of your data assumed to be generated by random processes. By "properties" here, we mean models, estimations, acceptance of hypothesis..., all the information that helps you understand the world of interests.

Different terms, same practices

- Most of terms used in "Statistics Modeling" and "Machine Learning" actually mean the same and are often interchangeable. In this class, these terms will be mixed, depending on the terms commonly used in literatures.

Statistical Modeling	Machine Learning
Explanatory/Independent Variable	Input, Predictor, Feature
Dependent/Outcome Variable	Output, Target
Model	Network, Graphic, Algorithm
Parameter	Weight
Fitting	Learning
Test data performance	Generalization
Regression/ Classification	Supervised Learning
Density Estimation/Clustering	Unsupervised Learning

So? What's New?

- Often, we measure the world and collect the data with carefully designed experiments, then work for years to learn plausible models from small datasets. The ways to select the key features and to explain the result of analysis rely heavily on domain experts, which sometimes, unfortunately, results in subjective and questionable interpretations of the world.
- In the age of Big Data, however, we start with large datasets with many features, and use statistical learning algorithms to find better models. Instead of manually doing inefficient & costly feature engineering, we tend to let machines automatically discover different (and even stacked) abstract representations of data that best predict the outcome and describe the characteristics of the data—a practice of [Feature Learning](#).

The Two Cultures of Modeling

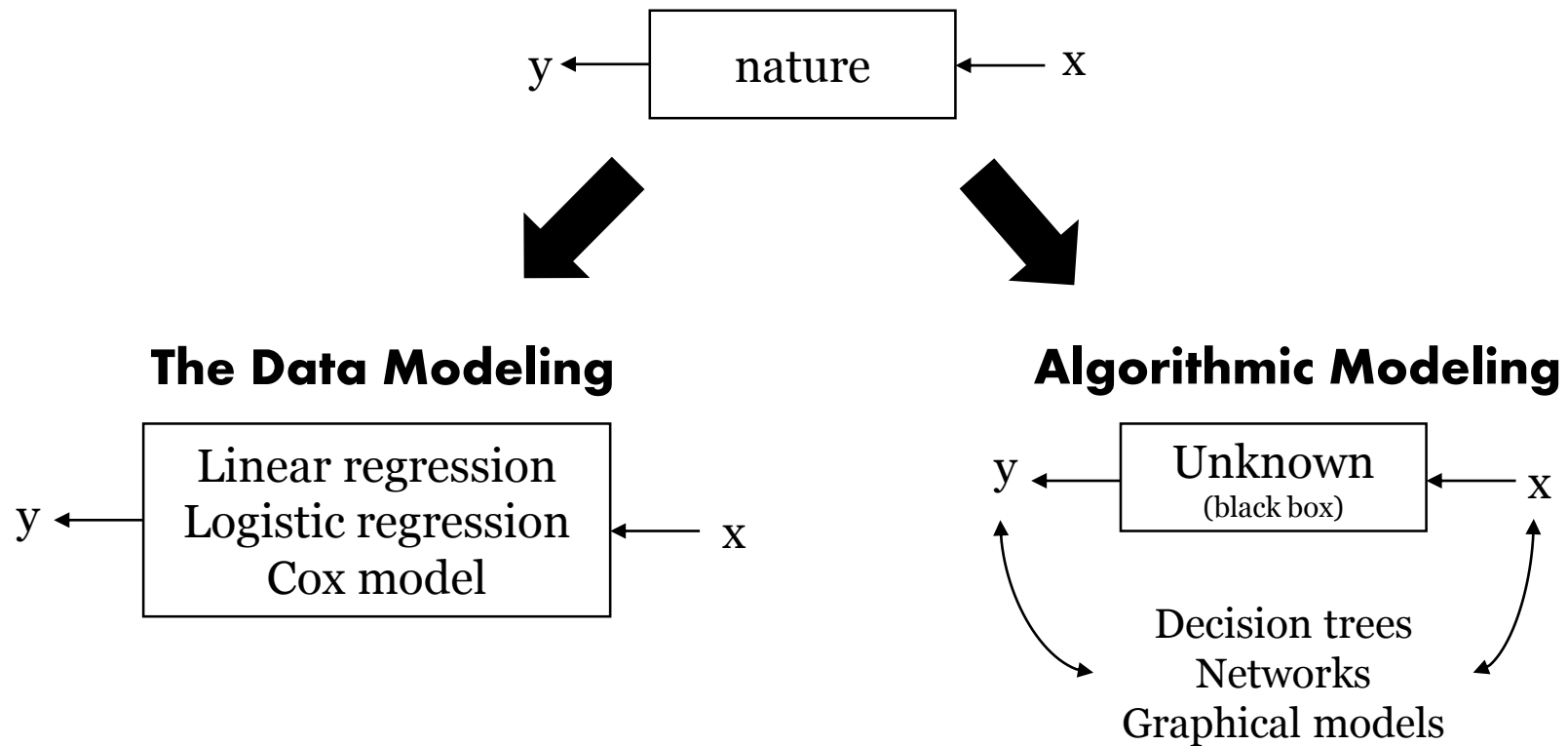


"...the focus in the statistical community on data models has led to irrelevant theory and questionable scientific conclusions..."

— [Leo Breiman](#)

- Remember to check out the article "[Statistical modeling: The two cultures](#)", also on our reading list.

The Two Cultures of Modeling_(cont.)



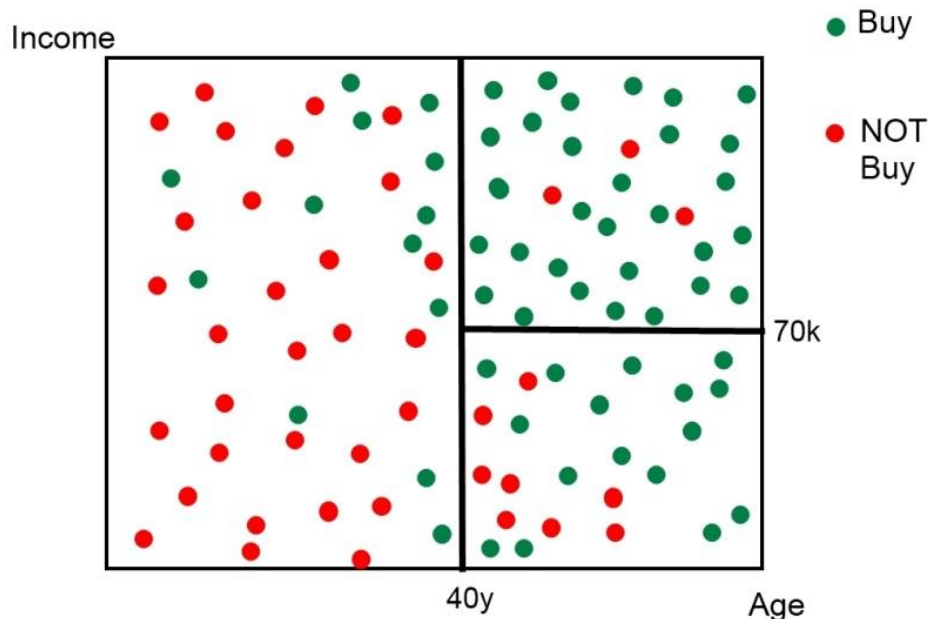
The Two Cultures of Modeling_(cont.)

- Breiman and some statisticians believe that it is questionable to characterize the nature's functions from the inputs to outputs with simple and unique models that make assumptions of data generation processes. They suggest that we should instead develop models that account for observed data well and generalize well to unseen test datasets.
- Now, such *Statistical Machine Learning* techniques are widely used and are proved successful in many different fields. However, some of these models & algorithms, such as Artificial Neural Networks and other ensemble learning techniques, are often complex and harder to interpret—just like the complexity of our worlds of interests. They are sometimes called "black-box" models.

The Two Cultures of Modeling_(cont.)

- There does have data scientists who are trying to [open the black box](#) (e.g. Rule Induction & Extraction), which is beyond the scope of this class. Still, how to choose modeling practices (cultures) depends on the goal of your data analytics projects. Again, a good analytics practice is to always focus on the goal and "see" the data from different angles (i.e. to identify better representations of features). Different representations of the data may capture hidden factors that could boost the model accuracy and interpretability.
- In many real-world data applications, you might find that simple models are easy-to-understand but come with lower accuracy and rough findings, whereas complex models are hard-to-interpret but have relatively higher predictive power. Note that, however, *there is no necessary connection between model accuracy and model interpretability/complexity.*

To Explain or to Predict?

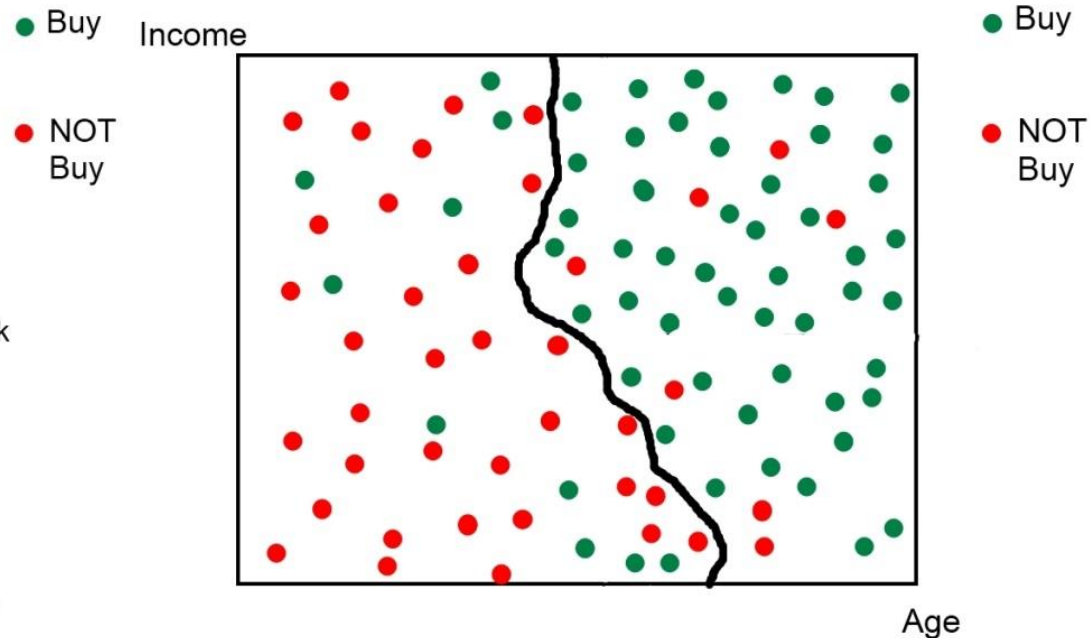


Misclassification Rate = 15% ??

Easy to understand the model
by these "rules":

- 1: IF Age >= 40y AND Income >= 70k
THEN Buy = True
- 2: IF Age >= 40y AND Income < 70k
THEN Buy = True
- 3: IF Age < 40y
THEN Buy = False

Simple!!



Misclassification Rate = 10% ?? (usually lower)

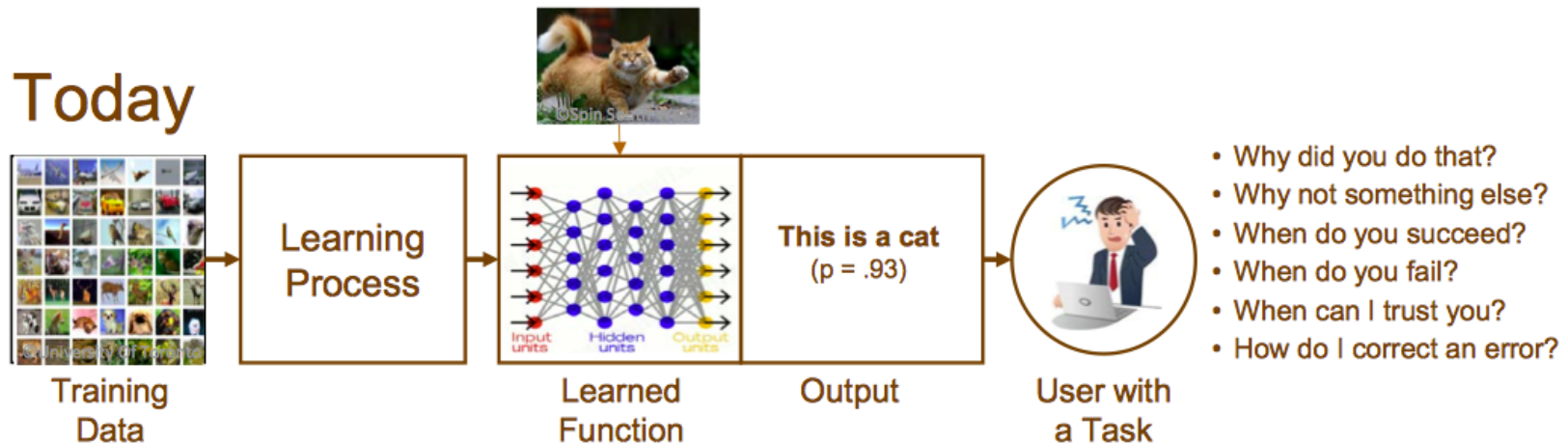
Rules ? What is that?

Obscure!!
Rule "extraction"?!

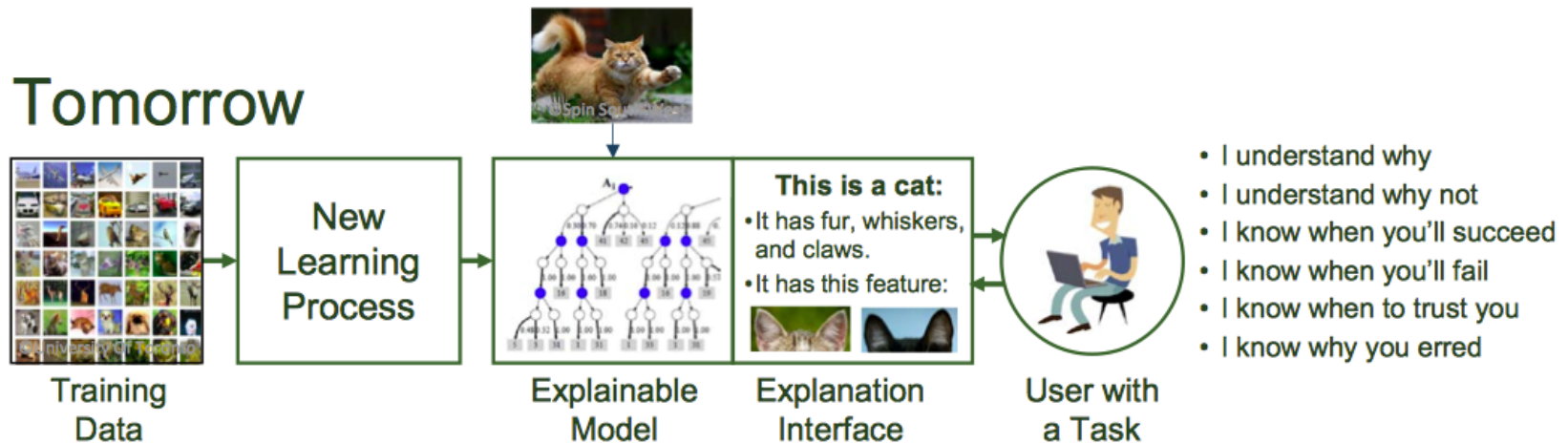
Also checkout article ["To Explain or to Predict?"](#)

Towards Explainable AI

Today

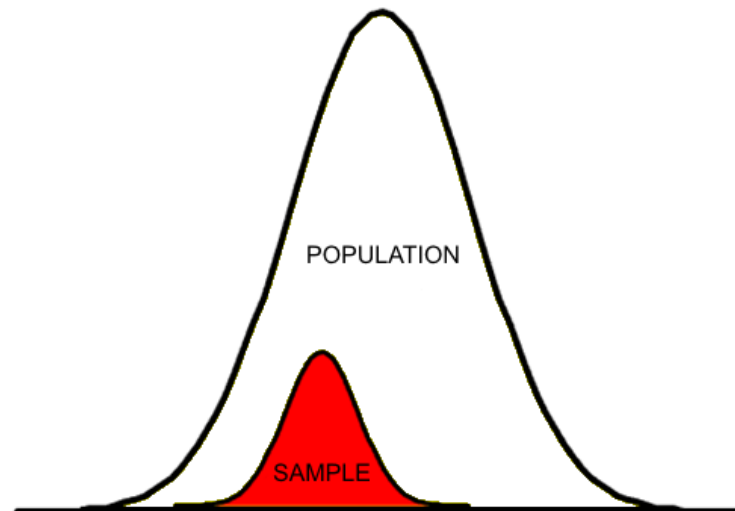
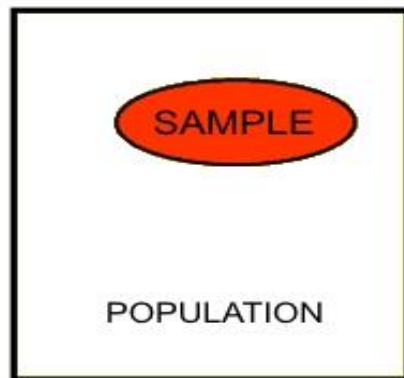


Tomorrow



Populations and Samples of Big Data

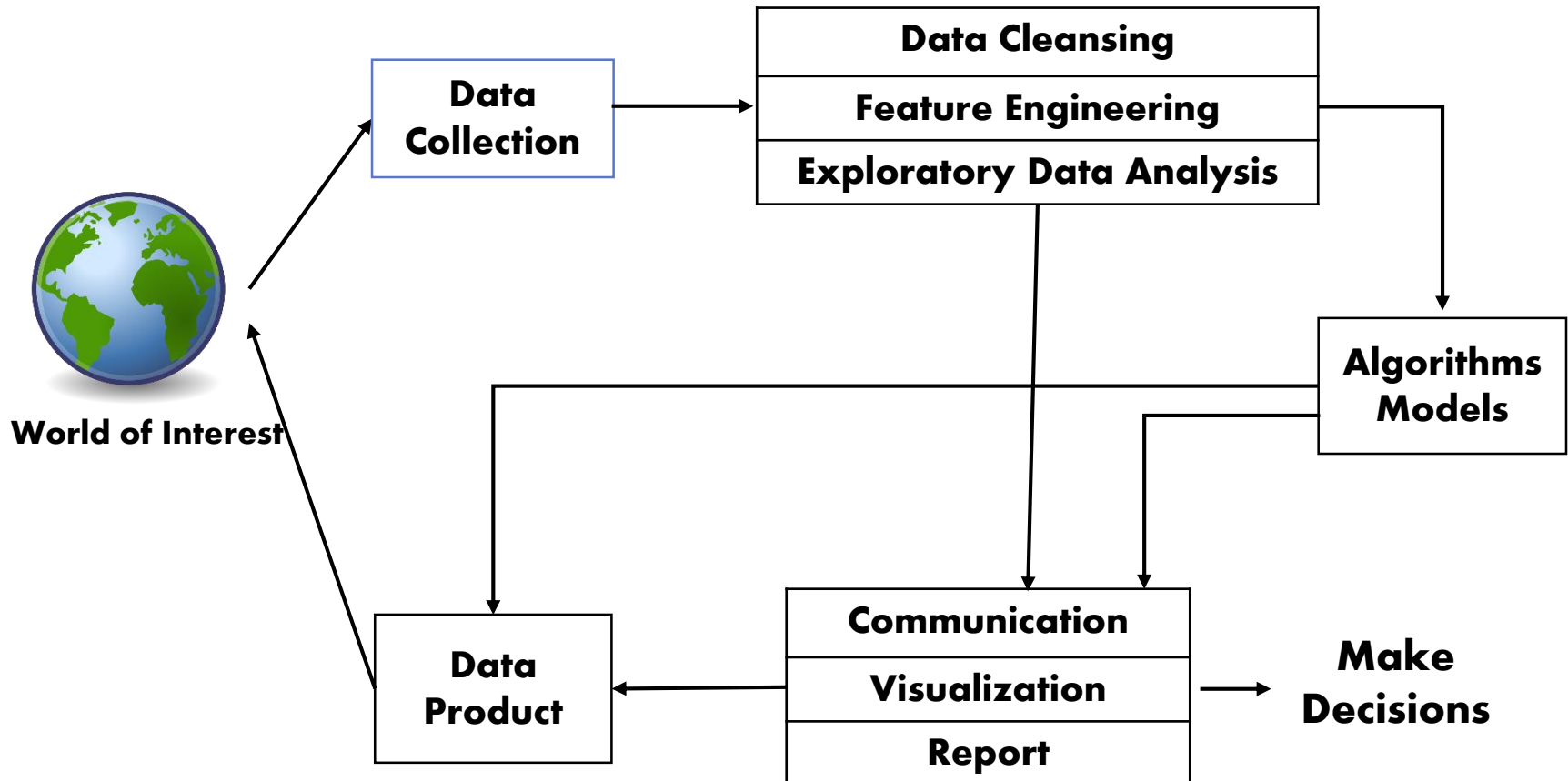
- "Big Data" doesn't mean the concepts of sampling is no longer important—simply, we would just have more information to understand the world.
- DO NOT believe those hype of Big Data that implies **N=ALL**. We would never observe everything (again, we're not the God!).



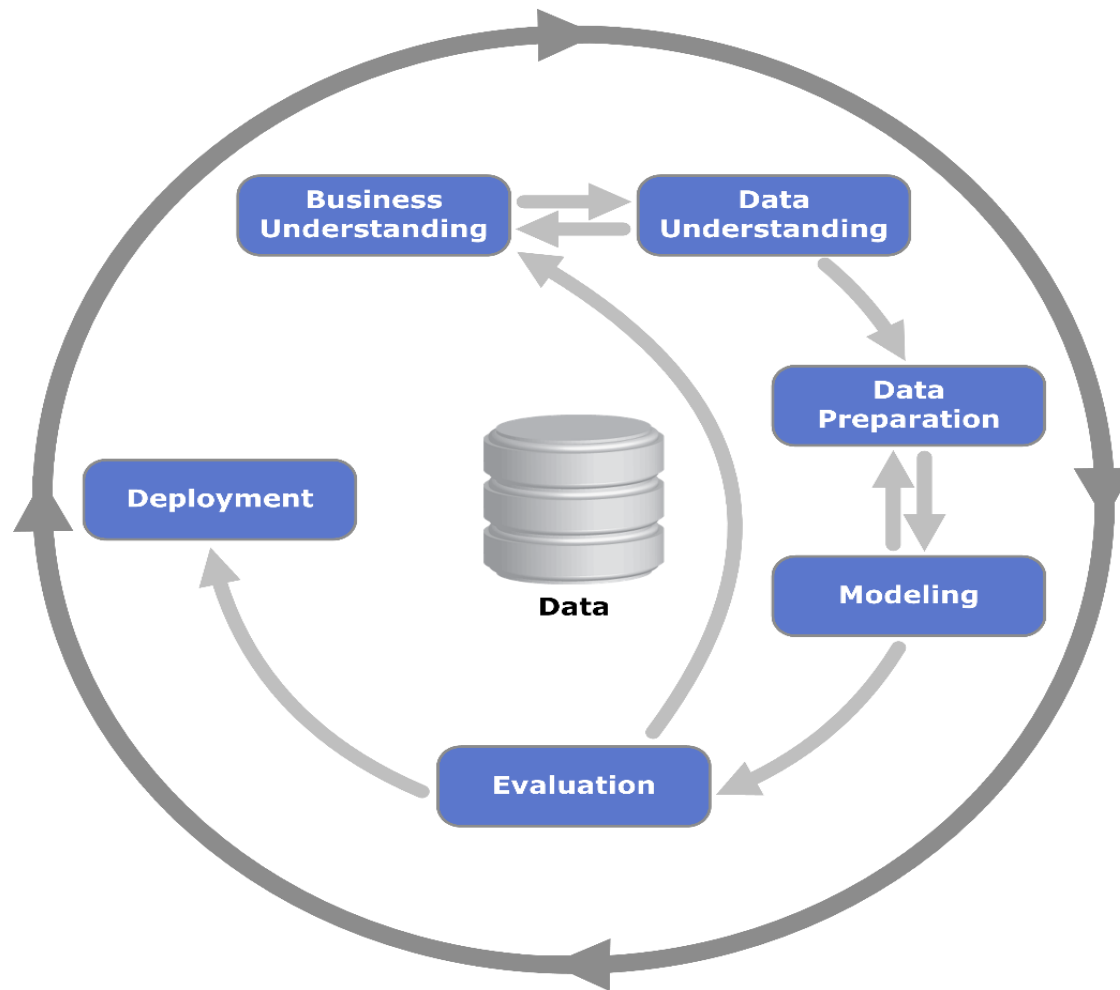
Populations and Samples of Big Data(cont.)

- Note that *biases are still there*, as we are making *assumptions* about our data (e.g. the underlying process that generated the data).
- Different interpretations of contexts may result in creating different models. Back in the days before the Big Data, we often look at the data, make assumptions, and then create parametric models for complex mechanism devised by the nature. Unfortunately, again, we would never find the perfect model, and conclusions drawn from our models remain questionable.
- Now, we have *more information* in the age of Big Data. When you analyze you data, use your imagination. Be careful and open-minded. Embrace different statistical learning techniques. And don't be afraid to make mistakes!

The Process of Data Analytics



Cross Industry Standard Process for Data Mining (CRISP-DM)



Source: https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png

Understanding our data

- Now, we are comfortable with daily data management tasks. However, *we still don't understand our data*. This baby step does matter! To get to know our data, we can compute some *descriptive statistics* and/or just plot the data!
- R is an expert in visualizing data. In addition to the built-in *base* plotting functions, package *lattice* and *ggplot2* allow you to produce high-quality & professional graphics.
- In this class, we will be using *ggplot2*, which features a [grammar of graphics](#) that enables us to manipulate and describe each component of a graphic.

Understanding our data_(cont.)

Univariate Analysis



Bivariate Analysis



Multivariate Analysis

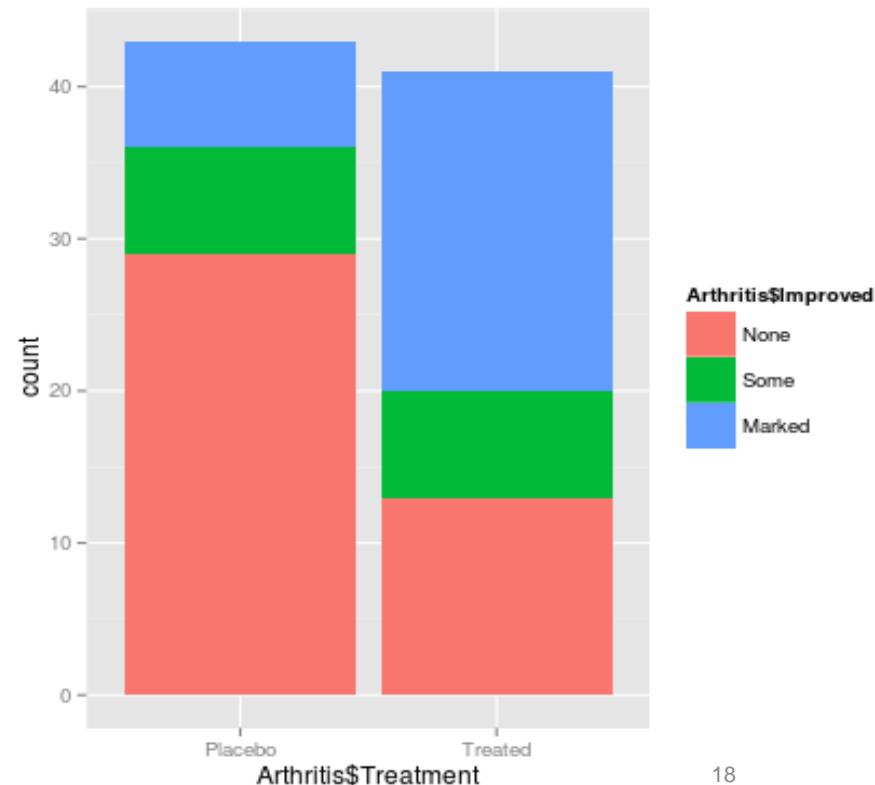
Understanding our data_(cont.)

- To deal with discrete random variables, we usually create frequency tables and compute marginal sums/proportions. Plotting *bar charts* or *histograms* may also help.

```
# package "vcd" for data "Arthritis"
library("vcd")
mytable = xtabs(~ Treatment + Improved,
data=Arthritis)
# marginal percentages
addmargins(prop.table(mytable))
```

	Improved			
Treatment	None	Some	Marked	Sum
Placebo	0.345	0.083	0.083	0.512
Treated	0.155	0.083	0.250	0.488
Sum	0.500	0.167	0.333	1.000

```
# Bar graph, Treatment * Improved
ggplot(Arthritis, aes(x =
Arthritis$Treatment,
fill=Arthritis$Improved)) +
  geom_bar()
```



Understanding our data_(cont.)

- For those of you who are familiar with PROC FREQ in SAS or CROSSTABS in SPSS, *gmodels::CrossTable()* should suit your need.

```
library(gmodels)
CrossTable(Arthritis$Treatment, Arthritis$Improved, chisq = T,
  fisher=T, format ="SAS")
```

Cell Contents		Arthritis\$Improved			
	N	None	Some	Marked	Row Total
Chi-square contribution		29	7	7	43
N / Row Total		2.616	0.004	3.752	
N / Col Total		0.674	0.163	0.163	0.512
N / Table Total		0.690	0.500	0.250	
		0.345	0.083	0.083	
Placebo					
		13	7	21	41
		2.744	0.004	3.935	
		0.317	0.171	0.512	0.488
		0.310	0.500	0.750	
		0.155	0.083	0.250	
Treated					
Column Total		42	14	28	84
		0.500	0.167	0.333	

Total Observations in Table: 84

Statistics for All Table Factors

Pearson's Chi-squared test

Chi^2 = 13 d.f. = 2 p = 0.0015

Fisher's Exact Test for Count Data

Alternative hypothesis: two.sided

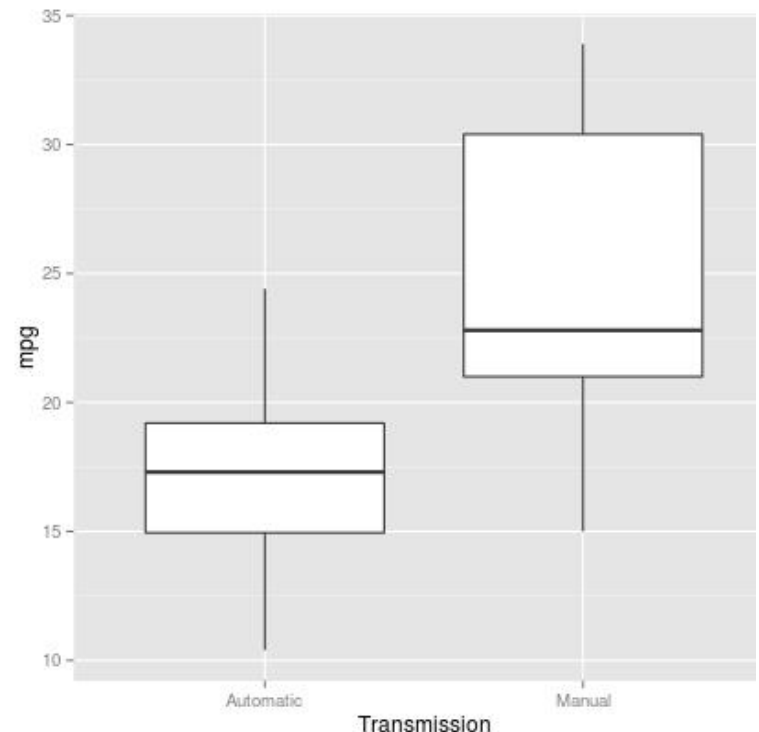
p = 0.0014

Understanding our data_(cont.)

- For continuous variables, the most intuitive way is to get five-number summary along with a *box plot*. Take the *mtcars* data again as an example:

```
# Get the five numbers for "mpg" of
# mtcars given different "am"
lapply(split(mtcars$mpg, mtcars$am), FUN = fivenum)
$`0`
[1] 10.4 14.9 17.3 19.2 24.4

$`1`
[1] 15.0 21.0 22.8 30.4 33.9
# Boxplot
qplot(factor(am, levels = c(0,1)), labels = c('Automatic', 'Manual'),
      mpg, data = mtcars, geom = "boxplot", xlab = "Transmission")
```

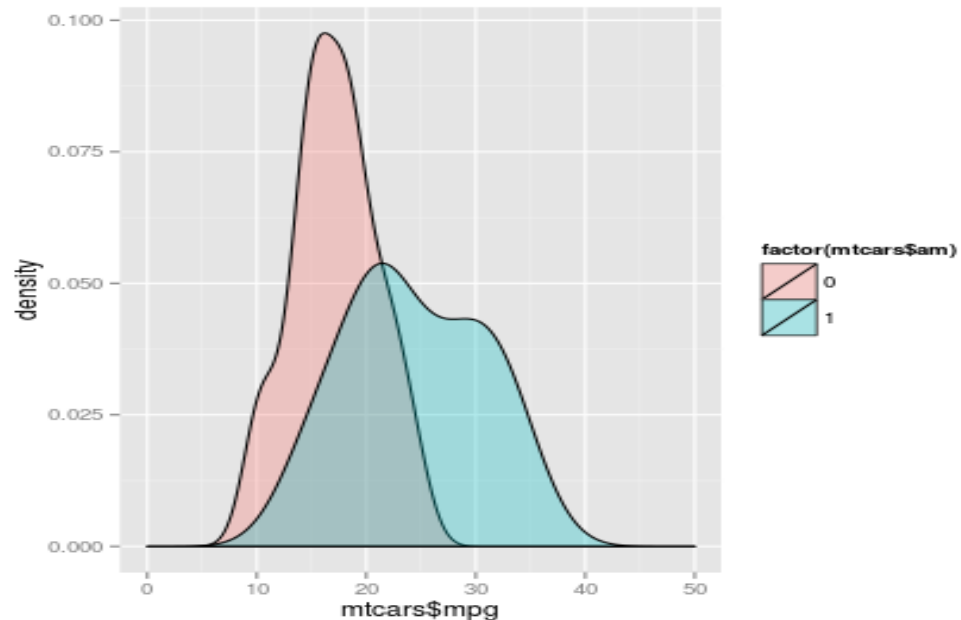


Understanding our data_(cont.)

- *Density plots*, on the other hand, may give you better visual senses of *probability distributions* of continuous variables. DO NOT make assumptions about the distributions of your data at the very beginning. Do some housekeeping, talk to domain experts, get related statistics, plot figures,..., try to explore a bit more first!
- If you have no clue to the distributions of your data and would like to know the [*Probability Density Functions*](#) (PDFs), check out function *density()* for a non-parametric way to estimate PDFs.

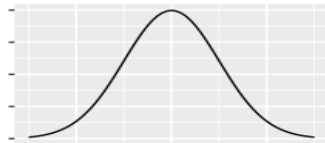
```
# Histogram & density curve for MPG
hist(mtcars$mpg, freq=F, breaks=10,
     col="red", xlab="MPG")
lines(density(mtcars$mpg),
     col="blue", lwd=2)

# By different "am"
ggplot(mtcars, aes(x = mtcars$mpg,
                  fill=factor(mtcars$am))) +
  geom_density(alpha=0.3) + xlim(0,50)
```

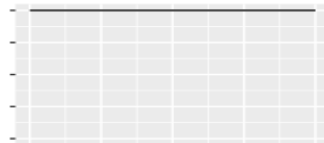


Probability Distributions

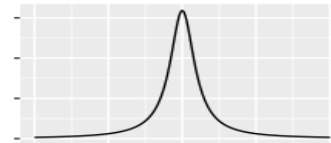
- Scientists observed real-world phenomenon, took measurements, and noticed that certain mathematical shapes kept reappearing. The classical example is the height of humans, following an approximately *normal (Gaussian) distribution*—a bell-shaped curve.
- Here is a list of probability distributions on continuous variables. Each of them has an PDF with parameters. The PDF can be considered a function with outcome (ranges) and probability as its input and output, respectively.



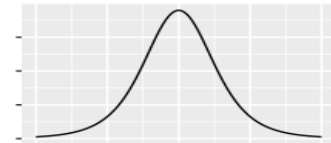
Normal Distribution



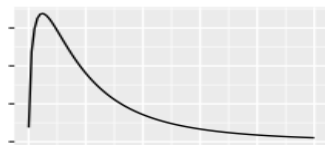
Uniform Distribution



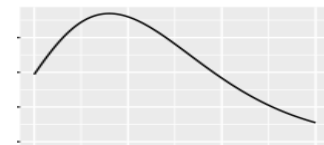
Cauchy Distribution



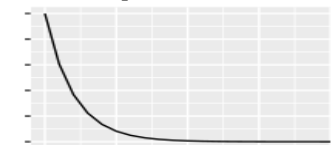
t Distribution



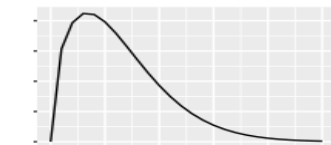
F Distribution



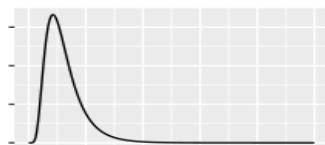
Chi-Square Distribution



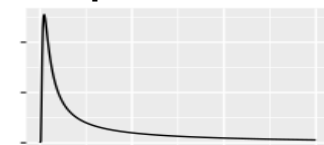
Exponential Distribution



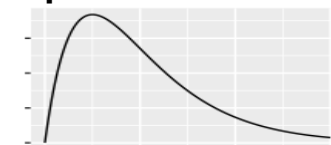
Weibull Distribution



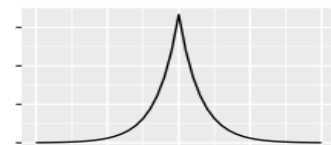
Lognormal Distribution



**Birnbaum-Saunders
(Fatigue Life) Distribution**



Gamma Distribution



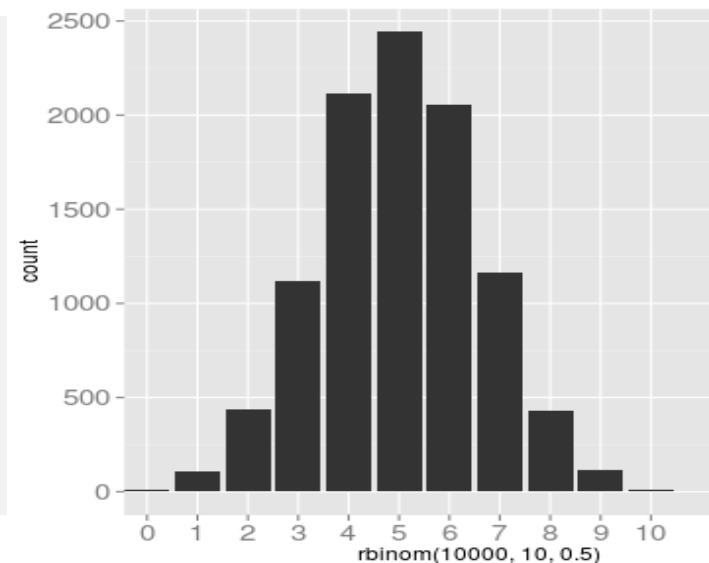
**Double Exponential²²
Distribution**

Probability Distributions_(cont.)

- For continuous distributions, the probability of a given exact outcome value (e.g. height = 170 cm) is almost equal to 0. So it might be easier to understand the concepts of the "density function" in discrete cases. For discrete variables, it's simply the probability of getting value x given parameter(s) θ , i.e. $P(X = x | \theta)$. Take the binomial distribution as an example, say we flip a fair coin 10 times. The probability of getting 5 heads $P(X = 5 | p = 0.5)$ should be around 0.246. We can simulate the result by generating binomial random numbers as we discussed in the class.

```
# Fair coin (p=0.5)
dbinom(x=5,size = 10,prob = 0.5)
0.246

# Histogram for binomial random numbers #
given p=0.5, size = 10
qplot(rbinom(10e3, 10, 0.5),
      geom = "histogram") +
  scale_x_continuous(breaks = 0:10) +
  theme(axis.text= element_text(size=15))
```



Probability Distributions_(cont.)

- In R, the probability functions take the form:

*[**d**;**p**;**q**;**r**]distribution_abbreviation()*

where the first letter refers to the aspect of the distribution returned: **d** = density function, **p** = cumulative distribution function, **q** = quantile function, **r** = random number generation.

Distribution	Abbreviation	Distribution	Abbreviation
Beta	<i>beta</i>	Logistic	<i>logis</i>
Binomial	<i>binom</i>	Multinomial	<i>multinom</i>
Cauchy	<i>cauchy</i>	Negative binomial	<i>nbinom</i>
Chi-squared	<i>chisq</i>	Normal	<i>norm</i>
Exponential	<i>exp</i>	Poisson	<i>pois</i>
F	<i>f</i>	Wilcoxon Signed Rank	<i>signrank</i>
Gamma	<i>gamma</i>	T	<i>t</i>
Geometric	<i>geom</i>	Uniform	<i>unif</i>
Hypergeometric	<i>hyper</i>	Weibull	<i>weibull</i>
Lognormal	<i>lnorm</i>	Wilcoxon Rank Sum	<i>wilcox</i>

Distributions? So What?

- Assuming specific parametric distributions on your data makes your life easier, as many parametric modeling techniques can be used.
- But, again, the world is *complex (adaptive, stochastic, and..., just not that simple!)* and your data is usually noisy.
- A good data analysis practice is to verify your assumptions on the data and also use other non-parametric techniques to justify your findings.

"Goodness of fit" test

- How can I know my sample follows a specific distribution? "Goodness of fit" tests may help. There are many such tests commonly used to check whether samples are drawn from the same distribution.
- Consider an example that we roll a dice for 100 times. All sample points are supposed to be equally-probable ($1/6$).

```
# Is it a fair dice?
pointFreq1 = c( 31, 22, 17, 13, 9, 8)
# Chi-squared goodness of fit test
chisq.test(x = pointFreq1, p = rep(1/6, 6))

# How about this one?
pointFreq2 = c( 17, 16, 15, 12, 19, 21)
chisq.test(x = pointFreq2, p = rep(1/6, 6))
```

"Goodness of fit" test_(cont.)

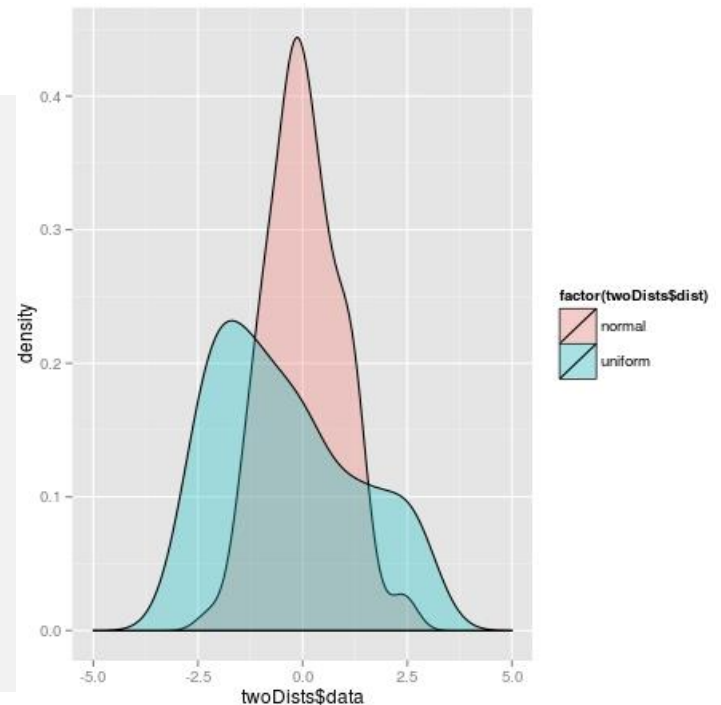
- Surely, there some tests for continuous distributions. Kolmogorov-Smirnov test (K-S test) is a popular nonparametric test of the equality of continuous distributions.

```
# normal vs uniform
twoDists = data.frame("data"= c(rnorm(100,0,1),
  runif(100, -3, 3)), "dist" = factor(
  c(rep("normal",100), rep("uniform",100)))) )

ggplot(twoDists, aes(x = twoDists$data,
  fill=factor(twoDists$dist) )) +
  geom_density(alpha=0.3) + xlim(-5,5)

twoDists_2c = split(twoDists$data,
  twoDists$dist)

ks.test(twoDists_2c$normal,twoDists_2c$uniform)
```



- Did you notice any problems of the K-S test?

A bit about "distances" among distributions

- In the case that we need a measure to evaluate "distances" among probability distributions, Kullback–Leibler divergence (D_{KL}) and k -ary Jensen–Shannon divergence (D_{JS}) might be good choices.
- D_{KL} is a non-symmetric measure, i.e. $D_{KL}(P_x, P_y) \neq D_{KL}(P_y, P_x)$, commonly used to compute difference between two probability distributions. D_{JS} , on the other hand, has been proved a better measure (in terms of its valuable properties, such as "symmetry"), as it could evaluate the information divergence among multiple probability distributions.

Test for normality

- Some data analysts tend to assume that the data follows a specific distribution, especially *normal distribution*. There does have some techniques that help you check the normality of your data. Back in the day, the most popular one is *Shapiro-Wilk test*.

```
# H0: sample is taken from a normal distribution  
shapiro.test(mtcars$mpg)  
Shapiro-Wilk normality test
```

```
data:  mtcars$mpg  
W = 0.9476, p-value = 0.1229
```

- Unfortunately, *every statistical hypothesis test has its limitations*. With large sample size ($n > 50$, as a rule-of-thumb), Shapiro-Wilk test is very easy to get low p-value from small deviations from normality.

Test for normality_(cont.)

- You may also use [*Lilliefors tests*](#) based on the K-S test, which is often considered a more powerful test for large sample.

```
nortest::lillie.test(faithful$waiting)
Lilliefors (Kolmogorov-Smirnov) normality test

data:  faithful$waiting
D = 0.1554, p-value < 2.2e-16
```

- Many data analysts believe that, according to [*Central Limit Theorem*](#), the violation of the normality assumption may not cause problems if we have large enough sample. However, we should not defend the normality of our data based on the results of these tests.

Think "Big"!

- So, what should we do to verify data distributions if we deal with big datasets?
 - ✓ Talk to domain experts (could be yourself) first. They may have better understanding of how the data is supposed to be distributed.
 - ✓ Still, run the distribution check tests (e.g. package *nortest* for more normality tests). Note that some of them are not reliable!
 - ✓ Plot your data. A density plot along with *normal quantile-quantile plot (q-q plot)* may give you the visual senses of the distribution and normality. Check out package for visualizing big dataset (e.g. *hexbin*, *tabplot* and *bigvis*). Also be aware that your eyes can be fooled by figures, especially the scales of features, too!

Try It!

- Load the dataset [DownloadFestival.dat](#). Create density and Q-Q plots for the variables, *day1*, *day2*, and *day3* (which are "Hygiene score" for people who were in the 3-day festival). Also please use the test functions in the package *nortest* to check the normality.

```
library(bigdataR); dlfest = bigDataR::DownloadFestival
# Create q-q plot for "day1"
qqnorm(dlfest$day1); qqline(dlfest$day1)

# Density plot for "day1"
qplot(dlfest$day1, geom = "density")
```

- Can we consider these scores "normal"? Also, can we say *day1* and *day2* scores of these people are significantly different? Use any statistical technique/test results to support your claim.







*"...if all you have is a hammer,
everything looks like a nail..."*

— *Abraham H. Maslow*

Big data as n observations by p variables

		Number of Variables (p)	
		Small p	Large p
Number of Observations(n)	Small n	<ul style="list-style-type: none"> ✗ Our models and inferences remain questionable. ✗ Our understanding of the world is limited by a few variables. 	<ul style="list-style-type: none"> ✓ Improve our ability to make inferences, as we can control for more variables and/or investigate more outcomes. ✗ How to select variables most relevant to the outcome? Are all variables required to make exogeneity plausible have been included?
	Large n	<ul style="list-style-type: none"> ✓ Increase the precision of estimates and the power of hypothesis tests. Many statistical learning techniques and/or estimation methods that require more observations could be used. ✗ We still cannot see the big picture of the worlds of interest. 	<ul style="list-style-type: none"> ✓ Creates numerous possibilities for descriptions, explorations, and inferences of the worlds of interest. ✗ More chance to get "coincident connections" among variables due to randomness. ✗ Harder to identify key predictors or factors.

Big data as n observations by p variables(cont.)

		Number of Variables (p)	
		Small p	Large p
Number of Observations(n)	Small n	 <p>Low accuracy and low precision</p>	 <p>Higher accuracy but low precision</p>
	Large n	 <p>Low accuracy but higher precision</p>	 <p>Higher accuracy and higher precision</p>

Relationships among variables

- The increasing availability of information has led to the analysis of datasets with very large *number of variables* (p) for each observation, instead of those datasets with large *number of observations* (n).
- When dealing with massive structured datasets, we tend to spend more time on discovering *relationships* (or *correlations*) among many variables, instead of finding *causations* within a smaller set of variables.
- Note that the phrase *correlation does not imply causation* still applies. We may even discover tons of correlations from massive datasets due to random coincidence.

Discovering bivariate relationships

- We have been using some statistical methods, such as Chi-squared test of independence and Fisher's exact test, to verify the relationships between two random variables.
- As discussed previously, many statistical methods can actually be represented as [predictive modeling](#) notations—as simple as both sides of an equation. Here we define:

$$[Y_1, Y_2, \dots Y_m; T_1, T_2, \dots T_n] = f([X_1, X_2, \dots X_i; A_1, A_2, \dots A_j])$$

where

Y: continuous output/dependent variable

T: discrete output/dependent variable

X: continuous input/independent variable

A: discrete input/independent variable

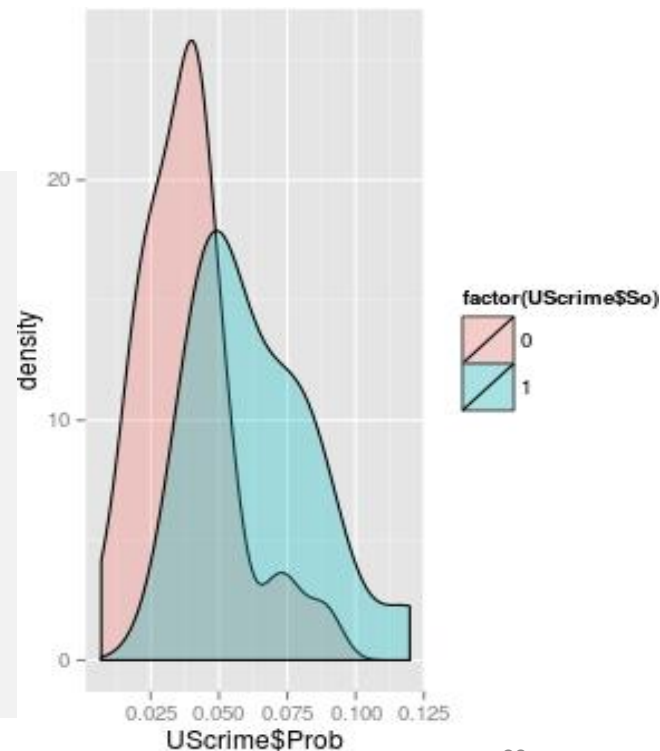
Discovering bivariate relationships(cont.)

- Take *2-sample t-test* as an example. It's simply the "relationship" between one continuous dependent variable Y versus one discrete independent variables A with *2 levels*, as:

$$Y = f(A)$$

```
# Consider built-in dataset MASS::USCrime"
ggplot(UScrime, aes(x = UScrime$Prob,
  fill=factor(UScrime$So))) +
  geom_density(alpha=0.3)

# We here compare "Southern and non-Southern
# states" (So) on the "probability of
# imprisonment" (Prob) without the
# assumption of equal variances of "So"
t.test(formula = Prob ~ So, data = UScrime )
```



Discovering bivariate relationships(cont.)

- In the case that your datasets don't seem to meet the parametric assumptions of t-test (i.e. normally-distributed outcome), nonparametric approach, [*Wilcoxon rank-sum test \(WRS\)*](#) may help.

```
# Normality tests
library("nortest")
Map(function(f) f(UScrime$Prob),
     c(shapiro.test, lillie.test, ad.test) )
# wilcoxon rank-sum test
wilcox.test(Prob ~ So, data = UScrime)
      wilcoxon rank sum test

data:  Prob by So
W = 81, p-value = 8.488e-05
alternative hypothesis: true location shift is not equal to 0
```

- Note that such nonparametric approaches do have assumptions. For example, we assume the dataset is *randomly-sampled* and *independently-observed*.

Discovering bivariate relationships(cont.)

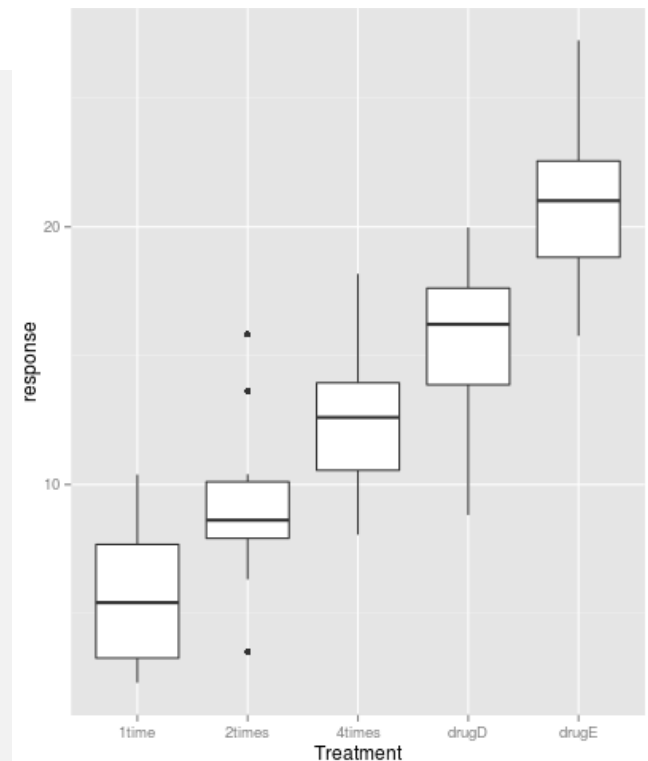
- To compare more than 2 groups (2 levels), we actually need to create a sort of "model". *ANalysis Of VAriance* (ANOVA) of [*General Linear Model*](#) might be the most popular parametric method (hammer!) used to describe such relationships.

```
library(multcomp) # Get "cholesterol" dataset
qplot(trt,response,data = cholesterol,geom =
"boxplot",xlab = "Treatment")
# H0: all group means are equal
cho_aov = aov(response ~ trt, data = cholesterol);
summary(cho_aov) # ANOVA table
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
trt	4	1351.4	337.8	32.43	9.82e-13 ***
Residuals	45	468.8	10.4		

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Equivalent to lm()
cho_lm = lm(response ~ trt, data = cholesterol);
car::Anova(cho_lm,type = 3)
# A trained model can also be used to "predict"
predict(cho_lm ,newdata =
  data.frame(trt=c("1time","drugD")))
```



Discovering bivariate relationships(cont.)

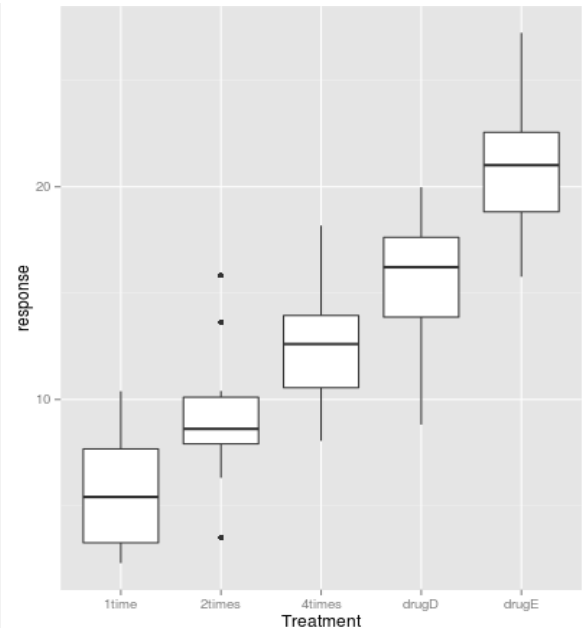
- You may want to know which treatment level is different from another. By default, the reference/baseline group is the first level of your treatment (factor) variable. You may change reference level by using *contrasts()* and *contr.treatment()*. Note that corrections (e.g. [Bonferroni](#)'s) on significant levels are required for further pairwise difference comparisons.

```
summary.lm(cho_aov)
```

```
...  
trt2times      3.443      1.443      2.385      0.0213 *  
trt4times      6.593      1.443      4.568 3.82e-05 ***  
trtdrugD       9.579      1.443      6.637 3.53e-08 ***  
trtdrugE      15.166      1.443     10.507 1.08e-13 ***  
...
```

```
cholesterol_c = cholesterol;  
#"4 times" as reference level  
contrasts(cholesterol_c$trt) = contr.treatment(5,base=3)  
cho_aov = aov(response ~ trt, data = cholesterol_c);  
summary.lm(cho_aov)
```

```
...  
trt1          -6.593      1.443     -4.568 3.82e-05 ***  
trt2          -3.150      1.443     -2.182 0.0344 *  
trt4           2.986      1.443      2.069 0.0443 *  
trt5           8.573      1.443      5.939 3.84e-07 ***  
...
```



Discovering bivariate relationships(cont.)

- And yes, there does have a "non-parametric version" of ANOVA—*Kruskal-Wallis rank sum test*.

```
# H0: medians of all groups are equal
kruskal.test(response ~ trt, data = cholesterol)

Kruskal-wallis rank sum test

data:  response by trt
Kruskal-wallis chi-squared = 36.5421, df = 4, p-value = 0.0000002238
```

- Note that it's null hypothesis is that all sample *medians* (instead of *means* in ANOVA) are equal. You may consider checking the result of your ANOVA with this test, if your outcome seems "not normal"!

Correlations

- Now, we know whether such bivariate relationships are "independent". So what? Consider analyzing the *correlations* among these variables may get your more information.
- A correlation between two variables is *between -1 and +1* indicating the direction (positive or negative) and strength of the relationship. The most common correlations are *Pearson's r* , *Spearman's ρ* , and *Kendall's τ* coefficients.
- Note that *correlation coefficients give no indication of the direction of causality!*

Correlations_(cont.)

- One assumption of using *Pearson's r* is that the sampling distributions has to be normally distributed.

```
# The pearson's correlations among mpg, hp, and wt
Hmisc::rcorr(as.matrix(mtcars[,c("mpg", "hp", "wt")]),
type = "pearson")
```

	mpg	hp	wt
mpg	1.00	-0.78	-0.87
hp	-0.78	1.00	0.66
wt	-0.87	0.66	1.00

n= 32

P

	mpg	hp	wt
mpg		0	0
hp	0		0
wt	0	0	

Correlations_(cont.)

- For ordinal variables, *Spearman's* ρ might be a better measure, as it compute the coefficient by using the *ranks* of data. This is also why outliers have less effect on the Spearman calculation.

```
# A strong positive correlation between x and y
x = c(3, 13,14,55,56); y = c(49,50, 66, 99,1000)
cor.test(x,y,method = "pearson")

...
sample estimates:
      cor 
0.650503 
cor.test(x,y,method = "spearman")

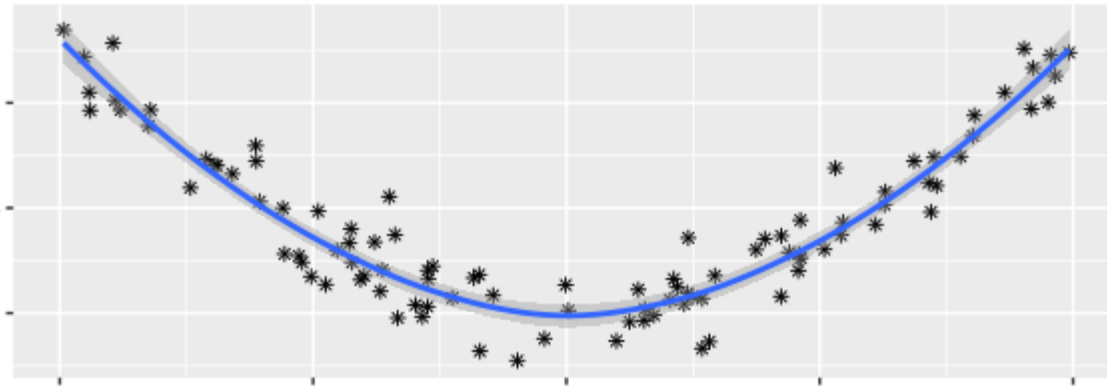
...
sample estimates:
      rho 
      1
```

- Again, check the results with non-parametric approach—the Kendall's τ .

```
# May not be able to compute p-value if sample size is big and there
# are some ties!
cor.test(x, y, method = "kendall")
```

Correlations_(cont.)

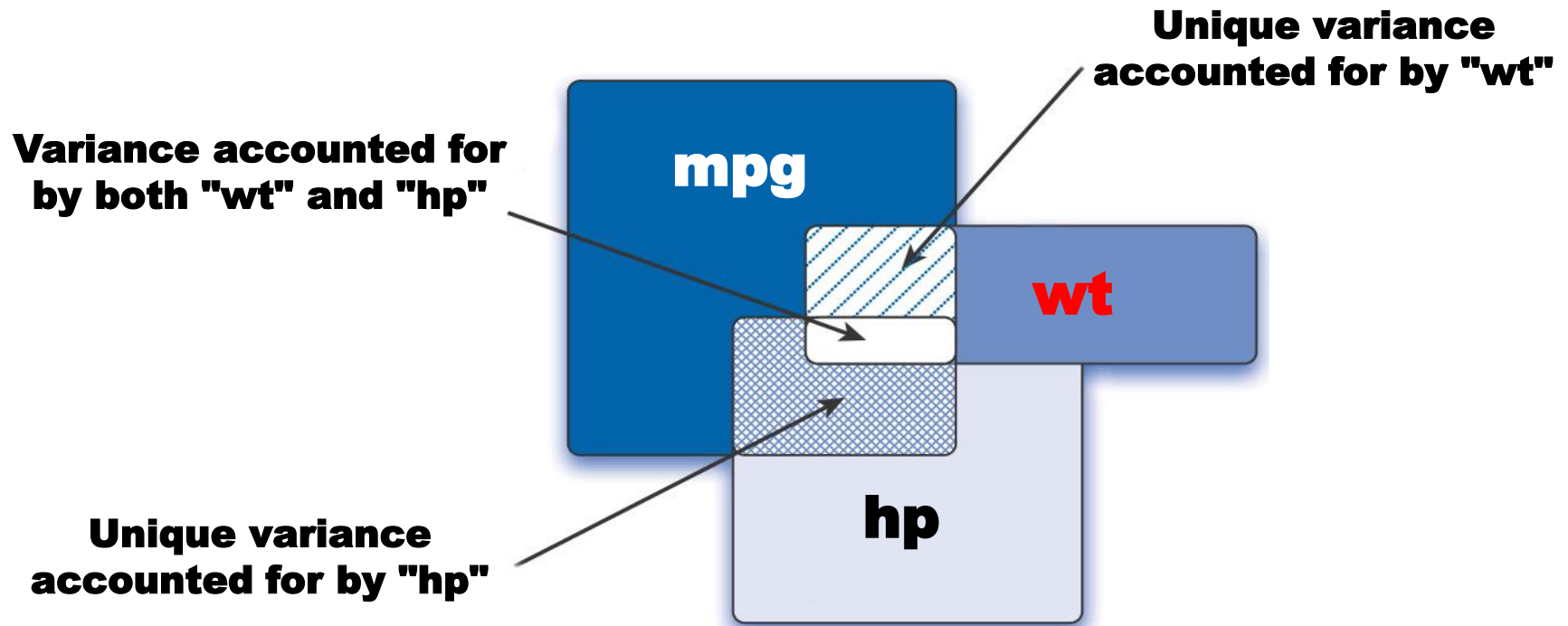
- Below variables have very low Pearson correlation coefficient. Why?



- A coefficient close to *zero* indicates that there is not a strong *linear* relationship, but it doesn't mean there's no relationship of any other kinds. For example, there's actually a *curvilinear* relationship between two variables in the above figure.
- A good data analysis practice is to *only consider correlation coefficient rough direction of the relationship*. Also, don't be obsessed by the p-values, as they depend on your sample size.

"Partial" correlations

- Unfortunately, the correlations may not necessarily that perfect as we believe. In the case that we'd like the "true" correlation between *mpg* and *hp* in *mtcars*, for example, we must consider that other variables (e.g. *wt*) might share part of their variations of the *mpg*.



"Partial" correlations_(cont.)

- So, to find out the size of the unique portion of variance of *hp*, we could use partial correlation to compute the correlation between *mpg* and *hp* while *controlling for* the effect of other variables (*wt* in this example).

```
# Correlation before controlling for wt
mpg_hp = cor(mtcars$mpg, mtcars$hp, method = "pearson")
mpg_hp ^ 2 # R squared (coefficient of determination)
0.602437

# Correlation after controlling for wt
mpg_hp.ct1.wt = ppcor::pcor.test(mtcars$mpg, mtcars$hp,
mtcars$wt, method = "pearson"); mpg_hp.ct1.wt
      estimate      p.value statistic  n gp Method
1 -0.546993 0.000433647  -3.51871 32  1  pearson

mpg_hp.ct1.wt$estimate ^ 2
0.299201
```


More about Model Thinking

- Previously, we consider bivariate relationships between two variables. But we also fit some simple linear models using *lm()* and calculated partial correlations with the help of R—we actually have been doing *multivariate data analysis*.
- Recall that we have predictive modeling notations used to represent models with multiple different types of outcome and predictors. In Statistical Machine Learning, we often say that we are dealing with *regression and classification problems* given that we have continuous/numeric and discrete/categorical outcomes, which means aforementioned linear models (e.g. ANOVA and t-test) are actually a sort of "regression".
- What is more interesting is that, *Logistic Regression* commonly used in *Categorical Data Analysis* is often used by data mining communities to classify data—to deal with "classification" problems!

More about Model Thinking_(cont.)

- Given the predictive modeling notations, we can then play the "modeling game".

- Two-sample t-test & One-way ANOVA? $Y = f(A)$

- Two-way ANOVA? $Y = f([A_1, A_2])$

- Multiple Linear Regression? $Y = f([X_1, X_2, \dots, X_i])$

- ANCOVA? $Y = f([X_1, X_2, \dots, X_i; A_1, A_2, \dots, A_j])$

- MANOVA? $[Y_1, Y_2, \dots, Y_m] = f([A_1, A_2, \dots, A_j])$

- MANCOVA & General Linear Model?

$$[Y_1, Y_2, \dots, Y_m] = f([X_1, X_2, \dots, X_i; A_1, A_2, \dots, A_j])$$

More about Model Thinking_(cont.)

- In addition to aforementioned typical statistical methods, the following statistical learning techniques are widely used by modern data scientists.

❑ Logistic Regression, Classification Tree, Support Vector Machine, k-Nearest Neighbors, Learning Vector Quantization Neural Networks...?

$$T = f([X_1, X_2, \dots X_i; A_1, A_2, \dots A_j])$$

❑ Regression Tree, Support Vector Regression, Bayesian Regression...?

$$Y = f([X_1, X_2, \dots X_i; A_1, A_2, \dots A_j])$$

❑ Probabilistic Graphic models, Random Forest, Generalized Linear Models...?

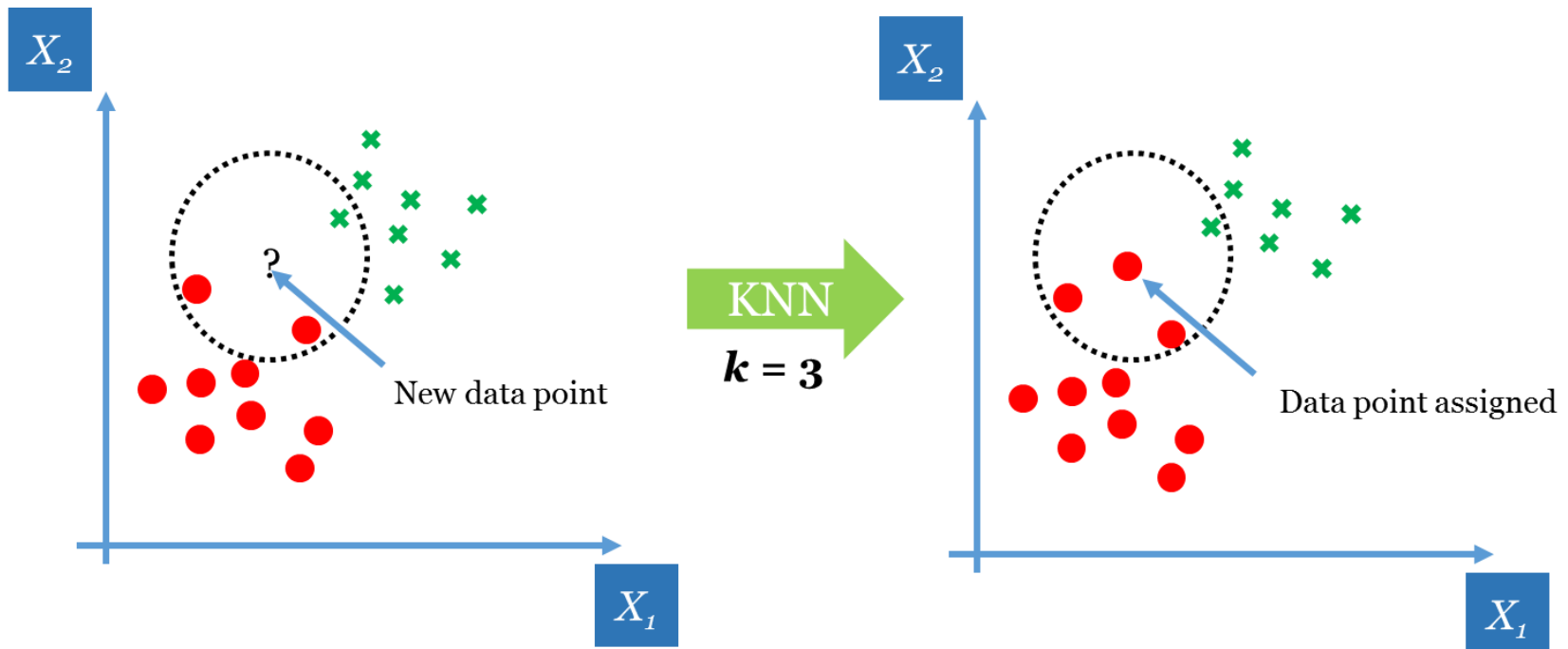
$$[Y_1, Y_2, \dots Y_m | T_1, T_2, \dots T_n] = f([X_1, X_2, \dots X_i; A_1, A_2, \dots A_j])$$

Feature Scaling & Transformation

- Some statistical tests and learning algorithms may be "fooled" by characteristics of features. For instance, those instance-based techniques, such as *k-Nearest Neighbors* (KNN) and *Principal Component Analysis* (PCA), are sensitive to the ranges of continuous features. They tend to give more weights to those features that exhibit larger variance and thus perform poorly.
- A quick, but not the best, solution is to make sure features are on the same (expected) scales, or just use scale-invariant techniques (e.g. tree-based models). Surely, R and most data analysis software allow us to create such data transformation/pre-processing "plans", which document how features would be re-scaled or transformed. Those plan objects can later be applied to training and testing (unseen) datasets to make sure all datasets are transformed into expected scales or ranges.

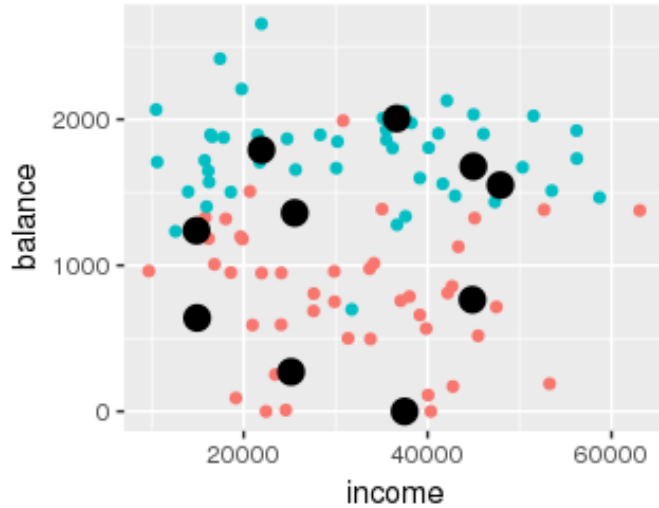
Feature Scaling & Transformation_(cont.)

- Consider using KNN to solve classification problems. As the "nearest neighbors" is usually defined based on Euclidean distance, features used in computing the distances (X_1 and X_2 in below example) must be on the same scale.

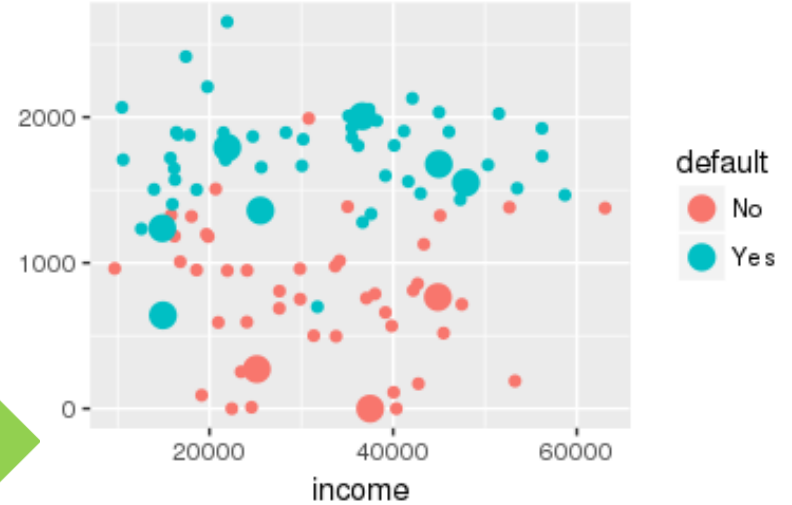


Feature Scaling & Transformation_(cont.)

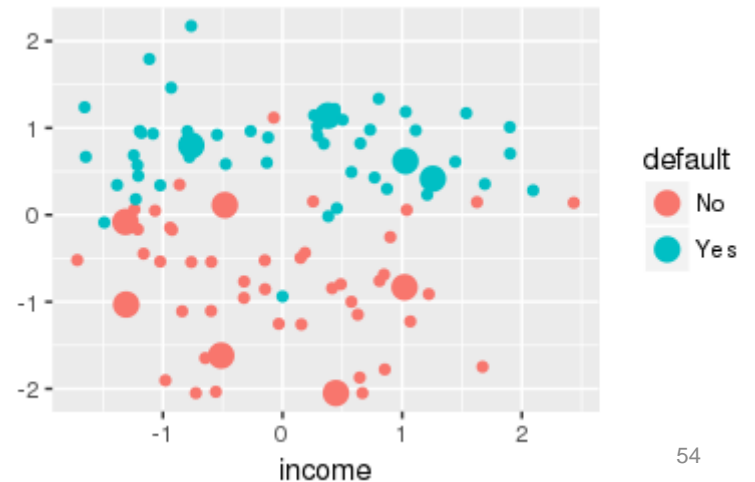
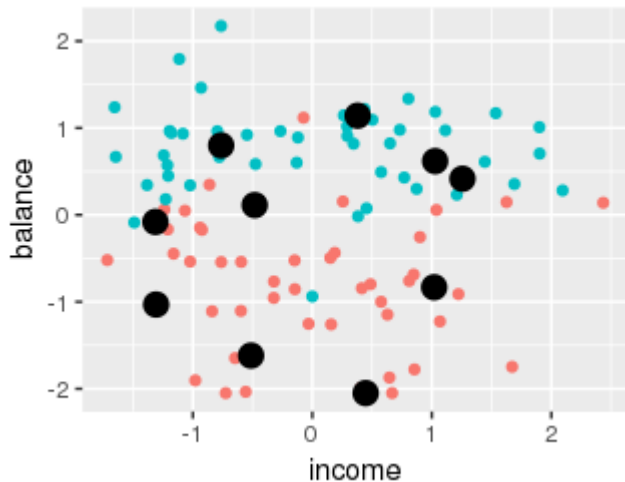
Unscaled



KNN
 $k = 3$



Scaled



Feature Scaling & Transformation_(cont.)

- Package [caret](#) provides a set of functions that help identify & apply transformations ("treatment") to continuous features. Below are common transformation methods in `caret::preProcess()`. Also note that those non-continuous features will be marked as "ignored".

Method	Description	Method	Description
<i>range</i>	Normalize values into the range of [0, 1]	<i>corr</i>	Remove highly correlated predictors.
<i>scale</i>	Divide values by its standard deviation	<i>center</i>	Subtract mean from values
<i>center, scale</i>	Standardize	<i>BoxCox</i>	Make features more "normal" by applying Box-Cox Transformation to them. Only work for non-negative values.
<i>pca</i>	transform data to the <i>principal components</i> .	<i>ica</i>	transform data to the <i>independent components</i> .
<i>zv</i>	remove predictors with a zero variance (all the same value).	<i>nzv</i>	remove predictors with a near zero variance (close to the same value).

Representations of Categorical Features

- For some statistical learning algorithms that cannot directly process discrete values (e.g. linear models), we often replace categorical features with *indicator variables* (*indicators*) that take the value 0 or 1 to indicate the absence or presence of some categorical effects that may shift the outcome.
- Different coding approaches are proposed to represent categorical features. The most popular is *dummy coding*, in which the reference/baseline group/level is assigned a value of 0 for each indicators, and the group of interest for comparison to the reference group is assigned a value of 1 for its specified indicator. In this case, $n-1$ indicators are required to represent a categorical variable with n levels.
- Also notice that we could manually merge/remove rare levels, or further "encode" categorical features by combining multiple variables/indicators. Ask domain experts and stakeholders and see if your encodings make senses!

Representations of Categorical Features(cont.)

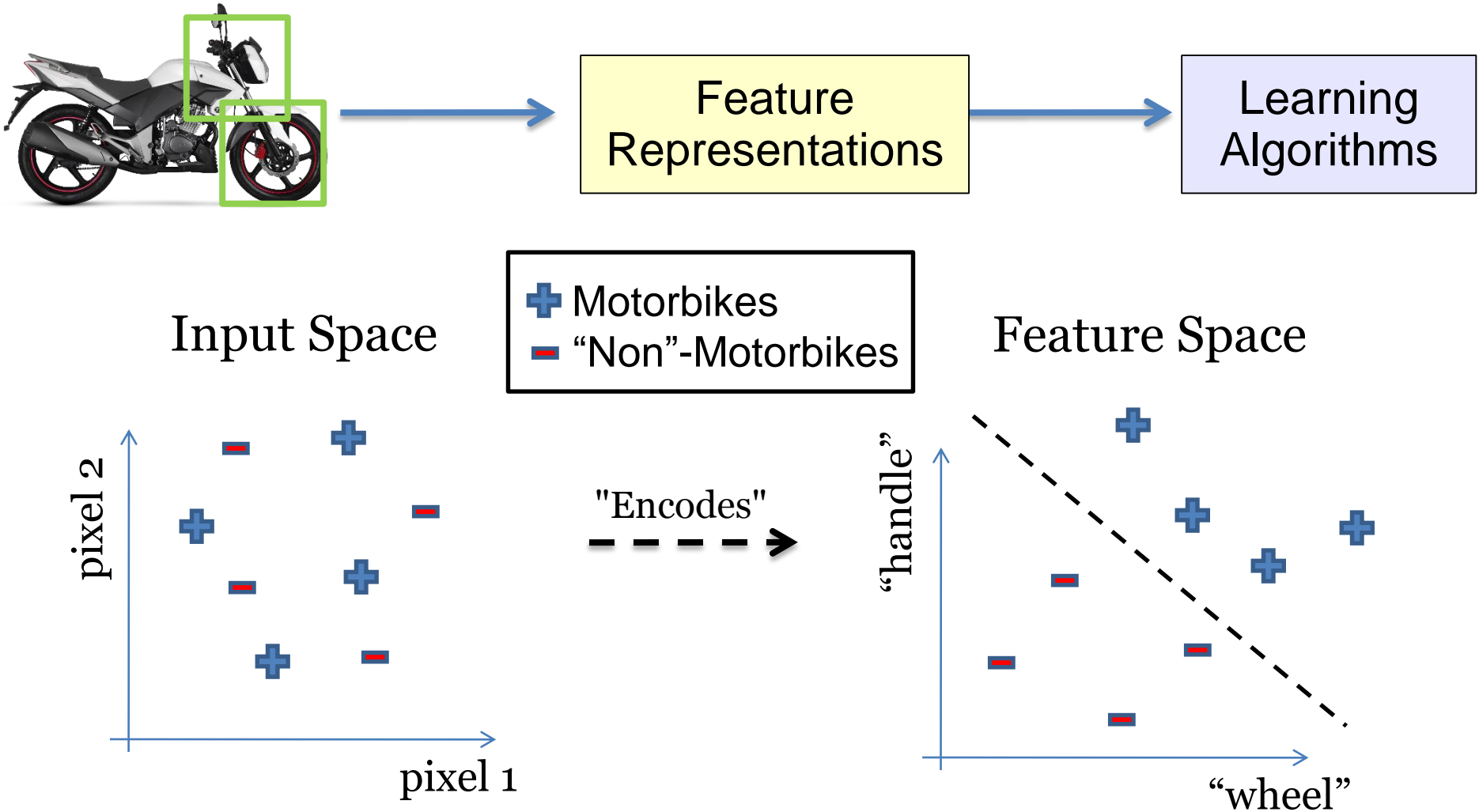
- We can surely create our own dummy coding/design matrix by using *model.matrix()* and *caret::dummyVars()*. Note that most of R model fitting functions have already done the coding for us.

```
# Dummy coding example
mt = mtcars; mt$cyl = factor(mt$cyl)
mt$am = factor(mt$am, levels = c(0, 1), labels = c("auto", "manual"))
# Checking the model matrix
model.matrix(mpg ~ am + cyl, data = mt)
# Coding matrix
contrasts(mt$cyl)
  6 8
4 0 0
6 1 0
8 0 1
# Change the reference group/level. The column names are the level values.
contrasts(mt$cyl) = contr.treatment(3, base = 2); contrasts(mt$cyl)
  1 3
4 1 0
6 0 0
8 0 1
model.matrix(mpg ~ am + cyl, data = mt)
# lm() will automatically do the coding for us
lm_mt = lm(mpg ~ am + cyl, data = mt, x = T); summary(lm_mt); lm_mt$x
```

Feature engineering is (still) the key

- Model learning with good features is always the key to the success of a data analytics project. Surely, we could further craft better features to improve model interpretability and prediction accuracy. Such tasks of feature selection, re-scaling, transformation, and encoding that create new features to boost model performance are called *Feature Engineering*.
- Note that feature engineering is not an one-shot task but an iterative process of modifying the data, identifying features, evaluate the results, and communicate with domain experts. Techniques are proposed to automate these processes (e.g. *best variable subset selection*), but feature engineering is still where the most of the effort in a data analytics project should go.
- Also, in addition to the simple re-scaling, good features are often created from some complex, linear or non-linear, transformations and combinations of features. For example, PCA is often used in capturing the maximal variance of continuous features, whereas ICA (Independent Component Analysis) is widely applied to the discovery of new independent features. We will soon talk about them later in this class.

Better Representations of Features



Parameter Estimation

- We have been fitting some *parametric models* and use them to infer, predict, and understand the phenomenon of interest. These models are based on assumptions that the data generations follow a certain "rules", i.e. *probability distributions*.
- These rules actually help us. Given parameters, we could infer the most probable outcome and even generate (random) outcomes that follow such rules. Take binomial distribution as an example:

```
# Binomial distribution given p(Head) = 0.3
dbinom(0:10,size = 10,prob = 0.3)
[1] 0.02825 0.12106 0.23347 0.26683 0.20012 0.10292 0.03676 0.00900
0.00145 0.00014 0.00001
```

```
# Generate outcomes that follow the binomial
# distribution with parameter  $\theta$ , p(Head) = 0.2
set.seed(1); x = rbinom(100,10,0.2); x
[1] 1 1 2 4 1 4 4 2 2 0 ...
```

Parameter Estimation_(cont.)

- In this case, we assume the distribution and parameter θ . We'd like to know the probabilities of the outcome \mathbf{x} . That is,

$$P(\mathbf{x} \mid \theta)$$

- Unfortunately, in real-world applications, we often don't know the actual θ . But, we can do it in the other way around—to guess" ("estimate", more appropriately) the most probable θ (i.e. $\hat{\theta}$), given the outcome \mathbf{x} . Here we introduce *likelihood functions* (instead of the above probability density/mass functions):

$$L(\hat{\theta} \mid \mathbf{x})$$

Parameter Estimation_(cont.)

- So, consider an example that we are flipping a coin. We don't know whether it's fair—the probability of getting a "head" should be around 0.5 , i.e. $P(H) = 0.5$. However, we can estimate the parameter θ ($P(H)$) by using the likelihood function L with the outcome \mathbf{x} —what is the most probable θ that maximize the likelihood function L ?

```
# Let's reuse previous "x" as that outcome for simulation
set.seed(1); x = rbinom(100,10,0.2)
# Function used to calculate the log-likelihood
LL_binom = function(p){
  ll = dbinom(x,10,p); return(-sum(log(ll)));
}
# Maximum likelihood estimate of the parameter, 0 <= P(H) <= 1
# The goal is to find a P(H) that maximize likelihood (minimize
# negative log-likelihood)
# One dimension root-finding using Brent's method
optim(par = 0.5, fn = LL_binom, method = "Brent", lower=0, upper=1)
$par
[1] 0.204
...
```

Parameter Estimation_(cont.)

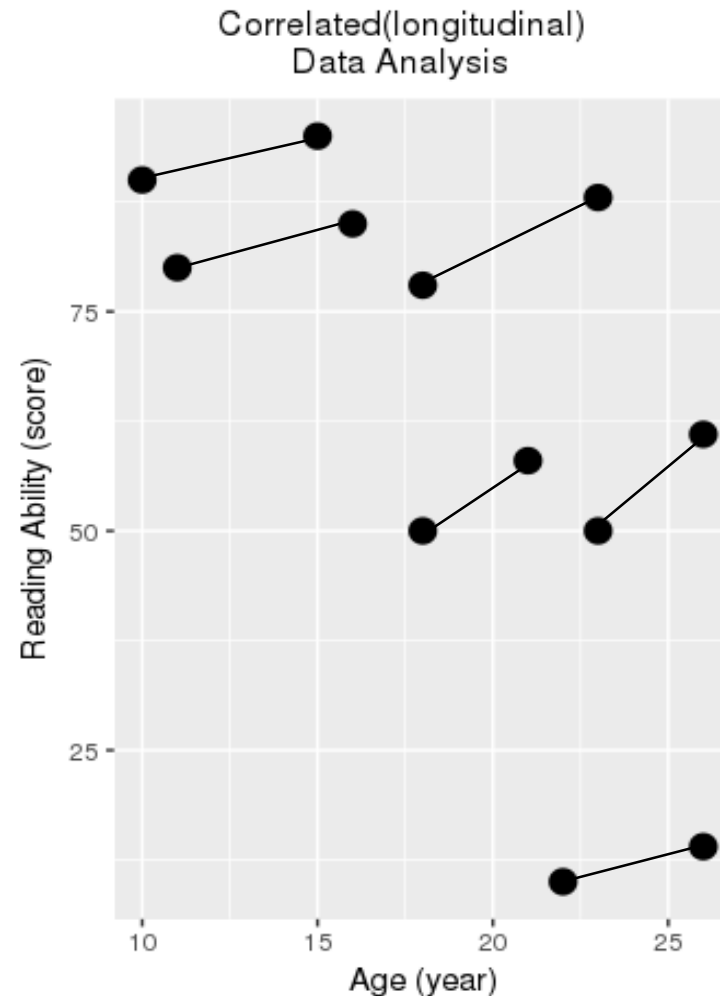
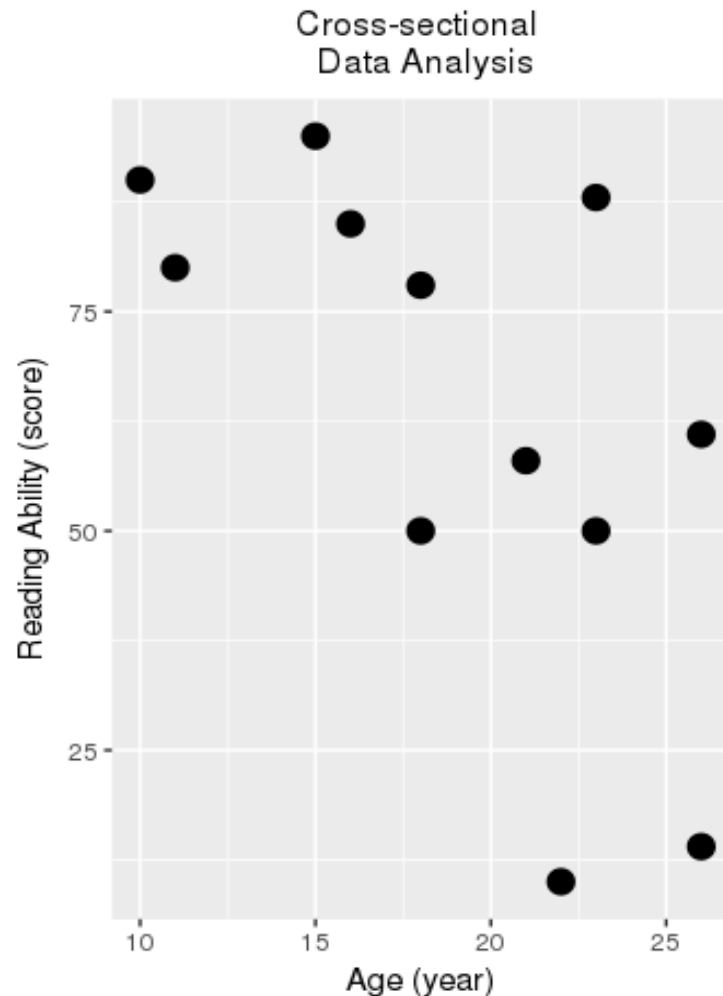
- The aforementioned technique, *Maximum Likelihood Estimation (MLE)*, has been widely used to estimate model parameters in Statistics world for years. Let's consider another example to estimate parameters of a simple linear model.

```
# lm() has estimated the coefficients of below model
lm(data = mtcars, formula= mpg ~ wt)

...
Coefficients:
(Intercept)          wt
      37.285      -5.344
# We can do it by using MLE instead!
LL_LM = function(beta0, beta1) {
  resid = mtcars$mpg - (mtcars$wt * beta1) - beta0; # residuals
  ll = dnorm(resid, 0, 1); # likelihood for the residuals
  return(-sum(log(ll))); #sum of the log likelihood
}
stats4::mle(LL_LM, start=list(beta0 = 0, beta1 = 0), method =
"Nelder-Mead")

...
Coefficients:
beta0      beta1
37.275657 -5.342921
```

A bit about Correlated Data Analysis



A bit about Correlated Data Analysis(cont.)

- *Correlated data* refers to a dataset with correlated response/outcome—observations are simply not independent. Consider previous dataset, for example. The observations are actually "paired" (or case-dependent), i.e. the observations were drawn by the same subjects in different time points (or conditions).
- Let's say we would like to know the relationship between the *reading ability* and the *age*. We may simply create a linear model or do the correlation analysis if we don't care such "dependences" among observations.

```
read_age = data.frame(  
  subjID= rep(1:6, each=2),  
  read_ability = c(90, 95, 80, 85, 78, 88, 50, 58, 50, 61, 10, 14),  
  age = c(10, 15, 11, 16, 18, 23, 18, 21, 23, 26, 22, 26 ),  
  gender = factor(rep(c('F','M','F','F','F','M'),each = 2), levels=c('F','M'))  
)
```

```
summary(lm(read_ability ~ age, read_age)) # simple linear regression
```

```
...
```

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	127.075	25.945	4.898	0.000625	***
age	3.345	1.313	-2.547	0.029007	*

```
...
```

```
# A negative(??) association between reading ability and the age...
```

A bit about Correlated Data Analysis(cont.)

- We may be told by domain experts saying that a young individual's reading ability should improve significantly as he or she grows older. Our analysis result tells us the opposite story?!
- Let's go back to the very basic idea. Are there any difference between these young people's reading abilities in two different time points? One quick-and-dirty solution is to test whether the "gap" of the two reading ability scores significantly different from 0.

```
# Replace the age with new variable "time point"
read_age$timepoint = rep(c("timepoint1","timepoint2"),6)
read_age_w = dcast(read_age, subjID ~ timepoint, value.var = "read_ability")
read_age_w$gap = read_age_w$timepoint1 - read_age_w$timepoint2

wilcox.test(read_age_w$gap, mu = 0)

wilcoxon signed rank test with continuity correction

data:  read_age_w$gap
V = 0, p-value = 0.03552
alternative hypothesis: true location is not equal to 0
```

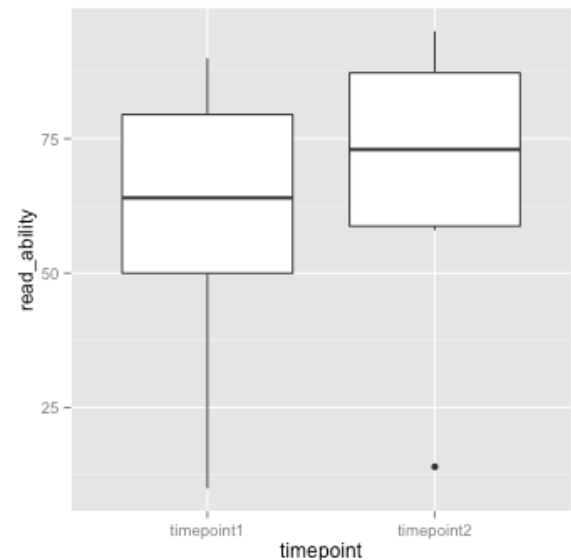
A bit about Correlated Data Analysis(cont.)

- Or, more formally, we could use some tests to compare the sample means/medians of the outcome (reading ability) in two (and only two) time points—the "paired" version of previous t-test and Wilcoxon signed rank test.

```
qplot(timepoint, read_ability, data=read_age,  
geom="boxplot" )  
  
# Equivalent to previous test on the "gap"  
wilcox.test(read_age$read_ability ~  
read_age$timepoint, paired=T)  
# Paired t-test also can do the trick, too.  
t.test(read_age$read_ability ~ read_age$timepoint,  
paired=T)
```

Paired t-test

```
data: read_age$read_ability by read_age$timepoint  
t = -5.9977, df = 5, p-value = 0.001849  
...
```



- Based the results, we might say that subjects' reading abilities should improve when subjects grow older, as suggested by the experts. However, we should simply consider it deserves further research.

A bit about Correlated Data Analysis(cont.)

- Depending on applications and fields, literatures name such correlated data in different names—longitudinal data, clustered data, multi-level data, repeated-measurements..., etc.
- In Statistics world, the most common technique used to cope with such datasets is called *Mixed Models*—modeling technique that takes both *fixed-effect* and *random-effect* terms together within the model. For simplicity, you may consider time-varying (e.g. age) and non-time-varying (gender) independent variables as the random effects and fixed effects.

```
# Linear mixed-effect model  
library("lme4")  
summary(lmer(read_ability ~ age + gender + ( 1| subjID), read_age)
```

- Remember to check out package *nlme* and *lme4* if you have such datasets.

P-values, the ~~hunger~~ hunting! games



Source: <http://www.peaya.com/peaya.php?comicsid=1013>

peaya.com

hunting! *P-values, the ~~hunger~~ games*(cont.)

- A [recent article](#) about the p-values reports that "*...half of all published psychology papers that use null-hypothesis significance testing contained at least one p-value that was inconsistent with its test statistic and degrees of freedom. One in eight papers contained a grossly inconsistent p-value that may have affected the statistical conclusion...*".
- The p-values has been misused for years. The most notorious issue is that it has become a criterion for whether work is publishable. Some researchers have been hunting for "p-values < 0.05" so that they can claim their findings are "statistically significant". It has even worsen in the age of Big Data, as we can make the p-values as small as we want by increasing the sample size and/or by using statistical tests sensitive to trivially small effects.
- Although the p-values should be considered outdated, it does not mean that we should skip p-values or shouldn't have large samples. Instead, when we interpret our findings, we must understand that the significance depends on the effect size and test sensitivity, and the "statistically significant" may not be meaningful. Also remember to check out [ASA Statement on Statistical Significance and P-values](#).

ASA Statement on Statistical Significance and P-values

- 1) "P-values can indicate how incompatible the data are with a specified statistical model."
- 2) "P-values do not measure the probability that the studied hypothesis is true, or the probability that the data were produced by random chance alone."
- 3) "Scientific conclusions and business or policy decisions should not be based only on whether a p-value passes a specific threshold. "
- 4) "Proper inference requires full reporting and transparency."
- 5) "A p-value, or statistical significance, does not measure the size of an effect or the importance of a result."
- 6) "By itself, a p-value does not provide a good measure of evidence regarding a model or hypothesis."

P-values in the age of Big Data

- So, if the p-values is "outdated", how to evaluate our models? My answer to it is that it simply means using p-values cautiously. Very often, we get almost everything statistically significant when learning models from massive datasets. Instead of just using p-values to select and evaluate models, we should also consider the accuracy of prediction (e.g. accuracy for classification problems and mean squared error for regression problems), variable importance (different learning algorithms have different importance evaluations), and model interpretability.
- Researchers have also proposed other possible solutions. For example, we can simplify models by finding sparse representations of the input data; learn simple [*surrogate models*](#) (e.g. linear models) with the outcomes from complex models; impose some regularizations/constraints on model learning may also help; or, just use those "big data-ready" learning algorithms, such as [*Elastic Net*](#) instead of typical generalized linear models.
- Don't worry about it for now. We will soon discuss these approaches. The data analytics is about learning from data. And [*we must build tools for massive datasets*](#). Again, getting "p-value < 0.05" simply means it's worth of a second look. It is just the beginning of your research, not the goal of it.

Final comments

- In this unit, we have a quick review on data analytics process, basic probability theory, and a few popular statistical methods with help of R. We understand how to do basic statistical analyses and simple statistical inferences. However, as we have discussed, these techniques have their own limitations, and *statistics cannot prove anything with certainty*.
- We also have discussed the role of Statistics in the age of Big Data. The big data create opportunities but brings challenges, too. We may create more precise and accurate models used to explain the world of interests, but we could get lost in the mist of the big data and make terrible mistakes.
- However, don't be afraid to make mistakes. Be open-minded. And ready to embrace changes and acquire new "hammers". In the following units, we discuss sophisticated big data tools and techniques that help you "clear the mist"! 😊