

Hadoop 기반 빅데이터 통계분석

김 승 환

swkim4610@inha.ac.kr

강사 소개



김 승 환 교수

- 인하대학교 통계학 박사(1997)
- 현, 인하대학교 공과대학
소프트웨어융합공학 연계전공
연구교수
- 현, 정보화진흥원 개인정보 비식별화
전문위원

[주요 경력 소개]

[경력]

- 1998.6 ~ 2000.4 나이스컨설팅 수석연구원
- 2000.5 ~ 2005.6 SK 에너지 CRM팀 부장
 - 엔크린 보너스 카드 고객분석 및 CRM 수행
- 2005.7 ~ 2007.12 SK 네트워크 고객사업부문 팀장
- 2008.1 ~ 2014.3 SK 마케팅앰컴퍼니 K-Hub 그룹장
 - 오케이캐쉬백 고객정보 관리 및 마케팅 총괄
 - 오케이캐쉬백 고객정보보호 책임자

[전문 분야]

- Statistical Learning, Deep Learning
- 개인정보 비식별화
- 빅데이터 시스템 개발

목차 및 학습 필요사항

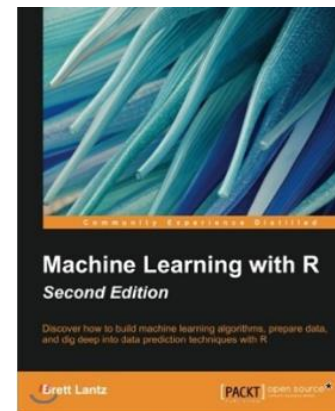
1. 빅데이터와 스몰데이터
2. Hadoop
3. R
4. RHadoop
5. 빅데이터 통계모형
6. 실전 분석

참고서적

본 강의에는 우측과 같은 책이
참조 되었습니다.

소스 및 데이터

<http://naver.me/xJn0Djcc>



1. 빅데이터와 스몰데이터

- 1.1 빅데이터 시대
- 1.2 분산처리
- 1.3 빅데이터 속성
- 1.4 정형과 비정형
- 1.5 데이터웨어하우스
- 1.6 Static과 Event Log Data
- 1.7 데이터 처리 변화
- 1.8 통계학
- 1.9 통계학과 빅데이터 접근 차이

1. 1 빅데이터 시대

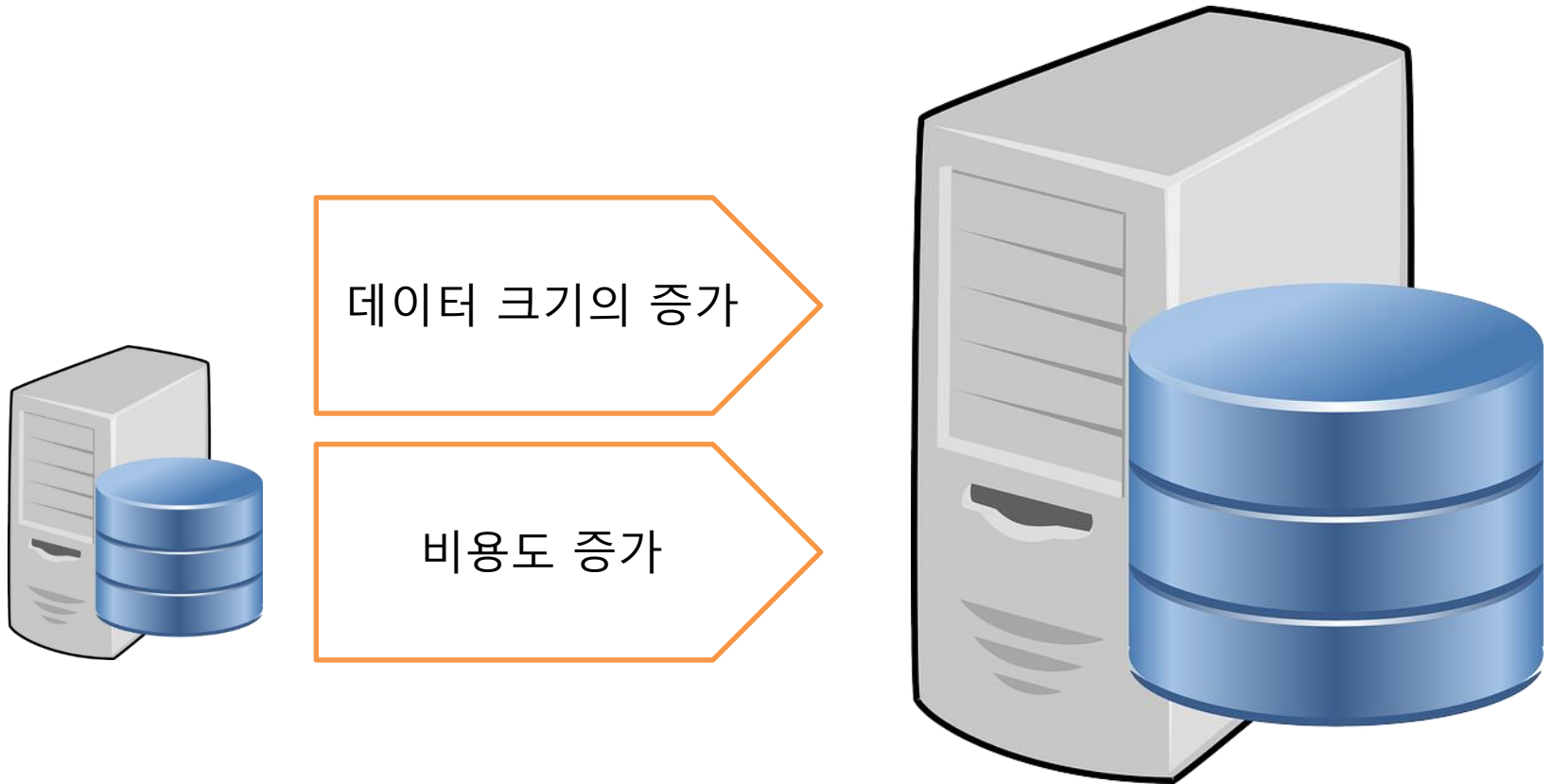
우리는 이미 상당량의 디지털 데이터를 발생시키는 주체이고 이용하는 주체이다.
이른바, 빅데이터 세상에 살고 있다.



- 뉴욕증권거래소 : 1일에 1테라 바이트의 거래 데이터 생성
- Facebook : 100억장의 사진, 수 페타바이트의 스토리지
- 통신사 : 시간당 10G 이상의 통화 데이터, 1일240G 생성, 월 생성 데이터의 크기 200T 이상

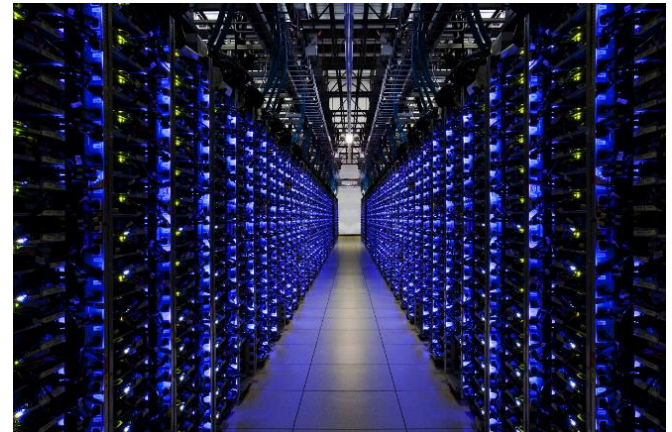
1. 1 빅데이터 시대

- ✓ 과거에는 발생하는 데이터 중 상당 부분을 버리거나 요약된 형태로 저장 관리했음
- ✓ 검색서비스, 유튜브 등이 나오면서 데이터가 급격히 증가



1.2 분산처리

- ✓ 이에 대한 해결책으로 ... → 분산 처리 Computing 방법론 탄생
- ✓ 거대한 디스크를 가진 컴퓨터가 아닌 PC 계열 리눅스 OS 탑재한 컴퓨터를 병렬로 연결
- ✓ 컴퓨터 군단을 병렬로 제어하는 구글 파일 시스템 탄생
- ✓ 빅데이터를 저장하는 비용이 획기적으로 감소하여 빅데이터를 저장하게 되었음
- ✓ 하지만, 이렇게 저장된 빅데이터를 활용하는 분야는 검색, 집계 등 단순 분야였음



1.3 빅데이터 속성

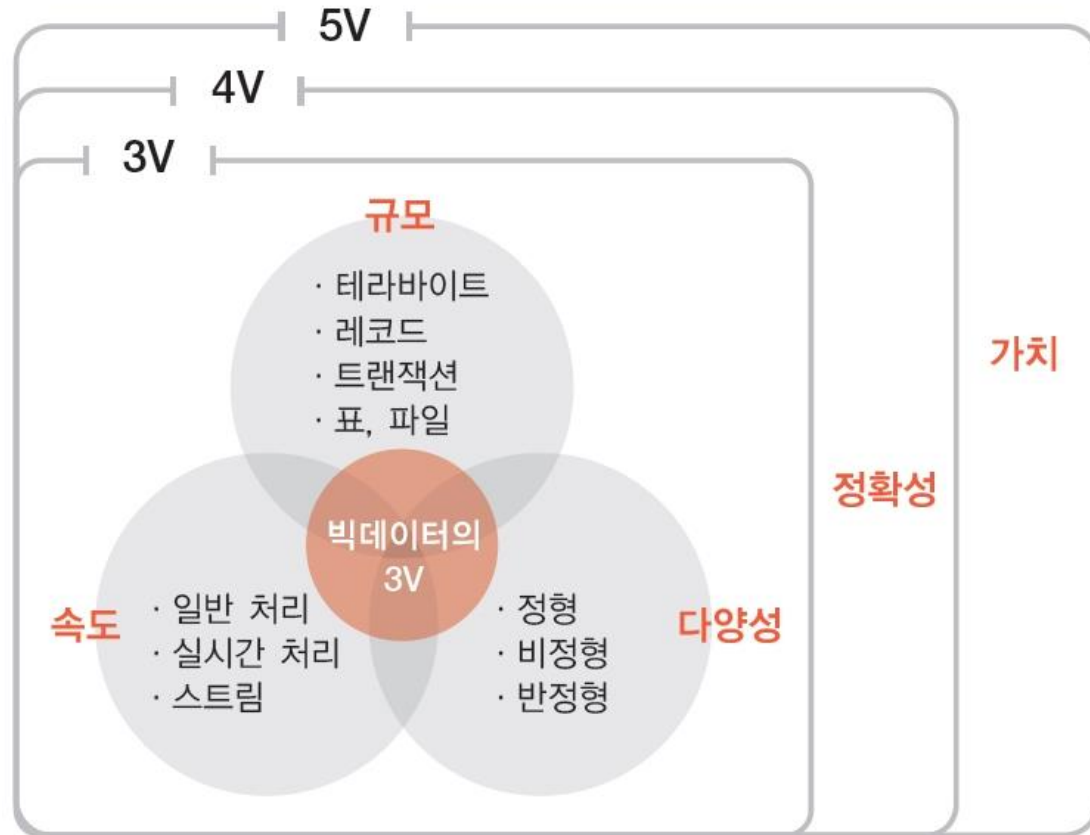


그림 1-2 빅데이터의 속성 [02]

<https://ikkison.tistory.com/66> 참조

1.4 정형과 비정형

ID	Company	Last Name	First Name	E-mail Address	Job Title	Business Home Phone	Mobile Phone Number	Address	City	State/Province	ZIP/Postal Code	Count
1	Company A	Birds	Anna	Owner	(12)955-0100	(12)955-0123	1st Street	Seattle	WA	99999	USA	
2	Company B	Graticos	Antonio	Owner	(12)955-0100	(12)955-0123	2nd Street	Boston	MA	99999	USA	
3	Company C	Awen	Thomas	Purchasin	(12)955-0100	(12)955-0123	3rd Street	Los Angeles	CA	99999	USA	
4	Company D	Lie	Christina	Purchasin	(12)955-0100	(12)955-0123	4th Street	New York	NY	99999	USA	
5	Company E	O'Donnell	Marlin	Owner	(12)955-0100	(12)955-0123	5th Street	Minneapolis	MN	99999	USA	
6	Company F	Pérez-Ota	Francisco	Purchasin	(12)955-0100	(12)955-0123	6th Street	Milwaukee	WI	99999	USA	
7	Company G	Xie	Ming-Yang	Owner	(12)955-0100	(12)955-0123	7th Street	Boise	ID	99999	USA	
8	Company H	Andersen	Elizabeth	Purchasin	(12)955-0100	(12)955-0123	8th Street	Portland	OR	99999	USA	
9	Company I	Mortensen	Sven	Purchasin	(12)955-0100	(12)955-0123	9th Street	Salt Lake City	UT	99999	USA	
10	Company J	Wacker	Roland	Purchasin	(12)955-0100	(12)955-0123	10th Street	Chicago	IL	99999	USA	
11	Company K	Kirchne	Peter	Purchasin	(12)955-0100	(12)955-0123	11th Street	Miami	FL	99999	USA	
12	Company L	Edwards	John	Purchasin	(12)955-0100	(12)955-0123	12th Street	Las Vegas	NV	99999	USA	
13	Company M	Ludick	Andre	Purchasin	(12)955-0100	(12)955-0456	13th Street	Memphis	TN	99999	USA	
14	Company N	Grilo	Carlos	Purchasin	(12)955-0100	(12)955-0456	14th Street	Denver	CO	99999	USA	
15	Company O	Kupkova	Helena	Purchasin	(12)955-0100	(12)955-0456	15th Street	Honolulu	HI	99999	USA	
16	Company P	Goldschm	Daniel	Purchasin	(12)955-0100	(12)955-0456	16th Street	San Francisco	CA	99999	USA	
17	Company Q	Bagel	Jean Philippe	Owner	(12)955-0100	(12)955-0456	17th Street	Seattle	WA	99999	USA	
18	Company R	Auder	Mc Catherine	Purchasin	(12)955-0100	(12)955-0456	18th Street	Boston	MA	99999	USA	
19	Company S	Eggerer	Alexander	Account	(12)955-0100	(12)955-0789	19th Street	Los Angeles	CA	99999	USA	
20	Company T	Li	George	Purchasin	(12)955-0100	(12)955-0789	20th Street	New York	NY	99999	USA	
21	Company U	Tham	Bernard	Account	(12)955-0100	(12)955-0789	21th Street	Minneapolis	MN	99999	USA	
22	Company V	Ramos	Luciana	Purchasin	(12)955-0100	(12)955-0789	22th Street	Milwaukee	WI	99999	USA	
23	Company W	Enin	Michael	Purchasin	(12)955-0100	(12)955-0789	23th Street	Portland	OR	99999	USA	
24	Company X	Hasselbe	Jones	Owner	(12)955-0100	(12)955-0789	24th Street	Salt Lake City	UT	99999	USA	
25	Company Y	Rodman	John	Purchasin	(12)955-0100	(12)955-0789	25th Street	Chicago	IL	99999	USA	
26	Company Z	Liu	Fan	Account	(12)955-0100	(12)955-0789	26th Street	Miami	FL	99999	USA	
27	Company AA	Toh	Karen	Purchasin	(12)955-0100	(12)955-0789	27th Street	Las Vegas	NV	99999	USA	
28	Company BB	Raghav	Amritansh	Purchasin	(12)955-0100	(12)955-0789	28th Street	Memphis	TN	99999	USA	
29	Company CC	Lee	Soo Jung	Purchasin	(12)955-0100	(12)955-0789	29th Street	Denver	CO	99999	USA	

정형 Data

```

1 #Software: Microsoft Internet Information Services X.X-
2 #Version: X-
3 #Date: 2010-03-24 07:00:01-
4 #Fields: date time s-sitename s-computername s-ip cs-method cs-uri-stem cs-uri-query s-port cs-status
5 2010-03-24 07:00:01 ZZZZC941948879 RUFFLES 222.222.222.222 GET / - 80 - 220.181.7.113 HTTP/1.1
6 2010-03-24 07:00:23 ZZZZC941948879 RUFFLES 222.222.222.222 GET /2009/12/im_not_mean_im_just_a
7 2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-blank.gif - 80 - 217.2
8 2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /grep-options.gif - 80 - 217.2
9 2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-cat.gif - 80 - 217.2
10 2010-03-24 07:00:32 ZZZZC941948879 RUFFLES 222.222.222.222 GET /terminal-pwd-cd.gif - 80 - 217
11 2010-03-24 07:00:39 ZZZZC941948879 RUFFLES 222.222.222.222 GET /robots.txt - 80 - 95.55.207.95
12 2010-03-24 07:00:39 ZZZZC941948879 RUFFLES 222.222.222.222 GET /rss-short.xml - 80 - 173.45.23
13 2010-03-24 07:00:43 ZZZZC941948879 RUFFLES 222.222.222.222 GET /2009/08/22-things-you-dont-knc
14 2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /screen.css - 80 - 98.88.35.133
15 2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/rss-header-red.gif - 80 -
16 2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/logo.jpg - 80 - 98.88.35.1
17 2010-03-24 07:00:44 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/input-emailsend.jpg - 80 -
18 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /images/cm-ebook-banner.gif - 8
19 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/bg.jpg - 80 - 98.88.35.133
20 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/bg-top.jpg - 80 - 98.88.35
21 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /21things/checkout-login.gif -
22 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /img/topnav-contact.jpg - 80 -
23 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /21things/portent-email-sub.gif
24 2010-03-24 07:00:45 ZZZZC941948879 RUFFLES 222.222.222.222 GET /rss-header.jpg - 80 - 98.88.35

```

Log Data

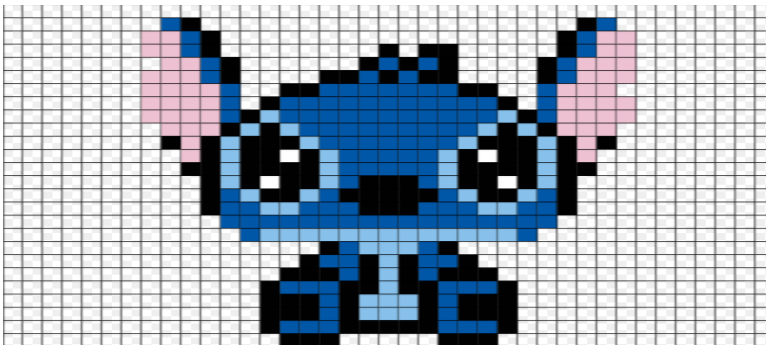
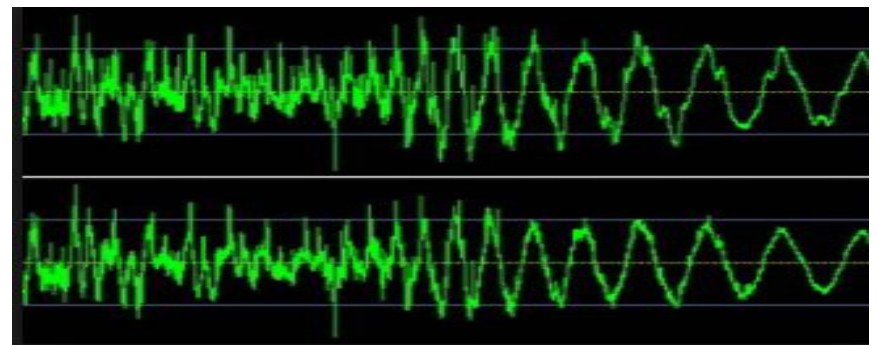


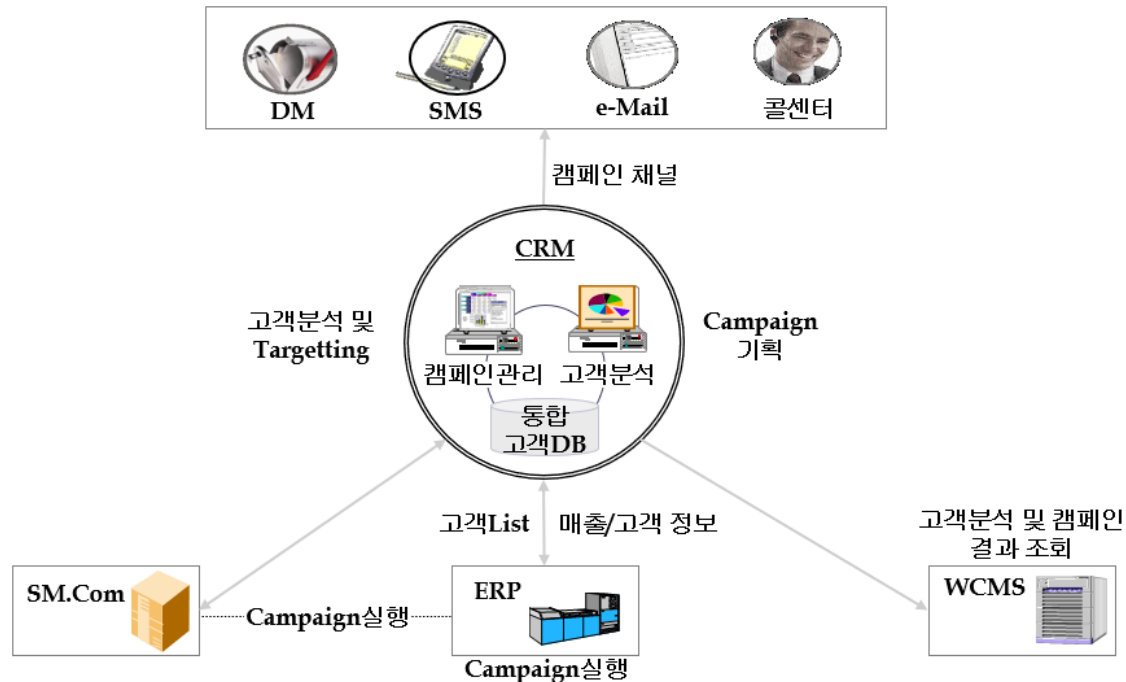
Image Data



Sound Data

1.5 데이터웨어하우스

- 아래는 2000년 초반의 데이터웨어하우스 구성도로 정형 데이터베이스로 구성되어 있음
- 현재는 위의 구성에 추가로 이메일, 콜센터 녹취 자동 Text 변환, .com 로그 정보 등이 추가되어 하둡 시스템으로 관리되고 있음
- 과거에는 콜센터 녹취나 .com 로그 정보는 관리하지 않았음
- 오프라인에서 온라인으로 고객과 접촉 채널이 바뀌면서 온라인 로그 정보가 중요해짐



1.6 Static Data와 Event Log Data

- Static Data / Event Log Data

Static Data는 개체의 속성에 해당하는 데이터로 시간에 따라 바뀌지 않는 데이터를 말함

예: 성별, 연령, 지역, 제조사, 생산일 등

Event Log Data는 개체의 상태에 해당하는 데이터로 시간에 따라 바뀌는 데이터를 말함

RDB 형태로 저장하는 것이 유리함

예: 현 위치, 조회 키워드, 클릭 페이지 등

일반적으로 행은 관측단위, 열은 변수로 지정되는 정형 데이터의 구조에서는

Event Log 데이터를 처리하기 어려움

하지만, 현재 Event Log를 이용한 데이터 처리 수요가 증가하고 있음

Event Log는 하둡 시스템을 저장하는 것이 유리함

예: 구글, 페이스북 보유 텍스트, 센서 데이터 등

1.6 Static Data와 Event Log Data

- 쿠팡이 보유한 정보

거래정보: 누가, 언제, 무엇을 구매했는지에 관한 정보

상품정보: 바코드, 상품분류, 상품명, 제조사, Seller 정보, 가격 등

고객정보: 배송, 과금을 위한 정보

Web, App 사용정보: 접속일자, 링크 클릭 정보, 검색 정보 등

각 정보를 RDB, Hadoop 중 어느 곳에 저장하는 것이 효율적일까요?



<https://youtu.be/IU9OLSVyluw> 참조

1.7 데이터 처리 변화

데이터수집

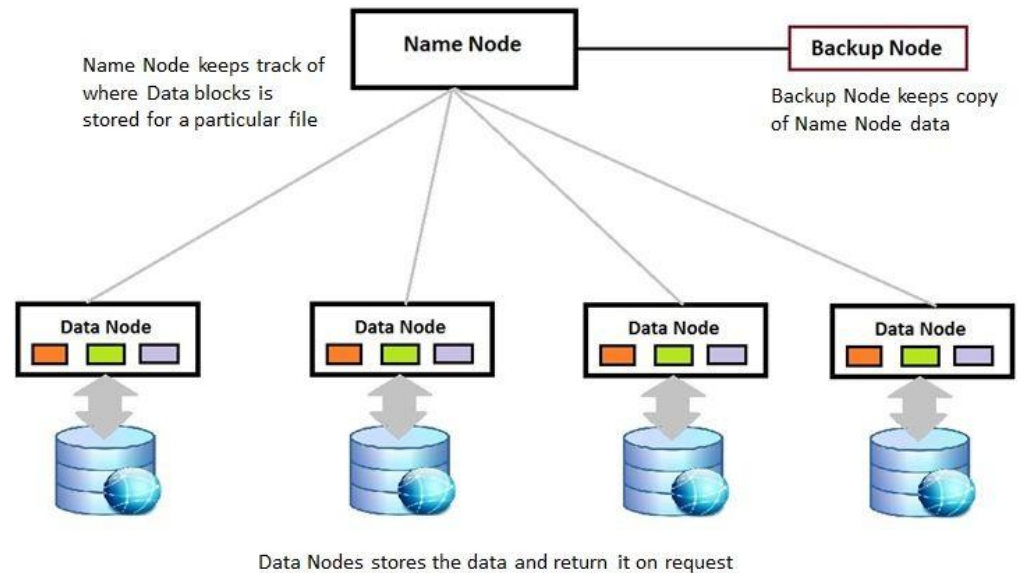
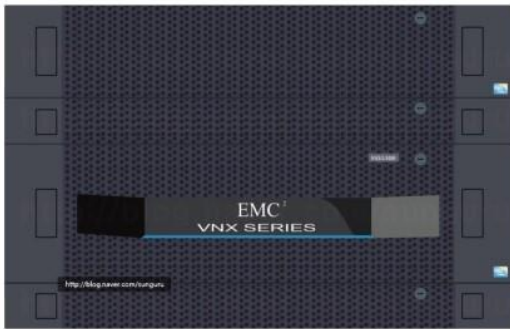
저장

집계

지능화



1.7 데이터 처리 변화



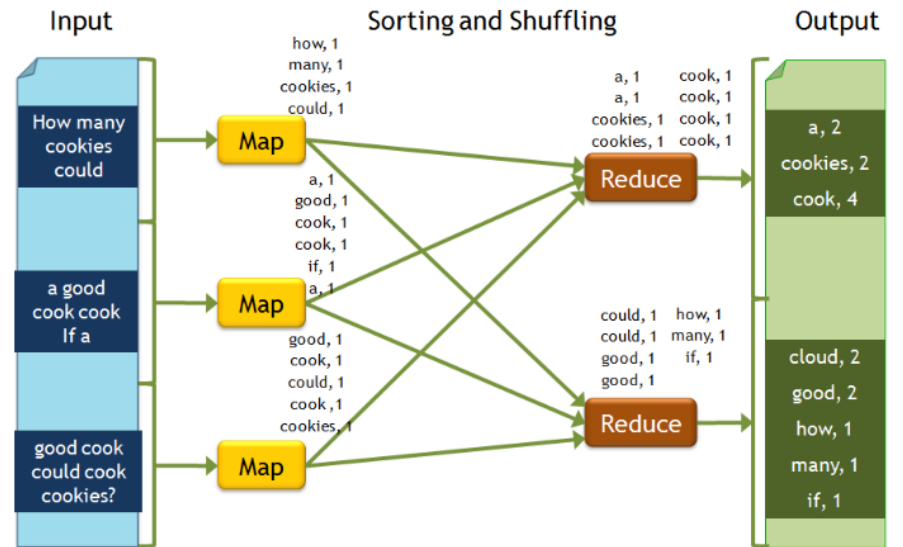
1.7 데이터 처리 변화

데이터수집

저장

집계

지능화



By Manaranjan Pradhan

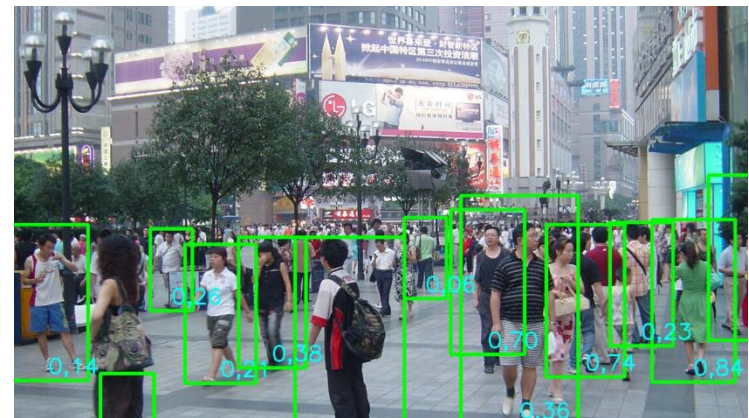
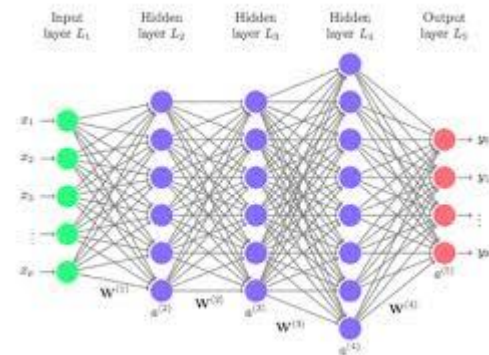
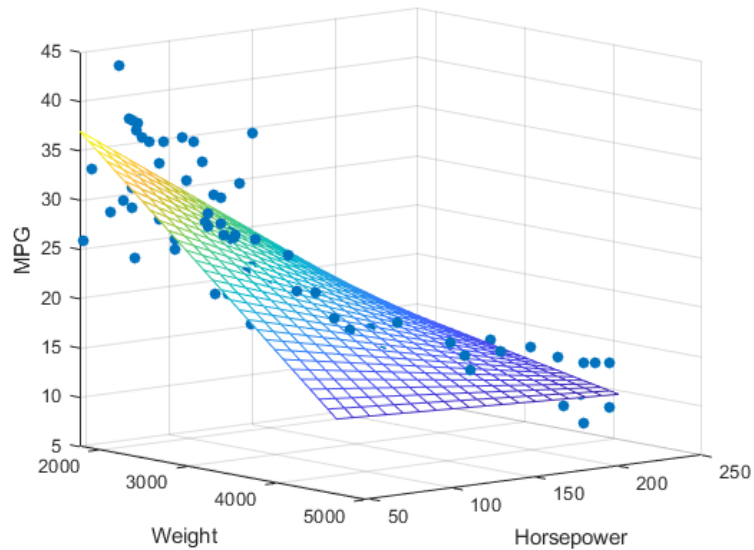
1.7 데이터 처리 변화

데이터수집

저장

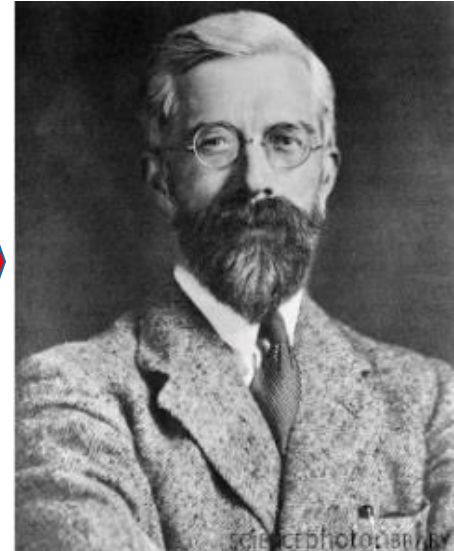
집계

지능화



1.8 통계학

통계학: 표본으로 부터 모집단의 정보를 추정하거나 모집단의 상태를 추측하는 과학



상관/회귀 기본개념
제시

상관/회귀분석 완성
중심극한정리

표본이론 정립
현대 추측 통계학 정립

빅데이터가 왜 필요한가?

1.8 통계학

간단한 문제: 인구 중에 키 185cm 이상인 사람의 비율은?

솔루션 1: 1,000명의 키를 검사하여 185cm 이상인 사람의 비율로 추정

솔루션 2: 185cm 이상인 사람이 나올 때까지 검사하여 비율은 $1/n$ 으로 추정

솔루션 3: 30명의 키를 측정하여 평균과 표준편차를 구해 정규분포 가정 하에 계산함

가장 정확한
방법은?

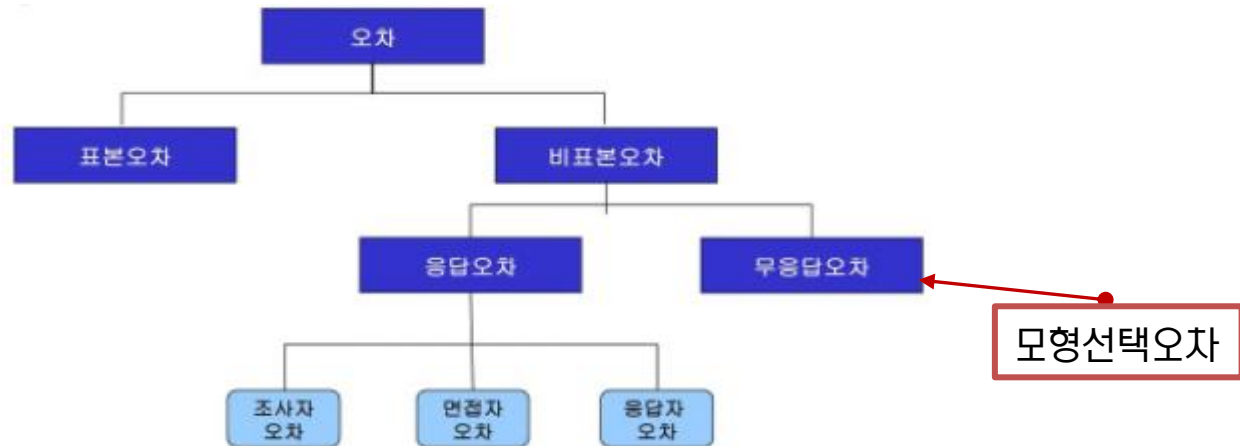


가장 효율적인
방법은?

1.8 통계학

기본적으로 표본 수를 최소화하고자 함(표본 수 증가는 비용의 증가)

표본오차는 표본 수가 증가하면 줄어들이지만, 비 표본 오차는 과학적으로 접근이 어려워 계산이 불가능함
비표본 오차에 의해 참값과 통계적 추정값이 다른 결과를 줄 가능성이 존재함



Small Data의 경우, 분석 정확도를 위해 모든 정보를 사용해 추정 혹은 의사결정 수행함
분포 가정을 통해 재현성을 검정하고자 함

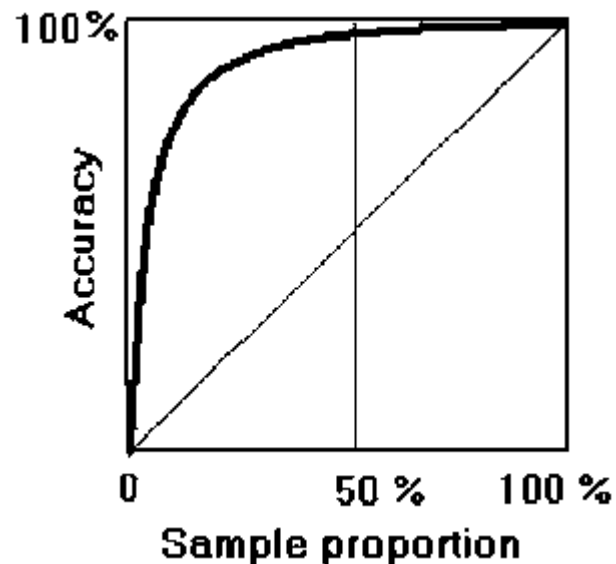
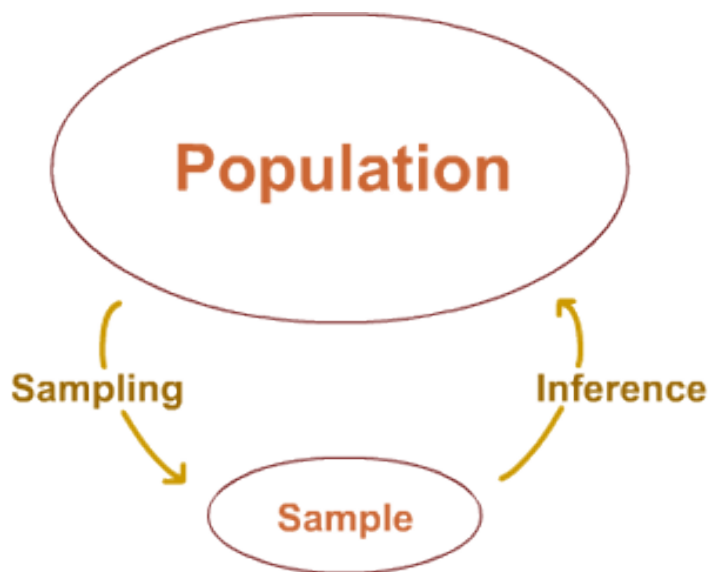
재현성: 다른 표본에서도 우리의 추정이나 의사결정이 재현됨을 증명하는 것

Bid Data의 경우는 풍부한 데이터 셋을 가지고 있기 때문에 재현성을 분포를 통해 증명하는 것이 아니고 실증적으로 재현이 가능함

1.8 통계학

기존 통계적 방법론은 빅데이터가 존재해도 추정 혹은 의사결정의 정확도가 크게 증가하지 않는다.

왜냐하면, 표본오차는 거의 "0"에 가깝게 줄었지만 비표본 오차는 그대로 존재하기 때문임



1.9 통계학과 빅데이터 접근 차이

2000년대 이후, 분석에 대한 요구사항은 다양해지고 있음

과거: 자신의 연구가설을 증명(주로 학문적 요구)

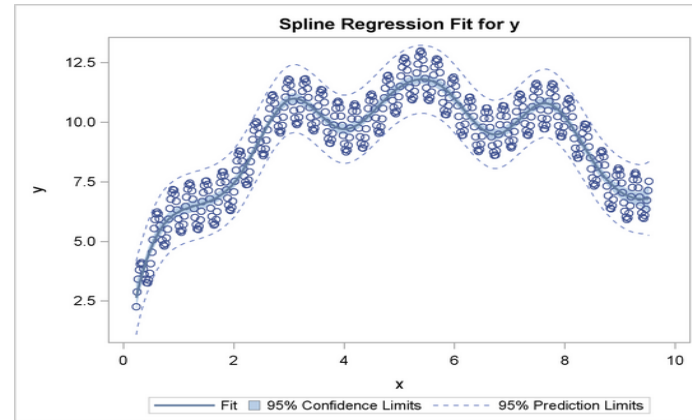
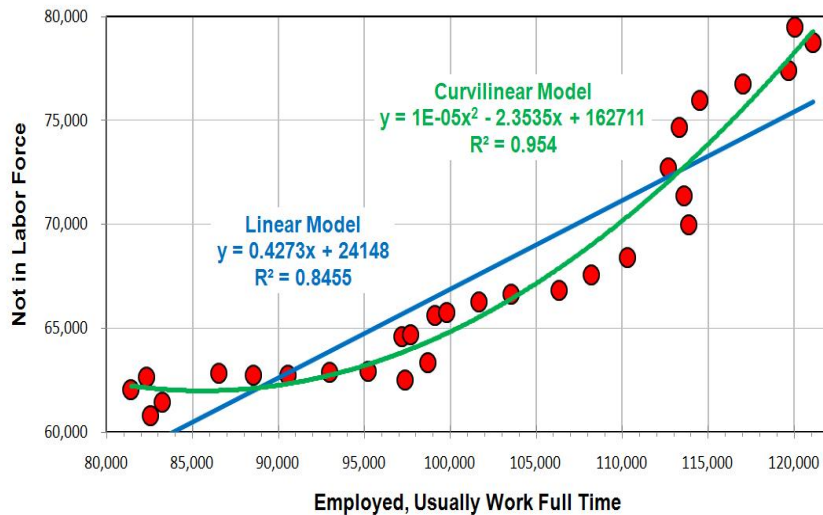
현재: Prediction/Forecasting, Decision Making 등 다양한 Biz. 요구 등장

	데이터	관점	모형	모수 절약
전통적 통계학	분석을 위해 필요 최소 데이터를 수집함	모집단의 구조파악을 통한 추론 (보수적 관점)	Linear 중심	설명변수의 수를 최소화하여 분석의 자유도를 확보하고자 함
빅데이터 접근	기존에 축적된 자료를 통해 분석함	모집단의 구조보다 예측에 초점 (적극적 관점)	Non-Linear로 확장	데이터가 많으므로 변수의 수에 구애 받지 않음

Q: Linear 모형을 사용하는 이유?

1.9 통계학과 빅데이터 접근 차이

Linear Regression and Non-Linear Regression Model



스몰 데이터에서 고차항의 모형을 사용하면 할 수는 있으나 Over-fitting 위험이 커진다.

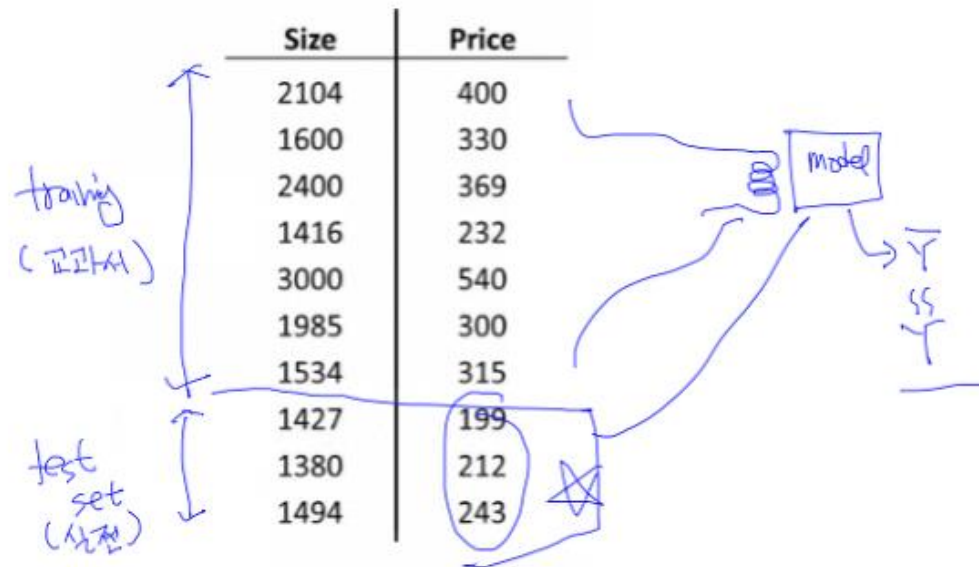
빅데이터에서는 복잡한 모형을 사용해도 Over-fitting 위험이 덜하다.

1.9 통계학과 빅데이터 접근 차이

빅 데이터의 경우, 모델을 만드는데 모든 데이터를 사용하지 않고 모델 검증을 위한 Test-Set을 남겨 둘 수 있는 여유가 존재한다.

이 방법이 실증적 재현성이다.

Training and test sets



Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

http://www.holehouse.org/mlclass/10_Advice_for_applying_machine_learning.html

1.9 통계학과 빅데이터 접근 차이

- ✓ 통계학은 분석 목적에 따라 데이터를 수집하므로 교락 등의 문제를 해결하는 형태로 데이터를 수집하지만, 빅데이터는 분석 목적과 상관없이 축적된 자료이기 때문에 분석 및 해석시 주의가 필요
- ✓ 빅데이터는 데이터가 자동으로 확보 됨에 따라 여러 분야에서 분석을 다양하게 시도할 수 있는 긍정적인 측면 존재
- ✓ 빅데이터에서는 보유자료가 모집단에 근사한 경우가 많아 통계적 접근이 아닌 집계에 가까움
이 경우, 집계결과를 확대해서 전체로 적용하는 오류 역시 존재함
- ✓ 빅데이터 분야에서도 오랫동안 연구 개발된 통계 이론을 적용하고 있음

Machine Learning <- Machine Learning + Statistics

- ✓ 통계학 이론도 빅데이터 상황에 맞도록 새로운 이론들이 개발되고 있음
(Bagging, Boosting, Lasso 등)

2. 하둡

2.1 하둡의 등장 배경

2.2 분산처리

2.3 하둡 에코 시스템

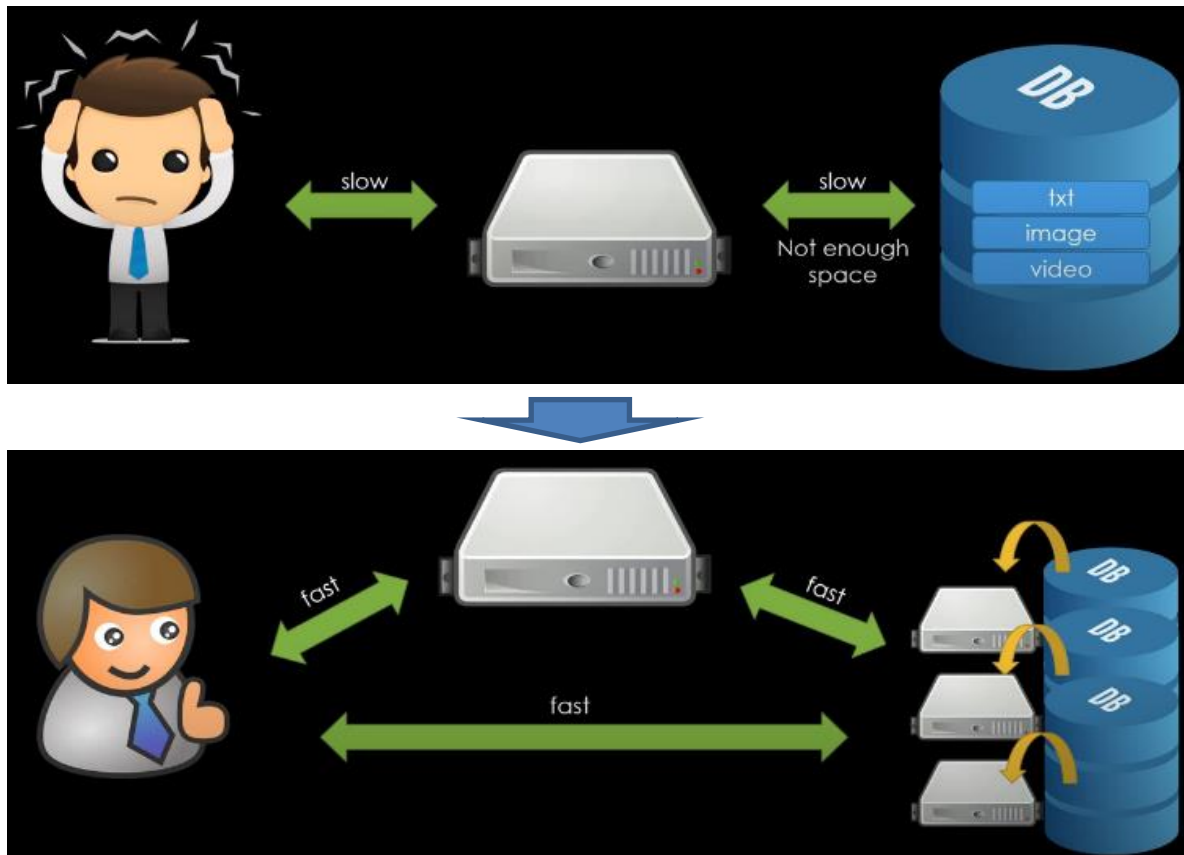
2.4 맵리듀스 프로그래밍

2.5 Word Count 예제

2.1 하둡의 등장 배경

- ✓ 구글의 고민은 데이터가 증가하면서 컴퓨터 처리 속도가 현저하게 감소 것임
- ✓ CPU 연산보다 Disk I/O에 시간이 많이 사용됨. 병렬 처리를 수행하는 파일 시스템 개발

<https://youtu.be/IU9OLSVyluw> 참조

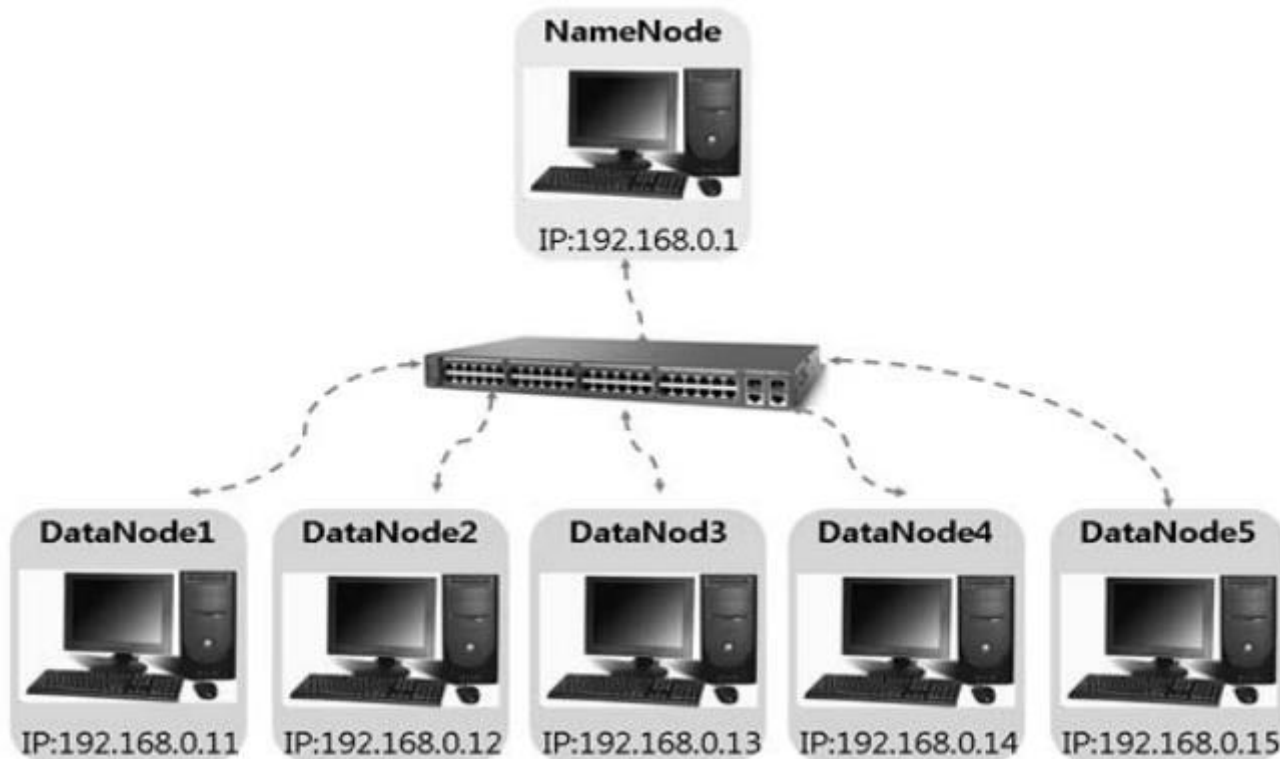


2.1 하둡의 등장 배경

- ✓ 하둡은 구글이 개발한 분산처리 오픈소스 프레임워크임
- ✓ 하둡은 HDFS(Hadoop Distributed File System)과 MapReduce로 구성됨
- ✓ HDFS는 파일 시스템으로 네임 노드(name node)와 데이터 노드(data node)로 구성
- ✓ HDFS는 Google File System으로 출발
- ✓ HDFS는 데이터 분산저장과 데이터 노드 모니터링(Heartbeat Monitor)를 통해 데이터 노드의 작동여부를 감시한다. 또한, 블록관리 기능을 통해 하나의 데이터를 여러 블록으로 복제하여 장애 발생에 대응한다.
- ✓ 네임 노드는 데이터 노드의 파일 구조를 정리한 곳이고 데이터 노드가 실제 데이터를 저장
- ✓ 맵리듀스는 Job Tracker와 Task Tracker로 구성되어 있음
- ✓ Job Tracker는 큰 “job”을 작은 Task로 나누어 실행시키고 결과를 병합 받음
- ✓ 작은 Task로 나누어 실행하는 것이 분산처리임

2.2 분산처리

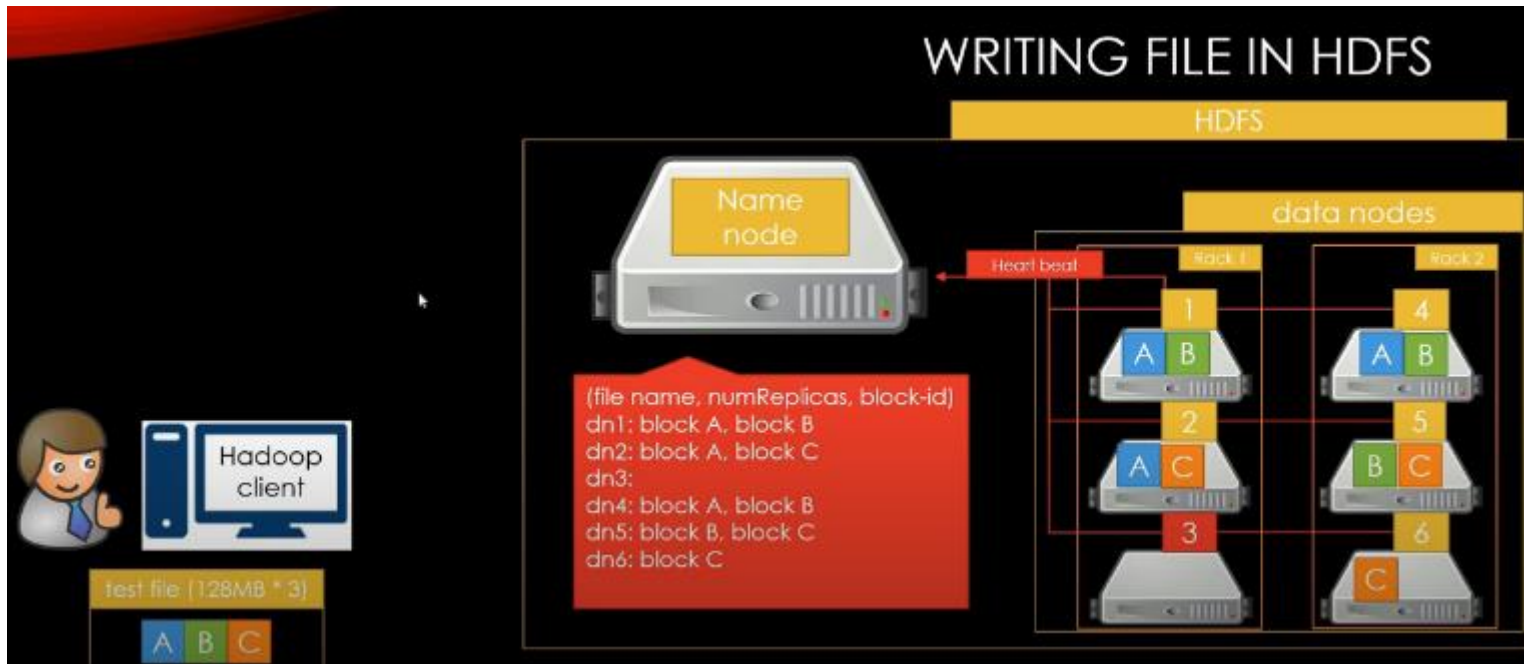
- ✓ 아래는 네임노드와 데이터노드의 연결에 대한 개념도임



참조: 신지은, 오윤식, 임동훈, 빅데이터 K-평균 클러스터링을 위한 RHadoop 플랫폼, 한국데이터정보과학회지, 2016

2.2 분산처리

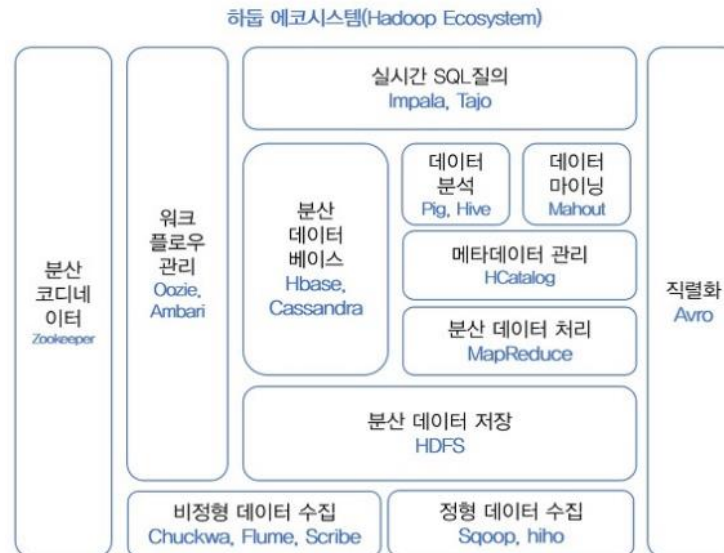
- ✓ 하둡 클라이언트는 데이터를 HDFS에 저장하는 역할을 하고, 네임노드는 데이터 노드의 정상 작동여부와 각 데이터가 어디에 저장되어 있는지의 정보를 가지고 있음
- ✓ 데이터 노드에는 원본 데이터를 여러 조각으로 나누어 분산 보관 및 데이터 손실에 대비한 여분을 보관함
- ✓ 파일을 읽을 때에는 네임노드에서 원하는 정보의 위치를 찾아 데이터노드를 Access함



<https://youtu.be/IU9OLSVyluw> 참조

2.3 하둡 에코 시스템

- ✓ 대용량 데이터를 여러 대의 컴퓨터가 분산 처리하는 것이 비용이 저렴한 경우 유용함
- ✓ RDB의 경우, 국내 대기업에서 1테라를 유지하는데 1억 정도 소요됨
- ✓ 하둡은 리눅스 OS에 설치하여 사용하고 멀티 분산 모드를 사용하여 작동함
- ✓ 가상분산모드: 한대의 컴퓨터에 여러 대의 컴퓨터를 만드는 가상분산 방식
- ✓ 멀티분산모드: 네임 노드 한대와 여러 데이터 노드를 서로 다른 컴퓨터에 설치하는 방식
- ✓ 하둡 에코시스템은 HDFS와 MapReduce외에 많은 유틸리티와 분석 도구로 이루어져 있음
- ✓ 하둡 에코 시스템 안에는 Pig(돼지), Hive(벌떼), ZooKeeper(사육사) 등
각종 동물 이름의 시스템이 있는데 이 프로젝트의 시초가 된 노란 코끼리 네이밍에 영향을 받음



2.4 MapReduce 프로그래밍

- ✓ 하둡은 자바언어로 만들어져 있고, 자바언어로 작동할 수 있음
- ✓ R, Hive 등 많은 분석 라이브러리가 존재하여 자바를 사용하지 않고도 분석 가능

```
1 package hadoop;
2
3 import java.io.IOException;
4 import java.util.StringTokenizer;
5
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.LongWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.Mapper;
10
11 public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
12     private final static IntWritable one = new IntWritable(1);
13     private Text word = new Text();
14
15     @Override
16     protected void map(LongWritable key, Text value,
17         Mapper<LongWritable, Text, Text, IntWritable>.Context context)
18         throws IOException, InterruptedException {
19         StringTokenizer itr = new StringTokenizer(value.toString());
20         while (itr.hasMoreTokens()) {
21             word.set(itr.nextToken());
22             context.write(word, one);
23         }
24     }
25 }
```

```
package hadoop;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
        Reducer<Text, IntWritable, Text, IntWritable>.Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

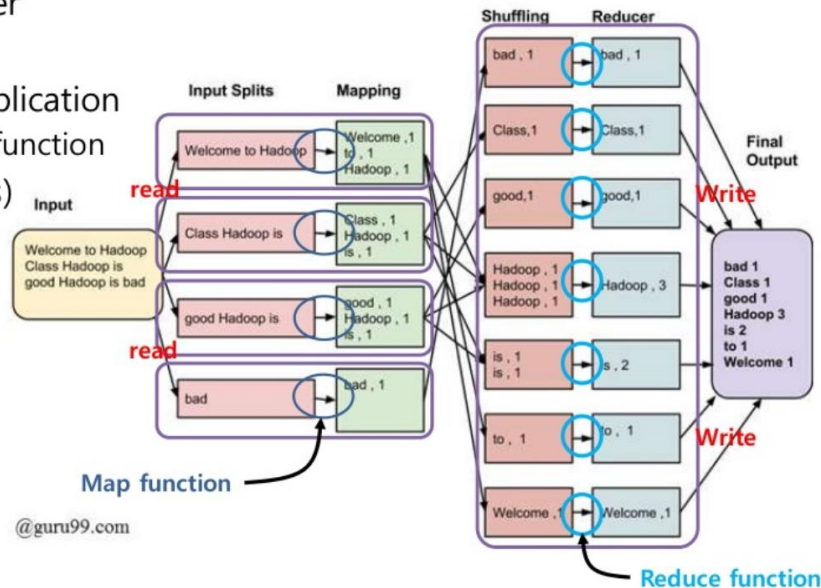
2.4 MapReduce 프로그래밍

- ✓ 모든 하둡 프로그램은 맵과 리듀스 단계를 거친다.
- ✓ Mapper는 데이터를 나누어 분산 처리하고 이를 다시 정렬하여 Reducer로 보낸다.
- ✓ Mapper가 데이터를 정렬하기 위해서 키가 필요하기 때문에 <key, value> 형태로 리턴 해야 한다.
- ✓ Reducer는 정렬된 <key, value> 를 받아 원하는 형태로 출력한다.

Word Counting Example

- Computer cluster
- Input File
- Hadoop MR application
 - Map/Reduce function
- Output File(hdfs)

Computing Node



2.4 MapReduce 프로그래밍

- ✓ 데이터를 N개의 조각으로 나누어 처리하고 이를 종합할 수 있는 단순 업무에 적합함
- ✓ 빅데이터라고 해도 하둡이 적합하지 않은 업무도 많이 있음
- ✓ 데이터를 분할하여 연산이 불가능한 경우, 사용할 수 없음

- 웹로그에서 특정 페이지를 조회한 고객을 찾아라
- <xml>에서 그룹별로 특정 값을 더하는 등의 단순연산
- 빅 데이터에서 회귀계수 구하기

$$X'X = \begin{bmatrix} N & \sum x_1 & \sum x_2 & \sum x_3 \\ \sum x_1 & \sum x_1^2 & \sum x_1x_2 & \sum x_1x_3 \\ \sum x_2 & \sum x_2x_1 & \sum x_2^2 & \sum x_2x_3 \\ \sum x_3 & \sum x_3x_1 & \sum x_3x_2 & \sum x_3^2 \end{bmatrix} \quad X'y = \begin{bmatrix} \sum y \\ \sum X_1y \\ \sum X_2y \\ \sum X_3y \end{bmatrix}$$

$$\hat{\beta} = (X'X)^{-1}X'y$$

2.5 Word Count 예제

- ✓ 하둡을 실행한다.

```
[hadoop@localhost sbin]$ start-all.sh
```

- ✓ 하둡이 제대로 실행되었는지를 JPS(JAVA Virtual Machine process status)확인
네임노드와 데이터 노드 프로세스가 성공적으로 올라옴

```
[hadoop@localhost sbin]$ jps
```

```
1984 NodeManager  
1430 SecondaryNameNode  
3430 Jps  
1255 DataNode  
1118 NameNode  
1646 ResourceManager
```

2.5 Word Count 예제

- ✓ 하둡 작동 Test를 위해 ~/stack/hadoop/README.txt 파일에 있는 단어 수를 세기
- ✓ ~/stack/hadoop/README.txt 파일을 HDFS에 /example 폴더로 복사
- ✓ 자바로 만들어진 hadoop-mapreduce-examples-2.9.2.jar의 wordcount class를 이용하여 /example/README.txt 파일의 단어를 센 결과를 /output 폴더에 저장

```
[hadoop@localhost mapreduce]$ hadoop fs -mkdir /example
[hadoop@localhost mapreduce]$ hadoop fs -copyFromLocal
                               ~/stack/hadoop/README.txt /example

[hadoop@localhost mapreduce]$ hadoop jar
hadoop-mapreduce-examples-2.9.2.jar wordcount /example/README.txt /output

[hadoop@localhost mapreduce]$ hadoop fs -ls /output 리눅스 명령
[hadoop@localhost mapreduce]$ hadoop fs -cat /output/part-r-00000 | more
```

3. R

3.1 R 설치

3.2 객체 변수

3.3 연산

3.4 벡터

3.5 Logical 연산

3.6 수학적함수

3.7 행렬

3.8 데이터프레임

3.9 if, for 문

3.10 function 문

3.1 R 설치

- ✓ R은 프롬프트에서 R 명령으로 실행
- ✓ R이 없다면 아래의 명령으로 설치

```
> R
bash: R: 명령을 찾을 수 없습니다...
> sudo yum install -y epel-release
> sudo yum update
> sudo yum install R
```

- ✓ 리눅스 OS를 확인한 결과, centos 7 이므로 적합한 r-studio를 다운로드하여 설치
- ✓ rstudio 명령으로 실행

```
> cat /etc/system-release
> wget https://download1.rstudio.org/desktop/centos7/x86_64/rstudio-1.2.5042-x86_64.rpm
> yum install rstudio-1.2.5042-x86_64.rpm
```

3.2 객체 변수

- 변수명은 알파벳, 숫자, 마침표(.), underscore(_) 등을 조합해 만듦
- 변수에 특정 값(혹은 변수)을 할당할 때 <-을 사용하면 됨. =을 사용할 수도 있음
- 모든 이름의 시작은 알파벳 또는 마침표(.)로 함

```
abc <- 3      # OK
.jeong <- abc # OK
2.res <- 3    # 예러...
```

- 여러 명령어를 한 줄에 입력할 때는 세미콜론(;)으로 구분

```
beta.0 <- 3; beta.1 <- 2
```

- 주석문(comment)은 #을 이용

```
rmnorm(10) # to generate 10 random numbers

## [1] 1.93923266 -0.33087280 -0.37523422 0.04858852 0.27363754
## [6] 0.24788490 -0.44575870 -1.34233768 -1.72312531 -1.02432005
```

빅데이터분석기초, 한국정보화진흥원, 2016 참조

3.3 연산

➤ 산술연산

```
x <- 11; y <- 3
x+y
## [1] 14

x-y
## [1] 8

x*y
## [1] 33

x/y
## [1] 3.666667

x^y
```

➤ 수학 함수

```
x <- 10; y <- 3.21; n <- 2
log(x)
## [1] 2.302585

log10(x)
## [1] 1

log(n, x)
## [1] 0.30103

exp(x)
## [1] 22026.47

sin(x) # cos(x), tan(x), asin(x), acos(x), atan(x)
## [1] -0.5440211

abs(x)
## [1] 10

sqrt(x)
## [1] 3.162278
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.4 벡터

벡터(Vector)

벡터를 구성하는 모든 성분은 같은 타입이어야 한다.

실수(double), 정수(integer), 문자열(string), 논리값(logical) 등으로 구성할 수 있다.

```
x <- c(1, 2.5, 3.2)           # double
y <- c(1L, 2L, 3L)           # integer
z <- c("KTX", "Saemaul", "Mugunghwa") # string
v <- c(TRUE, FALSE, FALSE, TRUE) # logical
```

```
fruit <- c(5, 3, 2)
names(fruit) <- c("apple", "orange", "peach")
fruit

##  apple orange  peach
##      5      3      2
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.5 Logical 연산

논리값으로 구성된 논리값 벡터(logical vector)를 사용할 수 있다.

```
x <- 1:10; y <- rep(5, 10)
z <- x<5          # less than
sum(z)

## [1] 4

x<=5             # less than or equal to

## [1] TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

R 은 다양한 수학/통계 함수를 제공한다.

```
x <- rnorm(10)

x

## [1] -0.34246270  0.58343596  1.54340829 -1.68275292 -0.07331261
## [6]  1.02802626 -0.71574842 -0.62199705  0.03085437 -1.39492435
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.6 수학 함수

```
max(x)      # 최대값
```

```
## [1] 1.543408
```

```
min(x)      # 최소값
```

```
## [1] -1.682753
```

```
sum(x)      # 모든 성분의 합
```

```
## [1] -1.645473
```

```
prod(x)     # 모든 성분의 곱
```

```
## [1] 0.0007493877
```

```
mean(x)     # 평균
```

```
## [1] -0.1645473
```

```
median(x)   # 중앙값
```

```
var(x)      # 분산
```

```
## [1] 1.034316
```

```
sd(x)       # 표준편차
```

```
## [1] 1.017013
```

```
cov(x, y)   # 공분산
```

```
## [1] -1.120272
```

```
cor(x, y)   # 상관계수
```

```
## [1] -0.363824
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.7 행렬

```
z <- array(1:20, dim=c(4,5))
z
```

	[,1]	[,2]	[,3]	[,4]	[,5]
## [1,]	1	5	9	13	17
## [2,]	2	6	10	14	18
## [3,]	3	7	11	15	19
## [4,]	4	8	12	16	20

```
A*B
```

	[,1]	[,2]	[,3]	[,4]	[,5]
## [1,]	1	25	81	169	289
## [2,]	4	36	100	196	324
## [3,]	9	49	121	225	361
## [4,]	16	64	144	256	400

```
x <- 1:4
y <- 5:8

cbind(x, y)

##      x y
## [1,] 1 5
## [2,] 2 6
## [3,] 3 7
## [4,] 4 8

rbind(x, y)

##      [,1] [,2] [,3] [,4]
## x      1   2   3   4
## y      5   6   7   8
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.7 행렬

```
A <- matrix(runif(20), 5, 4)
t(A)           # matrix transposition

##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.09322308 0.1439579 0.3752732 0.2081587 0.48607438
## [2,] 0.01076963 0.3981393 0.9109081 0.7525530 0.50526267
## [3,] 0.58973203 0.1532971 0.1512634 0.4680938 0.02143471
## [4,] 0.03015954 0.7546799 0.9172832 0.5942066 0.40688646

B <- t(A)%*%A    # %*%: Matrix multiplication
solve(B)         # Inverse matrix

##           [,1]      [,2]      [,3]      [,4]
## [1,] 10.8349300 -5.9461743 0.26166795 1.07313024
## [2,] -5.9461743 10.0950556 -0.90147753 -6.70089942
## [3,] 0.2616679 -0.9014775 2.30368313 0.06294342
## [4,] 1.0731302 -6.7008994 0.06294342 6.27789774
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.7 행렬

```
a <- matrix(1:20, 4, 5)
a

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20

  apply(a, 1, mean)    # to every row
## [1]   9 10 11 12

  apply(a, 2, mean)    # to every column
## [1]  2.5  6.5 10.5 14.5 18.5
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.8 데이터프레임

```
x <- c(100, 75, 80)
y <- c("A302043", "A302044", "A302045")
z <- data.frame(score=x, ID=y)
```

csv 파일을 읽어들이고 데이터프레임으로 저장하려면 `read.csv()` 함수 사용

```
x <- read.csv(file="table.csv", header=T)
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.9 if, for문

➤ ifelse() 함수

```
y <- -3:3
z <- ifelse(y < 0, -1, 1)
cbind(y, z)
```

```
##      y  z
## [1,] -3 -1
## [2,] -2 -1
## [3,] -1 -1
## [4,]  0  1
## [5,]  1  1
## [6,]  2  1
## [7,]  3  1
```

```
n <- 10
x <- 1:n
sum.so.far <- 0
for ( i in 1:n ) {
  sum.so.far <- sum.so.far + x[i]
  print(sum.so.far)
}
```

```
## [1] 1
## [1] 3
## [1] 6
## [1] 10
## [1] 15
## [1] 21
## [1] 28
## [1] 36
## [1] 45
## [1] 55
```

```
sum.so.far
```

```
## [1] 55
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

3.10 function 문

```
my.stat <- function(x)
{
  m <- mean(x)
  s <- sd(x)
  res <- list(m=m, s=s)

  par(mfrow=c(1, 2))
  boxplot(x, main="Boxplot", horizontal=T)
  hist(x, prob=T, col="skyblue", border="white", main="Histogram",
xlab="data")

  return(res)
}
```

빅데이터분석기초, 한국정보화진흥원, 2016 을 참조하였음

4. RHadoop

4.1 RHadoop 설치

4.2 RHadoop 이해

4.3 예제: GDP 기초 통계 구하기

4.4 예제: Iris 종별 평균 구하기

4.5 예제: 구글 ngram 시각화

4.1 R Hadoop설치

- ✓ RHadoop은 rhdfs, rhbase, rmr 패키지를 중심으로 작동함
- ✓ rhdfs: HDFS에 있는 데이터를 R로 호출한다.
<https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads> 에서
rhdfs_1.0.8.tar.gz 다운
- ✓ rmr: MapReduce를 실행한다.
 - > wget https://github.com/RevolutionAnalytics/rmr2/releases/download/3.3.1/rmr2_3.3.1.tar.gz
- ✓ rJava 패키지 설치를 위한 준비
 - > sudo R CMD javareconf

4.1 R Hadoop설치

- ✓ rstudio에서 rmr2 설치를 위한 의존성 패키지를 설치함

```
install.packages(c("Rcpp","RJSONIO","digest","functional","reshape2","stringr",  
"plyr","caTools"))
```

- > sudo R CMD INSTALL rmr2_3.3.1.tar.gz

```
install.packages("rJava")
```

```
HADOOP_CMD=/home/hadoop/stack/hadoop/bin/hadoop R CMD INSTALL  
rhdfs_1.0.8.tar.gz
```

- ✓ 아래 url은 rHadoop를 개발한 RevolutionAnalytics의 튜토리얼 페이지임

<https://github.com/RevolutionAnalytics/rmr2/blob/master/docs/tutorial.md>

4.2 R Hadoop 이해

- ✓ 아래 코드는 R에서 1,2,...,10의 숫자를 제공하여 출력하는 예제임

```
Sys.setenv(HADOOP_CMD="~/stack/hadoop/bin/hadoop")  
Sys.setenv(HADOOP_STREAMING="~/stack/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar")
```

hadoop 실행환경을 설정함

```
library(rhdfs)  
hdfs.init()  
library(rmr2)
```

```
small.ints <- to.dfs(1:10)
```

data를 hdfs로 전송

```
map.fn <- function(k,v) {  
  val = cbind(v, v^2)  
  return(keyval(k, val))  
}
```

hdfs의 data를 k, v 형태로 받아 v로 부터 val을 생성한 후 다시 k, val 형태로 Reduce Task로 전송하는 함수

```
result <- mapreduce(input = small.ints, map = map.fn)  
out <- from.dfs(result)  
out
```

맵리듀스 실행

실행결과를 hdfs에서 R로 전송

RHadoop_EX1.R

4.2 R Hadoop 이해

- ✓ 아래 코드는 홀수와 짝수의 수를 세는 프로그램임
- ✓ mapper는 홀/짝을 구분하고 Reducer는 mapper가 구분한 홀짝 결과 길이를 계산함

```
x <- c(2,6,67,85,7,9,4,21,78,45)
hadoop.dfs <- to.dfs(x)
```

x를 hdfs로 전송

```
mapper <- function(k, v){
  key <- ifelse (v %% 2 == 0, 2, 1)
  return(keyval(key, v))
}
```

hdfs의 x를 k, v 형태로 받아 key에는 홀짝 구분을,
value는 x값을 Reduce Task로 전송하는 함수

```
reducer <- function(k, v){
  value <- length(v)
  return(keyval(k, value))
}
```

mapper로 부터 k, v 형태로 받아 key에 대해 v의 길이를
(1, 6), (2, 4) 형식으로 반환

```
#output <- mapreduce(input = hadoop.dfs, map = mapper)
output <- mapreduce(input = hadoop.dfs, map = mapper, reduce = reducer)
result <- from.dfs(output)
result
```

```
$key
[1] 1 2
$val
[1] 6 4
```

RHadoop_EX2.R

4.2 R Hadoop 이해

- ✓ 아래는 mapper와 Reducer의 연산과정을 나타낸 결과임

Mapper

key	value
2	2
2	6
1	67
1	85
1	7
1	9
2	4
1	21
2	78
1	45

Reducer

key	value
1	6
2	4

4.2 R Hadoop 이해

- ✓ 아래 코드는 $N(100, 10^2)$ 의 난수를 100개 생성하고 이 중 100보다 작은 수, 큰 수를 카운트하는 예제임. reducer 없이 mapper 결과만 확인해 보자.

```
x <- rnorm(100,100,10)
hadoop.dfs <- to.dfs(x)

mapper <- function(., v){
  key <- ifelse(v < 100, "less", "greater")
  return(keyval(key, 1))
}

reducer <- function(k, v){
  value <- sum(v)
  return(keyval(k, value))
}

#output <- mapreduce(input = hadoop.dfs, map = mapper)
output <- mapreduce(input = hadoop.dfs, map = mapper, reduce = reducer)
result <- from.dfs(output)
result
```

RHadoop_EX3.R

4.2 R Hadoop 이해

- ✓ 아래는 WordCount 예제임
- ✓ README.txt를 hdfs 로 복사하고 텍스트를 공백 기준으로 리스트로 만든 다음, 각 단어를 키로 만들어 (단어, 1)로 Reducer에 보냄
- ✓ Reducer는 키 별로 $\text{sum}(v)$ 를 하여 단어의 수를 카운트함

```
hdfs.put("/home/hadoop/stack/hadoop/README.txt", "/tmp/")
hdfs.ls("/tmp/")
outfile <- "/tmp/test"
if(hdfs.exists(outfile)) hdfs.rm(outfile)
```

```
txt = "/tmp/README.txt"
mapper <- function(k,v){
  words.list <- strsplit(v, split=" ")
  words <- unlist(words.list)
  keyval(words, 1)
}
```

```
reducer <- function(k,v) {
  keyval(k, sum(v))
}
```

RHadoop_WordCount.R

4.2 R Hadoop 이해

- ✓ 결과를 데이터프레임 형태로 만든다.
- ✓ R하둡은 자바 언어에서 복잡하게 처리해야하는 문제를 쉽게 처리할 수 있도록 도와준다.
R이 없으면 자바언어로 통계적인 모듈을 만들어 사용해야 하는데 그 과정을 R이 대신 해주기 때문에 자바와 R은 같이 사용되는 경우가 많다.

```
output <- mapreduce(input = txt, input.format="text", output=outfile, map =  
mapper, reduce = reducer)  
result <- from.dfs(output)  
result  
  
result.df = as.data.frame(result)  
colnames(result.df)=c("word", "count")  
head(result.df, 10)
```

4.3 예제: GDP 기초통계

- ✓ 아래 코드는 나라별 GDP를 CSV 파일로 가져와 애플사 2014년 매출액 기준으로 poor, rich를 구분하고 카운트하는 예제임

```
setwd("~/Downloads/examples")
gdp <- read.csv("GDP1.csv", header=FALSE)
names(gdp) <- c("Country Code", "Number", "Country Name", "GDP")
gdp$GDP <- as.numeric(gsub(",", "", gdp$GDP))
gdp.dfs = to.dfs(gdp)

appleRevenue = 183000
mapper <- function(., v) {
  key <- ifelse(v$GDP < appleRevenue, "poor", "rich")
  return(keyval(key, 1))
}
reducer <- function(k, v) {
  value <- length(v)
  return(keyval(k, value))
}
#output <- mapreduce(input = gdp.dfs, map = mapper)
output <- mapreduce(input = gdp.dfs, map = mapper, reduce = reducer)
result <- from.dfs(output)
result
```

RHadoop_GDPData.R

4.4 예제: Iris 종별 평균 구하기

- ✓ 아래는 iris 데이터에서 종별로 sepal.length의 평균을 구하는 예제임

```
iris <- read.csv("iris.csv", header=T)
iris.hdfs = to.dfs(iris)

mapper <- function(k,v) {
  key <- v$Species
  value <- v$sepal.length
  return(keyval(key, value))
}

reducer <- function(k,v) {
  value=mean(v)
  return(keyval(k, value))
}

#output <- mapreduce(input = iris.hdfs, map = mapper)
output <- mapreduce(input = iris.hdfs, map = mapper, reduce = reducer)
result <- from.dfs(output)
result
```

RHadoop_IRIS1.R

4.4 예제: Iris 종별 평균 구하기

- ✓ 아래는 5번째 컬럼을 제외한 나머지 sepal.length, sepal.width, petal.length, petal.width를 가진 데이터프레임을 value로 지정하고 Species에 따라 평균을 구하는 예제임

```
mapper <- function(k,v) {  
  key <- v$Species  
  value <- iris[-5]  
  return(keyval(key, value))  
}  
  
reducer <- function(k,v) {  
  value=apply(v,2, mean)  
  return(keyval(k, value))  
}  
#output <- mapreduce(input = iris.hdfs, map = mapper)  
output <- mapreduce(input = iris.hdfs, map = mapper, reduce = reducer)  
result <- from.dfs(output)  
result
```

RHadoop_IRIS2.R

4.5 예제: 구글 Ngram 시각화

- ✓ 구글 Ngram은 1520년부터 2012년까지 출판된 책을 디지털화하고 이중 800만권의 데이터임
- ✓ Ngram에서 N은 1~5까지 수로 “United States of America”는 4gram이다.
- ✓ 여기서는 1gram 중 s로 시작하는 데이터를 받자. 무려, 2.3G 텍스트다.

<http://storage.googleapis.com/books/ngrams/books/datasetv2.html>

- ✓ 파일은 키워드, 년도, 빈도수, 책수의 형식으로 되어 있음
- ✓ 다운 받은 파일을 hdfs의 /tmp/google_1gram.csv 로 복사하고 확인함

```
> hadoop fs -put googlebooks-eng-all-1gram-20120701-s  
/tmp/google_1gram_s.csv  
> hadoop fs -ls /tmp
```

- ✓ 이 방대한 파일에서 statistically_ADV의 수를
년도별로 세어보고 이를 시각화 해보자.

statistic.7	2002	1	1
statistic.7	2003	2	2
statistic.7	2004	2	2
statistic.7	2005	2	2
statistic.7	2006	3	3
statistic.7	2007	3	3
statistic.7	2008	3	3
statistically_ADV		1600	1
statistically_ADV		1766	2
statistically_ADV		1798	1
statistically_ADV		1799	1

4.5 예제: 구글 Ngram 시각화

```
inputfile = „/tmp/google_1gram_s.csv“
```

```
input.format <- make.input.format(  
  format = „csv“, sep = „\t“,  
  col.names = c(„ngram“, „year“, „occurrences“, „book“),  
  colClasses = c(„character“, „integer“, „integer“, „integer“)  
)
```

```
mapper <- function(keys, values) {  
  values$ngram <- tolower(values$ngram)  
  T.or.F <- values$ngram %in% c(„statistically_adv“)  
  keys <- values$ngram[T.or.F]  
  values <- values[T.or.F, c(„year“, „occurrences“)]  
  keyval(keys, values)  
}
```

```
reducer <- function(key, values){  
  val <- tapply(values$occurrences, values$year, sum)  
  val <- data.frame(year=as.integer(names(val)), occurrences = val)  
  keyval(key, val)  
}
```

statistically_adv이면
TRUE 아니면 FALSE

TRUE인 결과만 데이터
프레임으로 만듦

년도별 occurrences 합을
구해 데이터프레임 저장

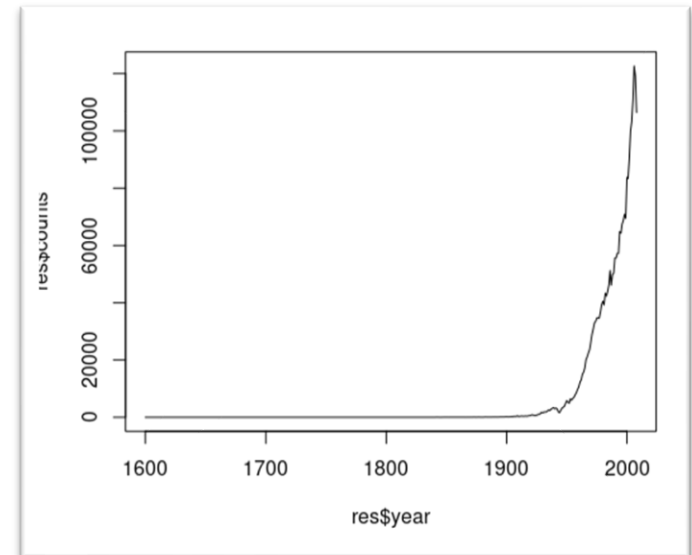
RHadoop_Ngram.R

4.5 예제: 구글 Ngram 시각화

- ✓ 맵리듀스 결과를 구하고 결과 파일은 /tmp/temp11에 저장한다.
- ✓ 년도별 출현 빈도를 데이터프레임으로 만들고 plot을 그린다.
- ✓ 통계학은 1950년도를 지나면서 급격하게 발전하였다. 2010년을 기점으로 꺾이는 모습임

```
job <- mapreduce(
  input=inputfile,
  input.format=input.format,
  output="/tmp/temp11",
  map= mapper,
  reduce=reducer
)

out = from.dfs(job)
out$val$year
out$val$occurrences
res <- data.frame(out$val$year, out$val$occurrences)
names(res) <- c("year","counts")
plot(res$year, res$counts, type="l")
```



4.6 예제: Airline Data

- ✓ Airline 데이터는 1987년부터 해마다 29개의 변수를 조사한 데이터임
- ✓ 2008.csv 파일을 대상으로 항공사 코드(UniqueCarrier) 별 지연시간을 분석해보자.
- ✓ > Hadoop fs -put 2008.csv /tmp 명령으로 데이터를 hdfs로 전송

1 Year	1987-2008		
2 Month	1-12	16 DepDelay	departure delay, in minutes
3 DayofMonth	1-31	17 Origin	origin IATA airport code
4 DayOfWeek	1 (Monday) - 7 (Sunday)	18 Dest	destination IATA airport code
5 DepTime	actual departure time (local, hhmm)	19 Distance	in miles
6 CRSDepTime	scheduled departure time (local, hhmm)	20 TaxiIn	taxi in time, in minutes
7 ArrTime	actual arrival time (local, hhmm)	21 TaxiOut	taxi out time in minutes
8 CRSArrTime	scheduled arrival time (local, hhmm)	22 Cancelled	was the flight cancelled?
9 UniqueCarrier	unique carrier code	23 CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
10 FlightNum	flight number	24 Diverted	1 = yes, 0 = no
11 TailNum	plane tail number	25 CarrierDelay	in minutes
12 ActualElapsedTime	in minutes	26 WeatherDelay	in minutes
13 CRSElapsedTime	in minutes	27 NASDelay	in minutes
14 AirTime	in minutes	28 SecurityDelay	in minutes
15 ArrDelay	arrival delay, in minutes	29 LateAircraftDelay	in minutes

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7>

4.6 예제: Airline Data

- ✓ key.df는 날짜와 항공사를 저장하고 val.df는 출발지연시간을 저장한다.

```
inputfile="/tmp/2008.csv"

input.format <- make.input.format(
  format = "csv", sep=",",
)

mapper = function(., fields) {
  DepDelay = as.numeric(as.character(fields[[16]]))
  filter = !is.na(DepDelay)
  Year = as.character(fields[[1]])
  Month = as.character(fields[[2]])
  Dayofmonth = as.character(fields[[3]])
  UniqueCarrier = as.character(fields[[9]])
  key.df = data.frame( date = paste(Year[filter], Month[filter], Dayofmonth[filter],
sep='-'), UniqueCarrier = UniqueCarrier[filter] )
  val.df = data.frame( DepDelay = DepDelay[filter] )
  val.df$DepDelay = as.numeric(val.df$DepDelay)
  keyval( key.df , val.df )
}
```

4.6 예제: Airline Data

- ✓ key별로 출발지연시간의 평균을 저장한 후, 데이터프레임으로 만든다.

```
reducer = function(key, value) {  
  value = data.frame(avg = mean(value$DepDelay, na.rm=T))  
  keyval(key, value)  
}  
  
job = mapreduce(  
  input = inputfile,  
  input.format = input.format,  
  output = "/tmp/temp3",  
  map = mapper,  
  reduce = reducer  
)  
out = from.dfs(job)  
out1 <- out$key  
out2 <- out$val  
result = as.data.frame(out)  
colnames(result) = c("date", "carrier", "mean of depDelay")  
rownames(result) = NULL  
result
```

5. 빅데이터 통계 모형

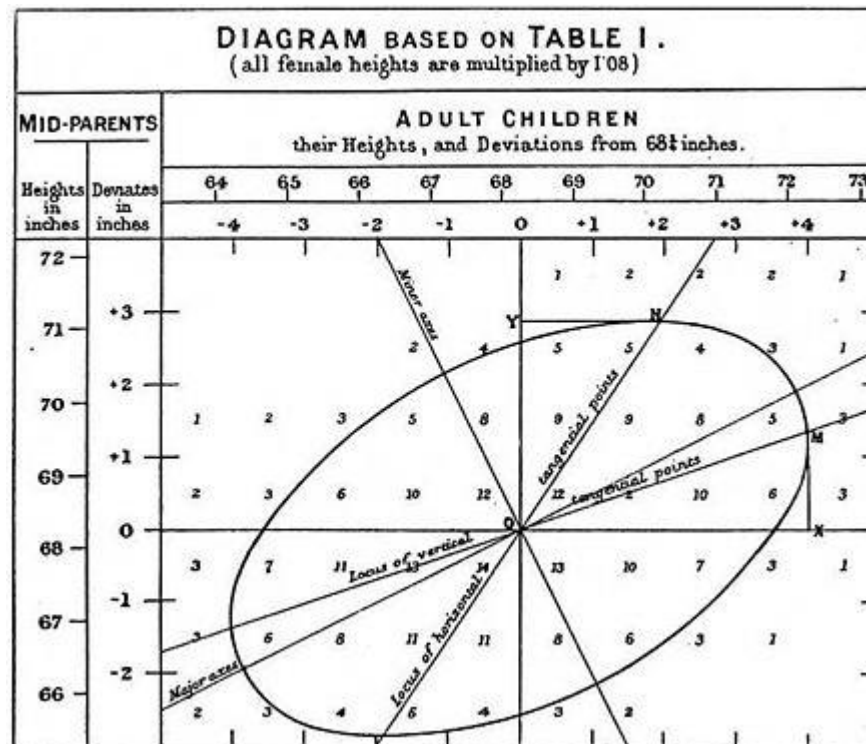
5.1 회귀 분석

5.2 로지스틱 회귀분석

5.3 k means 군집분석

5.1 회귀분석

[Francis Galton's](#) 1875 illustration of the correlation between the heights of adults and their parents. The observation that adult children's heights tended to deviate less from the mean height than their parents suggested the concept of "[regression toward the mean](#)", giving regression its name.



5.1 회귀분석



그녀가 원하는 것을 알 수 있는 함수가 있다면?



Function Notation

$$y = f(x)$$

Output

Name of
Function

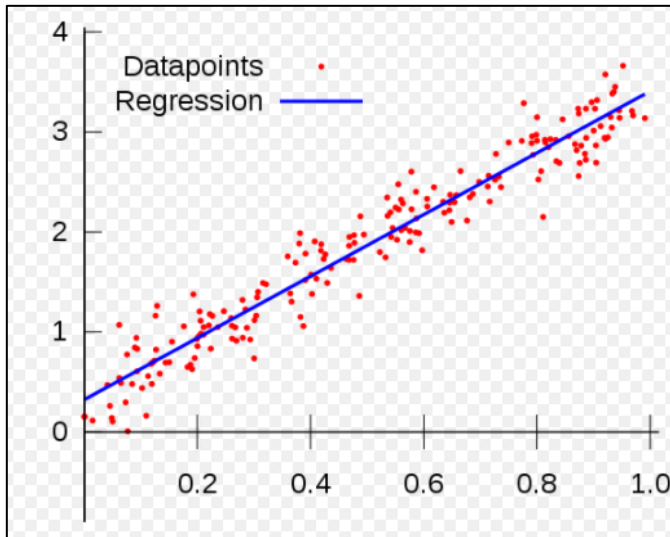
Input



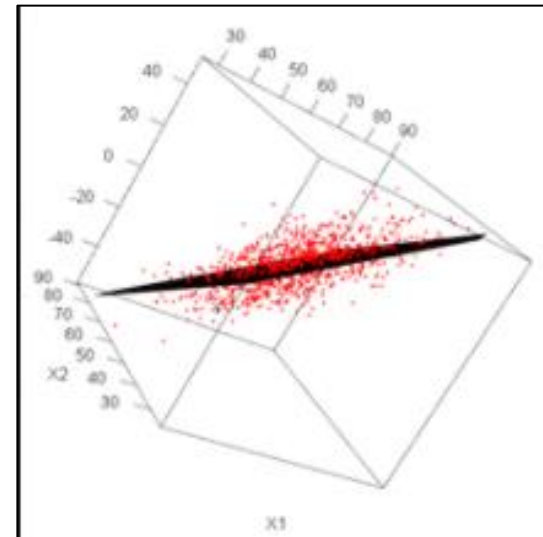
면접 시, 원하는 것을 알 수 있는 함수가 있다면?

5.1 회귀분석

회귀분석은 관측 값을 가장 잘 지나가는 직선 혹은 곡선의 방정식을 구하는 방법론



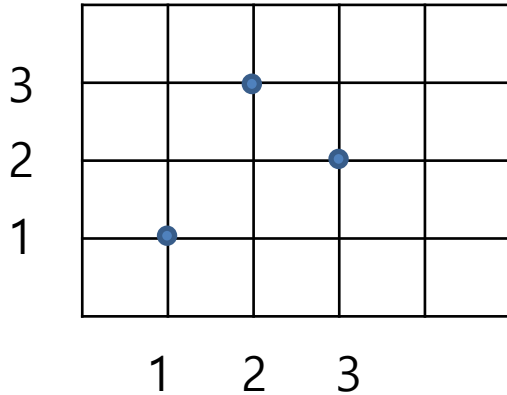
$$Y = \alpha + \beta x + \epsilon$$



$$Y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

직선으로 예측했을 경우, 오차 ϵ 이 존재, 오차가 작을 수록 좋은 모형임

5.1 회귀분석



i	x	y
1	1	1
2	2	3
3	3	2

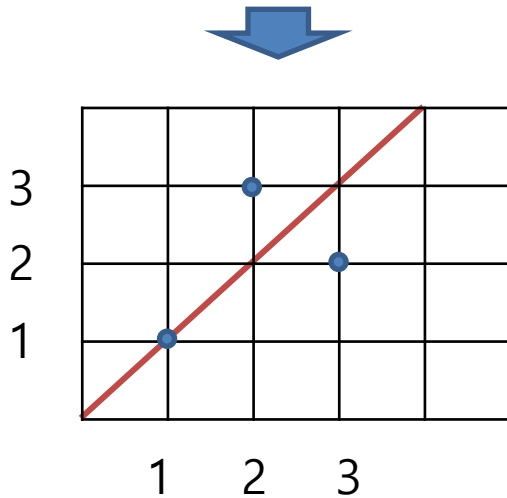
$$y_i = \alpha + \beta x_i + \epsilon_i$$



$$1 = \alpha + \beta * 1 + \epsilon_1$$

$$3 = \alpha + \beta * 2 + \epsilon_2$$

$$2 = \alpha + \beta * 3 + \epsilon_3$$



$$\begin{aligned}\hat{y}_i &= \alpha + \beta x_i \\ &= 0 + 1 \cdot x_i\end{aligned}$$

$$1 = 0 + 1 * 1 + \epsilon_1, \epsilon_1 = 0$$

$$3 = 0 + 1 * 2 + \epsilon_2, \epsilon_2 = 1$$

$$2 = 0 + 1 * 3 + \epsilon_3, \epsilon_3 = -1$$

$$\sum_{i=1}^3 |\epsilon_i| \text{ 혹은 } \sum_{i=1}^3 \epsilon_i^2 \text{ 를 최소화하는 } \alpha, \beta = ?$$

5.1 회귀분석

$$\sum_{i=1}^n x_i = x_1 + x_2 + \cdots + x_n$$

$$\sum_{i=1}^n c = c + c + \cdots + c = n \cdot c$$

$$\sum_{i=1}^n c \cdot x_i = c \cdot x_1 + c \cdot x_2 + \cdots + c \cdot x_n = c \sum_{i=1}^n x_i$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n x_i^2 - n \cdot \bar{x}^2$$

$$\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^n x_i y_i - n \cdot \bar{x} \cdot \bar{y}$$

5.1 회귀분석

- ✓ 회귀모형에서 미지수는 SSE를 최소화하는 미지수 알파, 베타를 구하는 계산을 수행하여 얻어짐
(SSE를 일반적으로 cost 함수라고 함)

$$SSE = \sum (\varepsilon_i^2) = \sum (Y_i - \alpha - \beta x_i)^2$$

- Least Square Estimation for α, β

$$\frac{\partial SSE}{\partial \alpha} = 2 \sum (Y_i - \alpha - \beta x_i)(-1) = 0 \dots\dots\dots ①$$

$$\frac{\partial SSE}{\partial \beta} = 2 \sum (Y_i - \alpha - \beta x_i)(-x_i) = 0 \dots\dots\dots ②$$

①× \bar{x} 를 하면,

$$\alpha \sum x_i + \beta \bar{x} \sum x_i = n\bar{x}\bar{Y} \dots\dots\dots ③$$

$$\alpha \sum x_i + \beta \sum x_i^2 = \sum x_i Y_i \dots\dots\dots ④$$

④-③을 하면,

$$\hat{\beta} = \frac{\sum x_i Y_i - n\bar{x}\bar{Y}}{\sum x_i^2 - n\bar{x}^2} = \frac{\sum (x_i - \bar{x})(Y_i - \bar{Y})}{\sum (x_i - \bar{x})^2}, \quad \hat{\alpha} = \bar{Y} - \hat{\beta}\bar{x}$$

5.1 회귀분석

다중 회귀모형은 종속변수에 영향을 주는 다수의 독립변수가 존재하는 경우의 회귀모형임

예를 들어, 연봉을 종속변수로 볼 때, 연봉에 영향을 주는 요인으로 나이, 성별, 학력, 직업, 월 근무시간 등이 있을 것이다. 이를 수식으로 표현하면 아래와 같다.

$$\text{연봉} = \beta_0 + \beta_1 \text{나이} + \beta_2 \text{성별} + \beta_3 \text{학력} + \beta_4 \text{직업} + \beta_5 \text{월근무시간} + \epsilon$$

이를 일반화하여 표현하면 아래와 같다.

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i, \epsilon_i \sim N(0, \sigma^2), i = 1, \cdots, n \quad \Rightarrow \quad Y = X \cdot \beta + \epsilon$$

나이, 성별, 학력, 직업, 월 근무시간

$$X = \begin{pmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}$$

연봉

5.1 회귀분석

여기서 미지 모수 β 는 아래와 같이 구할 수 있다.

$$\begin{aligned}SSE &= (Y - X\beta)'(Y - X\beta) \\&= Y'Y - Y'X\beta - \beta'X'Y + \beta'X'X\beta \\&= Y'Y - 2\beta'X'Y + \beta'X'X\beta\end{aligned}$$

$$\frac{\partial SSE}{\partial \beta} = 2(X'X)\hat{\beta} - 2(X'Y) = 0$$

$$\therefore \hat{\beta} = (X'X)^{-1}X'Y$$

5.1 회귀분석

- ✓ 회귀계수 베타는 아래와 같이 n 개의 관측값을 나누어 각 행단위 연산의 합으로 표현 가능함
- ✓ n 개의 데이터 뭉치를 한 개씩 나누어 계산하여 합하면 엄청나게 큰 n 에 대해 회귀연산을 수행할 수 있음. 이 때, 분산처리를 하면 속도가 빨라짐.

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \left(\sum \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left(\sum \mathbf{x}_i y_i \right)$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} \quad \sum \begin{pmatrix} 1 \\ x_{1i} \\ x_{2i} \end{pmatrix} \begin{pmatrix} 1 & x_{1i} & x_{2i} \end{pmatrix} = \sum \begin{pmatrix} 1 & x_{1i} & x_{2i} \\ x_{1i} & x_{1i}^2 & x_{1i}x_{2i} \\ x_{2i} & x_{1i}x_{2i} & x_{2i}^2 \end{pmatrix}$$



$$X'X = \begin{bmatrix} N & \sum x_1 & \sum x_2 \\ \sum x_1 & \sum x_1^2 & \sum x_1 x_2 \\ \sum x_2 & \sum x_2 x_1 & \sum x_2^2 \end{bmatrix} \quad X'y = \begin{bmatrix} \sum y \\ \sum X_1 y \\ \sum X_2 y \end{bmatrix}$$

5.1 회귀분석

- ✓ 아래는 난수로 10개의 변수와 200개의 관찰치를 가진 행렬 X를 만들고, 첫 컬럼에 1을 더해 200×11 행렬을 만듦. Y역시 200개의 관찰치 행렬로 200×1 임
- ✓ R에서는 lm 명령으로 회귀계수를 구할 수 있음

```
# Defining data variables
X = matrix(rnorm(2000), ncol = 10)
X_mat = cbind(1,X)
y = as.matrix(rnorm(200))

# Bundling data variables into dataframe
train_data <- data.frame(X,y)

# Training model for generating prediction
lmodel<- lm(y~ train_data $X1 + train_data $X2 + train_data $X3 + train_data $X4
+ train_data $X5 + train_data $X6 + train_data $X7 + train_data $X8 + train_data
$X9 + train_data $X10,data= train_data)
summary(lmodel)
```

regression.R

5.1 회귀분석

- ✓ X 는 $n \times p$ 로 큰 행렬이지만, $X^t X$ 는 $p \times p$ 행렬로 작아짐.
- ✓ Map을 이용해 $\text{list}(t(X_i) \%*\% X_i)$ 를 계산하고 Reduce를 이용해 Sum을 계산
- ✓ `cbind(1:nrow(X_mat), X_mat)` 명령을 통해 X 행렬에 번호를 붙여 놓음

```
X.index = to.dfs(cbind(1:nrow(X_mat), X_mat))
```

```
XtX =  
  values(  
    from.dfs(  
      mapreduce(  
        input = X.index,  
        map =  
          function(., Xi) {  
            Xi = Xi[,-1]  
            keyval(1, list(t(Xi) \%*\% Xi)),  
          },  
        reduce =  
          function(., YY)  
            keyval(1, list(Reduce('+', YY)))  
      )))[1]
```

Reduce는 리스트 대해 주어진
연산을 하는 R 함수임

5.1 회귀분석

- ✓ Xty 는 $p \times 1$ 행렬로 작음.
- ✓ Map을 이용해 $\text{list}(t(X_i) \%*\% y_i)$ 를 계산하고 Reduce를 이용해 Sum을 계산
- ✓ 여기서, y_i 는 X 행렬이 분할되어 X_i 가 만들어질 때, 쌍이 되는 y_i 임
- ✓ X 행렬 번호를 이용해 Y 로 부터 y_i 를 추출함

```
Xty = values(  
  from.dfs(  
    mapreduce(  
      input = X.index,  
      map = function(., Xi) {  
        yi = y[Xi[,1],]  
        Xi = Xi[,-1]  
        keyval(1, list(t(Xi) \%*\% yi)),  
      reduce =  
        function(., YY)  
          keyval(1, list(Reduce('+', YY)))  
      )[[1]]  
    solve(XtX) \%*\% Xty
```

solve는 역행렬 함수임

5.2 로지스틱 회귀분석

예측하고자 하는 종속변수가 Binary 일 경우,
예를 들어, $Y=0$ 은 정상 제품, $Y=1$ 은 불량 제품을 나타내고, $X = (1, 2, 1)$ 은 부품의 속성을 나타내는 독립변수이라고 하자. 아래의 Data에서는 $X = (1, 2, 1)$ 일 때, $Y=0$ 이므로 정상제품임을 의미한다.

$$X = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 3 & 4 \\ 1 & 5 & 5 \\ 1 & 7 & 5 \\ 1 & 2 & 5 \end{pmatrix}, Y = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} \quad \text{여기서, } \beta \text{는 미지수임}$$

$$\ln\left(\frac{p_i}{1-p_i}\right) = X \cdot \beta + \epsilon, \quad p_i = \Pr(Y = 1)$$

우리의 목표는 베타들을 구하고 $X=(1, 2, 4)$ 와 같이 새로운 제품이 들어 왔을 때, 이 제품이 불량인지 아닌지 로지스틱 회귀모형을 통해 판단해보는 것이다.

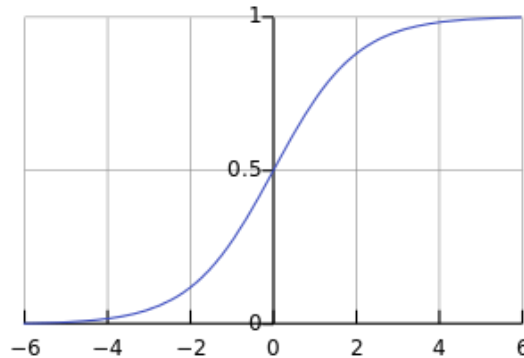
5.2 로지스틱 회귀분석

로지스틱 회귀모형은 회귀모형에서 종속변수 $Y=0,1$ 과 같이 Binary Factor 변수일 경우의 모형임

일반적인 회귀모형 $Y = \alpha + \beta X + \epsilon$ 으로 표현할 경우, 좌변의 범위는 $(0, 1)$ 이고 우변은 $(-\infty, \infty)$ 인 문제가 발생한다. 이 문제를 해결하기 위해 로지스틱 함수를 사용한다.

로지스틱 함수는 0~1 사이에 값을 가지는 함수로 S 혹은 Sigma 모양과 비슷하여 Sigmoid 함수 라고도 한다. 우변에 Logistic 함수를 적용하여 양변의 스케일 문제를 해결할 수 있다.

이처럼 Logistic 함수를 이용하여 회귀분석을 수행하는 것이 Logistic Regression(Cox, 1958)이라고 부른다.



$$Y = \frac{1}{1 + e^{-x}}$$

$$Y = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_k x_{ki} + \epsilon))} = \frac{1}{1 + \exp(-(\beta X + \epsilon))}$$

$$\log\left(\frac{p}{1-p}\right) = X\beta + \epsilon = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + \epsilon \quad p = \Pr(Y = 1)$$



5.2 로지스틱 회귀분석

로지스틱 회귀모형은 아래와 같은 수식으로 표현 가능하다.

$$Y = \frac{1}{1 + \exp(-\beta X + \epsilon)}$$

$$\frac{Y}{1 - Y} = \frac{\frac{1}{1 + \exp(-\beta X + \epsilon)}}{\frac{\exp(-\beta X + \epsilon)}{1 + \exp(-\beta X + \epsilon)}} = \frac{1}{\exp(-\beta X + \epsilon)} = \exp(\beta X + \epsilon)$$

$\log\left(\frac{p}{1-p}\right) = \beta X + \epsilon = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon$ 로 표현한다.

$\hat{\beta}$ 는 최대가능도 추정으로 경사 하강법을 사용하여 구한다.

$\hat{\beta}$ 이 주어졌을 때, Y 에 대한 추정 값은 아래의 식으로 구할 수 있다.

$$\hat{Y} = \frac{1}{1 + \exp(-\hat{\beta} X)} = \frac{1}{1 + \exp(-\hat{\beta} X)} \cdot \frac{\exp(\hat{\beta} X)}{\exp(\hat{\beta} X)} = \frac{\exp(\hat{\beta} X)}{1 + \exp(\hat{\beta} X)}$$

5.2 로지스틱 회귀분석

Cross Entropy 함수는 Logistic Regression의 Maximum Likelihood 함수다.
최대가능도 함수(Maximum Likelihood function)에 대해 알아보자.

$Y_i = \frac{1}{1+e^{(-\beta_0-\beta_1x_i+\epsilon)}}$ 의 식에서 Y_i 는 0 혹은 1의 값을 갖는 확률변수다.

그러므로 $Y_i \sim \text{Bernoulli}(p_i)$ 이다. 여기서, $p_i = \Pr(Y_i = 1)$ 이고 확률질량함수(P.M.F.)는 아래와 같다.

$$f(y) = P(Y = y) = p_i^y(1 - p_i)^{1-y}$$

만약, $Y_i|X_i = [y_1, y_2, \dots, y_n]$ 이 관측되었다면 이렇게 관측될 확률은 아래와 같다.

$$(p_i^{y_1}(1 - p_i)^{1-y_1}) \times (p_i^{y_2}(1 - p_i)^{1-y_2}) \times \dots \times (p_i^{y_n}(1 - p_i)^{1-y_n}) = \prod_{i=1}^n p_i^{y_i}(1 - p_i)^{1-y_i}$$

이 수식을 미지 모수 p_i 에 대한 최대가능도함수라고 한다.

위 수식에서 y_i 는 주어진 값이고 미지수는 오직 p_i 뿐이다.

이 함수를 최대화 하는 p_i 가 가장 관측값을 잘 설명하는 값이라고 생각하는 것이 MLE 추정(Maximum Likelihood Estimation)이다.

5.2 로지스틱 회귀분석

p_i 는 y_i 에 대한 추정값으로 $p_i > 0.5 \rightarrow y_i = 1$ 아니면 $y_i = 0$ 이 되므로 아래와 같은 식이 성립한다.

$$p_i = \frac{1}{1 + e^{-\hat{\beta}_0 - \hat{\beta}_1 x_i}}, \quad L(p_i) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}$$

이 식을 최대화 하는 β_0, β_1 이 우리가 구하고 하자 하는 미지수이고 이를 MLE(Maximum Likelihood Estimator) 라고 한다. Likelihood 함수에 로그를 취하면, 아래와 같다.

$$\begin{aligned} \ln L(\beta_0, \beta_1) &= \sum_{i=1}^n y_i \ln \frac{1}{1 + e^{-\hat{\beta}_0 - \hat{\beta}_1 x_i}} + (1 - y_i) \ln \left(1 - \frac{1}{1 + e^{-\hat{\beta}_0 - \hat{\beta}_1 x_i}} \right) \\ &= \sum_{i=1}^n y_i \ln p_i + (1 - y_i) \ln (1 - p_i) \end{aligned}$$

이 식에 “-1”을 곱한 식이 Cross Entropy 함수다.

최대가능도 함수를 최대화하는 것이나 Cross Entropy 함수를 최소화하는 것 이나 같은 것이다.

이 함수를 최소화하는 β_0, β_1 를 구하면 MLE가 된다.

5.2 로지스틱 회귀분석

문제는 Cross Entropy 함수를 최소화하는 미지 모수를 수학적으로 구할 수 없다는 것이다.

이러한 문제를 수치해석적 방법으로 해결하는 알고리즘이 최대 경사 하강법(Steepest Descending Algorithm) 이다.

Convex Function에서 기울기가 “0” 인 지점을 찾아가기 위해 임의의 초기값 W 에서 기울기 방향으로 조금씩(learning rate: α) 움직여 가는 방법을 사용한다.

즉, 임의의 초기값에서 출발하여 기울기 방향으로 움직이면 언젠가는 기울기가 “0” 인 지점에 도착한다.

$$Cost(w, b) = - \sum \left[y_i \log(S(wx_i + b)) + (1 - y_i) \log(1 - S(wx_i + b)) \right]$$

$$\frac{\partial}{\partial b} Cost(w, b) = - \sum \left[y_i - S(wx_i + b) \right]$$

$$\frac{\partial}{\partial w} Cost(w, b) = - \sum \left[y_i - S(wx_i + b) \right] x_i$$

$$w = w - \lambda \frac{\partial}{\partial w} Cost(w, b)$$

$$b = b - \lambda \frac{\partial}{\partial b} Cost(w, b)$$

하둠에서는 위 식에서 sum 부분을 분산처리로 구할 수 있다.

5.2 로지스틱 회귀분석

- ✓ foodstamp 데이터는 저소득층에 제공하는 식량보조를 받는가 여부를 측정한 것이다.
- ✓ TEN: 차용여부, SUP: 사이드 job 여부, INC: 로그를 취한 월수입이다.
- ✓ 이 데이터에서 TEN, SUP, INC로 y 를 예측하는 회귀모형을 mapReduce로 만들자.

```
library(dplyr)
library(catdata)
data(foodstamp)
stampData <- foodstamp
stampData[,4] <- scale(stampData[,4])
g = function(z) {1/(1 + exp(-z))}

map1 <- function(k,v){keyval(1:(dim(v)[1]), v)}
reduce1 <- function(k,v){
  Y <- v[1]
  X <- v[-1]
  theta <- g(plane %*% X)
  temp <- (theta - Y) * X
  keyval(k, t(temp))
}
```

logisticRegression.R

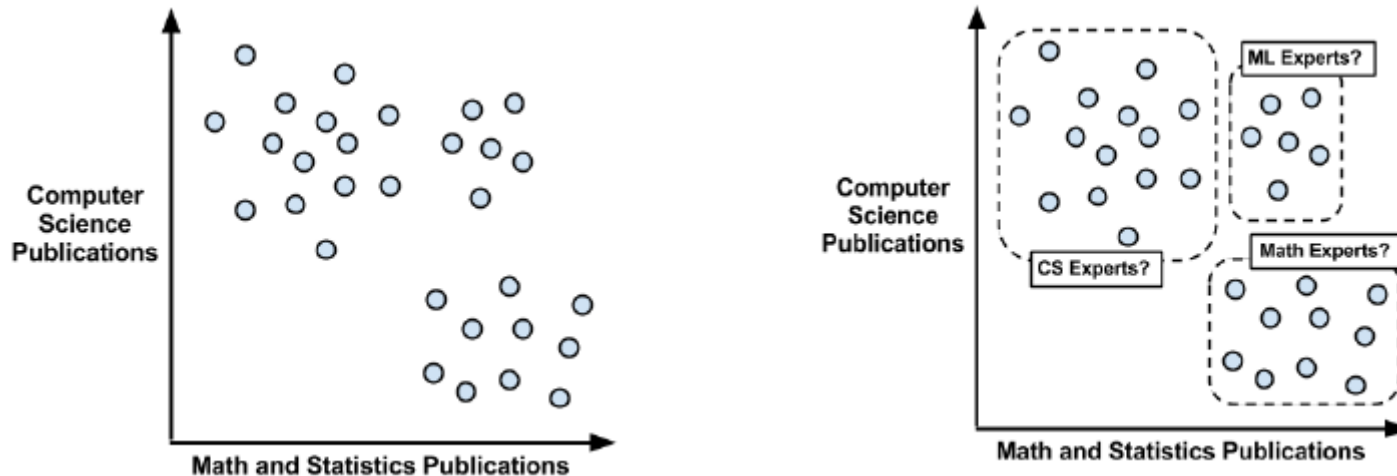
5.2 로지스틱 회귀분석

```
map2 <- function(k,v){keyval(1, v)}
reduce2 <- function(k,v){
  keyval(k, apply(v,2, sum))
}
stampData_hdfs <- to.dfs(as.matrix(data.frame(y=stampData[,1], intercept=1,
stampData[,-1])))
plane <- t(rep(0,4))

for (i in 1:10){
  result <- mapreduce(input = stampData_hdfs, map = map1, reduce =
reduce1) %>% mapreduce(map = map2, reduce = reduce2)
  gradient <- values(from.dfs(result))
  plane <- plane - 0.05 * gradient
  print(plane)
}
plane
```

5.3 k-mean 군집분석

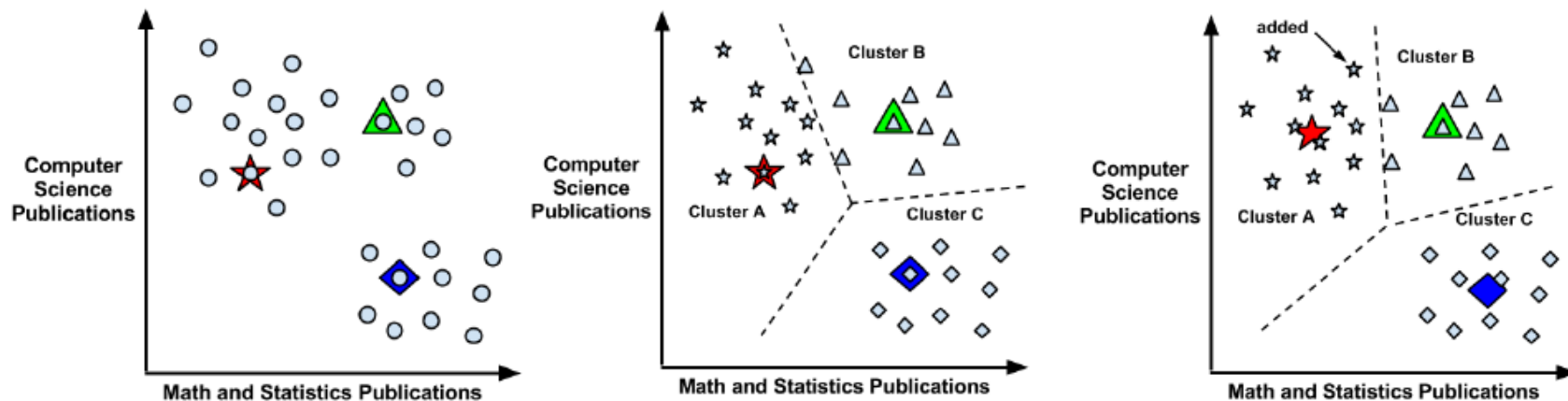
- ✓ 군집화는 유사한 개체끼리 묶어주는 방법론이다.
- ✓ 군집화를 활용하여 마케팅에서는 유사한 고객끼리 묶어 세그먼트를 나누기도 하고, 복잡한 데이터를 몇개의 범주로 간단하게 만들기도 한다.
- ✓ 아래의 그림은 대학교수들이 발표한 두 분야의 논문 수에 대한 도표이다.



위 그림으로 우리는 각 군집의 성격을 유추하는 것이 가능하다. 이러한 유추를 Un-Supervised Learning이라고 한다.

5.3 k-mean 군집분석

K-Means Algorithm은 원하는 군집의 수를 k 개할 때, k 개의 개체를 무작위로 선정하고 이를 초기 군집의 중앙값으로 선정한다. 이후, 군집중앙값과 각 개체의 거리를 계산하여 가까운 군집에 개체를 편입시킨다. 이후, 군집에 속한 개체의 중심을 구하고 그 값으로 군집중앙값을 대체한다. 이렇게 조정된 군집 중앙값을 이용하여 각 개체와의 거리를 재계산하고 가까운 군집으로 개체를 편입을 조정하는 방식이다. 이 방식으로 반복하여 더 이상 개체들이 속한 군집에 변화가 없으면 종료한다.

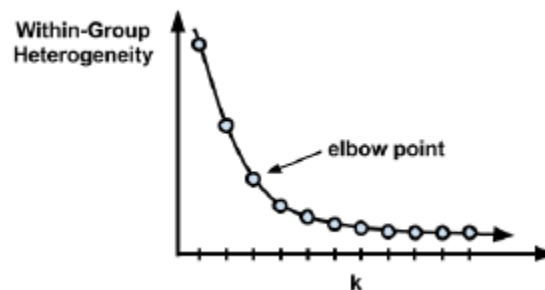
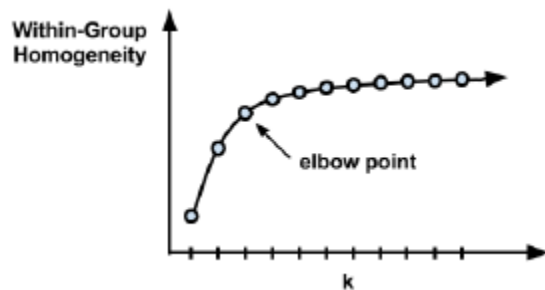


Lantz, Machine Learning with R, PACKT 참조

5.3 k-mean 군집분석

Choosing the appropriate number of clusters

군집의 수는 다루는 문제의 성격에 따라 정해질 수도 있고, 사전 지식이 없을 경우, Data에 의해 정해질 수도 있다. Data로 부터 정하는 방법으로는 그룹내 동질성 혹은 이질성을 군집의 수에 따라 plotting해서 변곡점을 찾는 방식이다.



Lantz, Machine Learning with R, PACKT 참조

5.3 k-mean 군집분석

같은 방식으로 정리한 결과이다.

Cluster 1 (N = 3,376)	Cluster 2 (N = 601)	Cluster 3 (N = 1,036)	Cluster 4 (N = 3,279)	Cluster 5 (N = 21,708)
swimming cheerleading cute sexy hot dance dress hair mall hollister abercrombie shopping clothes	band marching music rock	sports sex sexy hot kissed dance music band die death drunk drugs	basketball football soccer softball volleyball baseball sports god church Jesus bible	???
Princesses	Brains	Criminals	Athletes	Basket Cases

Lantz, Machine Learning with R, PACKT 참조

1번 군집의 이름을 공주, 2번 군집은 music, 3번 군집은 노는아이들, 4번 군집은 운동선수로 규정할 수 있다. 이러한 개체들의 성격 규정은 이들을 대상으로 하는 1:1 마케팅에 중요한 정보로 사용된다.

5.3 k-mean 군집분석

- ✓ k means 군집분석에서 각 개체는 k개의 중심값과 거리를 계산해야 수차례 반복적으로 계산
- ✓ 하둡은 거리를 계산하고 계산결과에 의해 군집을 결정하는 작업을 분산처리하여 속도를 개선함

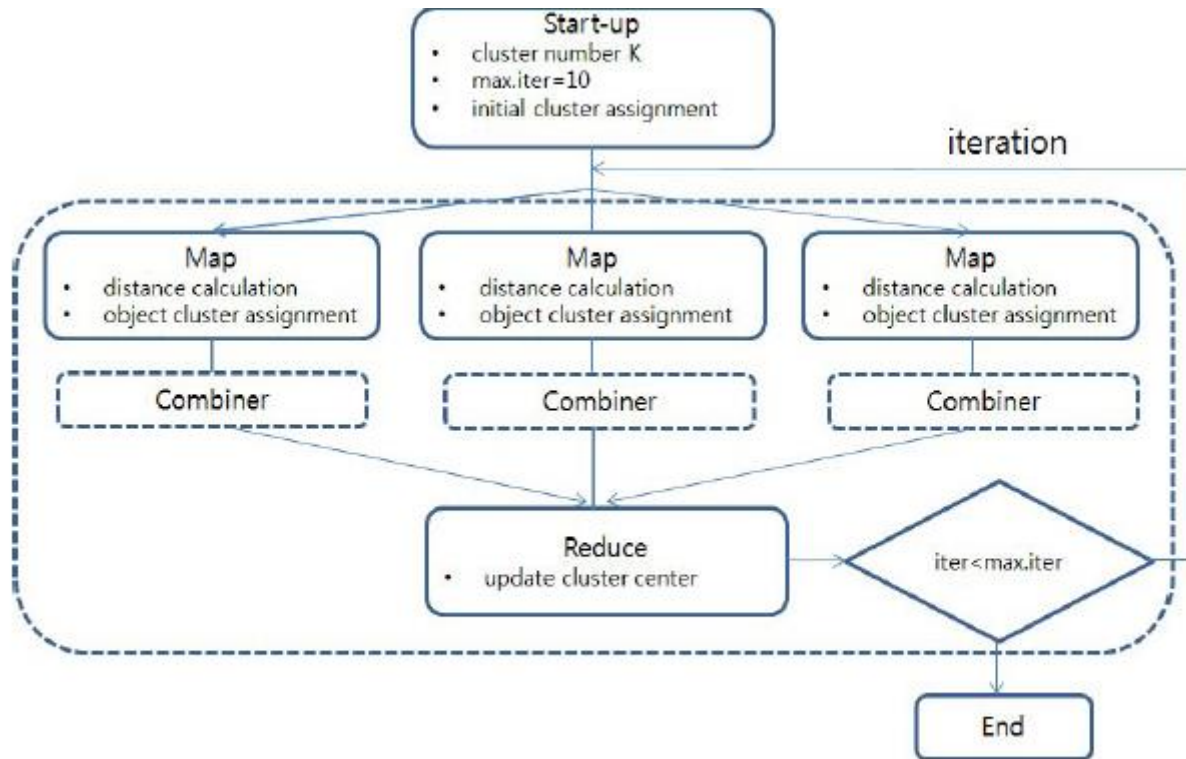


Figure 4.2 MapReduce K-means clustering flowchart

참조: 신지은, 오윤식, 임동훈, 빅데이터 K-평균 클러스터링을 위한 RHadoop 플랫폼, 한국데이터정보과학회지, 2016

5.3 k-mean 군집분석

- ✓ mapper는 P의 모든 개체에 대해 가장 가까운 C를 찾아 C와 P를 묶어 리턴함
- ✓ reducer는 mapper가 리턴한 C별 P값의 평균을 계산하여 C를 갱신함

```
dist.fun <- function(C, P) apply(C, 1, function(x) colSums((t(P) - x)^2))
```

```
## kmeans.map: return nearest centroid
```

```
kmeans.map <- function(., P) {  
  nearest <-  
    if(is.null(C))  
      sample(1:num.clusters, nrow(P), replace = TRUE)  
  else {  
    D <- dist.fun(C, P)  
    nearest <- max.col(-D)  
  }  
  keyval(nearest, P)  
}
```

```
## kmeans.reduce: compute the centroid mean by key
```

```
kmeans.reduce <-  
  function(., P) t(as.matrix(apply(P, 2, mean)))
```

5.3 k-mean 군집분석

- ✓ P는 데이터 포인트로 이 예에서는 (100,2) 행렬임
- ✓ C는 12개 군집의 중심점으로 5회 반복하면서 군집의 위치를 갱신함

```
P <- do.call(rbind, rep(list(matrix(rnorm(10, sd = 10), ncol=2)),20)) +  
              matrix(rnorm(200), ncol =2)  
  
C <- NULL  
P_hdfs <- to.dfs(P)  
num.clusters <- 12  
num.iter <- 5  
for(i in 1:num.iter ) {  
  C <- values(from.dfs(  
    mapreduce(P_hdfs, map = kmeans.map, reduce = kmeans.reduce)  
  ))  
  if(nrow(C) < num.clusters) {  
    C <-rbind(C, matrix(  
      rnorm((num.clusters - nrow(C)) * nrow(C)),  
      ncol = nrow(C)) %*% C  
    )  
  }  
}  
C
```

5.3 k-mean 군집분석

- ✓ 모든 포인트 P에 대해 거리가 최소가 되는 군집에 Assign 한다.

```
output <- mapreduce(P_hdfs, map = kmeans.map)
result <- from.dfs(output)
clusterRes <- as.data.frame(result)
```

key	val.1	val.2
1	13.393136	-15.81202031
1	12.218032	-14.34574956
1	12.405275	-15.51829244
1	13.133151	-13.81940874
1	12.393836	-13.88782814
1	13.781455	-16.35533952
1	14.387446	-14.65822411
1	11.563952	-16.16729060
1	12.895680	-13.92924342
1	12.879684	-15.65168358
1	12.988272	-15.08161468
1	12.871806	-14.68625990
1	12.648699	-13.51631788
1	13.730512	-14.80052527
1	11.832453	-15.46991780
1	12.210563	-15.48995654

5	5.142958	23.55570114
5	3.016029	24.82216821
6	13.159977	0.43712404
6	13.556340	0.41406465
6	11.928954	2.31204818
6	12.608468	1.62260648
6	12.504778	0.92344443
6	12.770874	0.53616922
7	14.454530	-3.39537870
7	15.739357	-3.04372939
7	14.070135	-3.53942984
7	14.753981	-2.92956792
7	14.923440	-2.13064802
12	5.824712	23.54515557

kmeans.R

6. 실전 분석

6.1 통계청 인구 센서스 기초 통계 분석하기

6.2 소상공인 실태조사 분석하기

6.1 인구센서스 통계분석 하기

- ✓ 2015년 인구센서스 자료를 <https://mdis.kostat.go.kr/index.do> 에서 받는다.
- ✓ 이 데이터로부터 자전거로 출퇴근하는 사람의 비율을 구해보자.

 **2015 POPULATION AND HOUSING CENSUS**
인구주택총조사

함께해요

• 이 조사는 「통계법 제5조의 3」에 의하여 실시되며, 「제32조」에 따라 모든 국민은 성실히 응답할 의무가 있습니다.
• 통계작성 과정에서 알려진 비밀사항은 「통계법 제33조」에 따라 보호됩니다.

☞ 조사표 작성방법

- 필기구는 검은색 볼펜 사용
- 문자는 네모 칸 안에 정자체로 기입 **부산광역시**
- 숫자는 오른쪽 칸에 맞추어 기입 **5 2** 세
- 선택 문항은 ●로 표시 **남자** **여자**
- 는 조사원이 기입하는 칸임

홈페이지 www.census.go.kr 문의 전화(무료) 080-200-2015

비 고

2015년 11월 1일 0시 현재, 혈연이나 주민등록과 관계없이 이 가구에서 실제로 같이 살고 있는 모든 사람을 대상으로 조사합니다.

이 가구에서 조사 대상으로 할 사람

- 출장, 친지 방문, 여행, 입원, 환자 간호 등으로 잠시 집을 떠나 있는 사람
- 예비군 훈련 또는 공익근무요원으로 근무 중인 사람, 군부대 밖에서 거주하는 직업군인
- 선박, 항공기, 철도, 시외버스·관광버스 등의 탑승 승무원
- 숙식을 함께하는 가사도우미와 하숙인
- 한국인과 함께 사는 외국인

이 가구가 아닌 다른 곳에서 조사 대상으로 할 사람

- 군인, 의무경찰 등으로 입대된 가족 명
- 학업 때문에 따로 살고 있는 가족 국내 명 해외 명
- 직장 때문에 따로 살고 있는 가족 국내 명 해외 명
- 보육원, 노인요양시설, 부녀보호시설 등 사회복지시설에 들어가 있는 가족 명

(가구원 수) 이 가구에서 살고 있는 사람은 모두 몇 명입니까? 명

가구일련번호	행정구역(읍면동부군)	행정구역(읍면동부군)	조사구특성	가구원번호	성별	만나이	가구주와
1502000001	35	30	2	1	1	2	85
1502000001	35	30	2	1	2	1	63
1502000002	34	370	2	1	1	1	78
1502000003	31	60	3	1	1	1	46
1502000003	31	60	3	1	2	2	47
1502000004	35	40	2	1	1	1	73
1502000005	36	10	3 A	1	1	1	32
1502000005	36	10	3 A	2	2	2	33
1502000005	36	10	3 A	3	3	2	4
1502000005	36	10	3 A	4	4	1	2
1502000006	31	280	3	1	1	2	30
1502000007	38	30	1	1	1	2	80
1502000008	36	380	2	1	1	1	71

6.1 인구센서스 통계분석 하기

- ✓ 서울과 경기도 가장 많은 비율을 가짐을 알 수 있음

```
library(gmodels)
census <- read.csv("~/Downloads/census2015.csv", skip = 1, header = F)
census$V1 <- as.factor(census$V1)
census[is.na(census)] <- 0

ctab<-function(v1, v2) {
  CrossTable(x = v1, y = v2, prop.t=FALSE, expected=TRUE, chisq =TRUE)}

ctab(census$V2, census$V57)
```

census.R

6.1 인구센서스 통계분석 하기

✓ 하둡을 이용한 프로그램은 아래와 같다.

```
keeps <- c("V2","V57")
census1 <- census[keeps]
census_hdfs <- to.dfs(census1)

mapper <- function(k, v) {
  keys <- v$V2
  values <- v$V57
  keyval(keys, values)
}

reducer <- function(k, v){
  bycle_Y <- sum(v)
  bycle_N <- length(v) - bycle_Y
  keyval(k, c(bycle_Y, bycle_N))
}

output <- mapreduce(input = census_hdfs, map = mapper)
output <- mapreduce(input = census_hdfs, map = mapper, reduce = reducer)
result <- from.dfs(output)
result
```

- ✓ 2013년 소상공인 조사 자료를 <https://mdis.kostat.go.kr/index.do> 에서 받는다.
- ✓ 이 데이터로부터 소상공인 매출예측과 폐업여부를 예측하는 모델을 만들어 보자.

A	B	C	D	E	F	G	H
주소_시도	업종	성별	연령	사업체_형태	창업년도	근로자수_상	근로자수_하
강원	1	1	2	1	1	0	0
강원	1	1	2	1	1	1	0
강원	1	1	2	1	2	0	0
강원	1	1	2	1	2	1	1
강원	1	1	3	1	1	0	0
강원	1	1	3	1	1	1	0
강원	1	1	3	1	2	0	0
강원	1	1	3	1	2	0	1
강원	1	1	3	1	2	0	1
강원	1	1	3	1	2	1	0
강원	1	1	3	1	2	1	0
강원	1	1	3	1	3	0	0

6.2 소상공인 실태 조사 분석 하기

- ✓ V58은 매출액 구간이고, future는 미래 폐업 여부(0: 유지, 1:폐업 혹은 전업) 임
- ✓ V7은 가족구성원 수, V8은 가족 외 구성원 수이고, emp는 종업원 중 외부인 비율임
- ✓ V10은 보증금, V11은 월세임
- ✓ 회귀모형으로 매출액을 예측해 보자

```
soho <- read.csv("~/Downloads/examples/SOHO2013.csv", skip = 1, header = F)
keeps <- c("V58", "V34", "V7", "V8", "V10", "V11")
soho <- soho[keeps]
```

```
soho$emp <- soho$V8 / (soho$V7 + soho$V8) # 가족 외 구성원 비율
soho$future <- ifelse(soho$V34 == 1, 0, 1) # 유지=0, 전업 혹은 폐업=1
soho1 <- soho[complete.cases(soho), ]
```

```
lmodel <- lm(V58 ~ V7 + emp + V10 + V11, data=soho1)
summary(lmodel)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	1.088e+00	6.760e-02	16.100	<2e-16	***
V7	5.632e-01	4.807e-02	11.718	<2e-16	***
emp	1.070e+00	7.016e-02	15.246	<2e-16	***
V10	2.454e-05	1.106e-05	2.218	0.0266	*
V11	3.975e-03	2.496e-04	15.926	<2e-16	***

6.2 소상공인 실태 조사 분석 하기

- ✓ SOHO2013.csv 파일이 빅데이터라고 가정하고 이 데이터 가공을 하둡으로 ...
- ✓ $V34 > 1$ 이상인 경우가 상대적으로 작으므로 $V34 == 1$, $V34 > 1$ 을 5:5 맞춘 표본을 구해 데이터 크기를 줄여 놓고 로지스틱 회귀를 수행한다.

```
soho <- read.csv("~/Downloads/examples/SOHO2013.csv", skip = 1, header = F)
soho_hdfs <- to.dfs(soho)
```

```
mapper <- function(., v){
  key <- ifelse(v$V34 == 1, 0, 1)
  v$future <- key
  keeps <- c("V58", "V34", "V7", "V8", "V10", "V11", "future")
  value <- v[keeps]
  return(keyval(key, value))
}
```

```
reducer <- function(k, v){
  value <- v[1:886,]
  return(keyval(k, value))
}
```

SOHO2013.R

6.2 소상공인 실태 조사 분석 하기

```
mapRes <- mapreduce(input = soho_hdfs, map = mapper)
tmp <- from.dfs(mapRes)
output <- mapreduce(input = mapRes, reduce = reducer)
result <- from.dfs(output)

table(result$key)
table(result$val$V34)

sample <- result$val
sample$emp <- sample$V8 / (sample$V7 + sample$V8)

gmodel <- glm(future ~ V58 + V7 + emp + V10 + V11, data=sample, family = "binomial")
summary(gmodel)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	1.359e+00	3.255e-01	4.175	2.98e-05	***
V58	-4.510e-01	8.817e-02	-5.115	3.14e-07	***
V7	-6.899e-01	2.428e-01	-2.841	0.00450	**
emp	-9.578e-01	3.336e-01	-2.871	0.00409	**
V10	3.507e-04	6.505e-05	5.392	6.98e-08	***
V11	-3.361e-03	1.405e-03	-2.393	0.01673	*

6.2 소상공인 실태 조사 분석 하기

```
mapRes <- mapreduce(input = soho_hdfs, map = mapper)
tmp <- from.dfs(mapRes)
output <- mapreduce(input = mapRes, reduce = reducer)
result <- from.dfs(output)
```

```
table(result$key)
table(result$val$V34)
```

```
sample <- result$val
sample <- sample[complete.cases(sample),]
sample$emp <- sample$V8 / (sample$V7 + sample$V8)
```

```
gmodel <- glm(future ~ V58 + V7 + emp + V10 + V11, data=sample, family = "binomial")
summary(gmodel)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.359e+00	3.255e-01	4.175	2.98e-05 ***
V58	-4.510e-01	8.817e-02	-5.115	3.14e-07 ***
V7	-6.899e-01	2.428e-01	-2.841	0.00450 **
emp	-9.578e-01	3.336e-01	-2.871	0.00409 **
V10	3.507e-04	6.505e-05	5.392	6.98e-08 ***
V11	-3.361e-03	1.405e-03	-2.393	0.01673 *

```
> xtabs(~ sample$future + sample$pred)
      sample$pred
sample$future    0    1
              0 218 104
              1  97 203
> |
```



끝?

끝? 이 아닌 이제 시작이다.

