

데이터 분석 콘텐츠 활용 매뉴얼

1 농산물



미래창조과학부



한국정보화진흥원



KBIG
빅데이터
전략센터

CONTENTS

Beginning Level 초급과정

I 개요

개요	9
----	---

II 수집

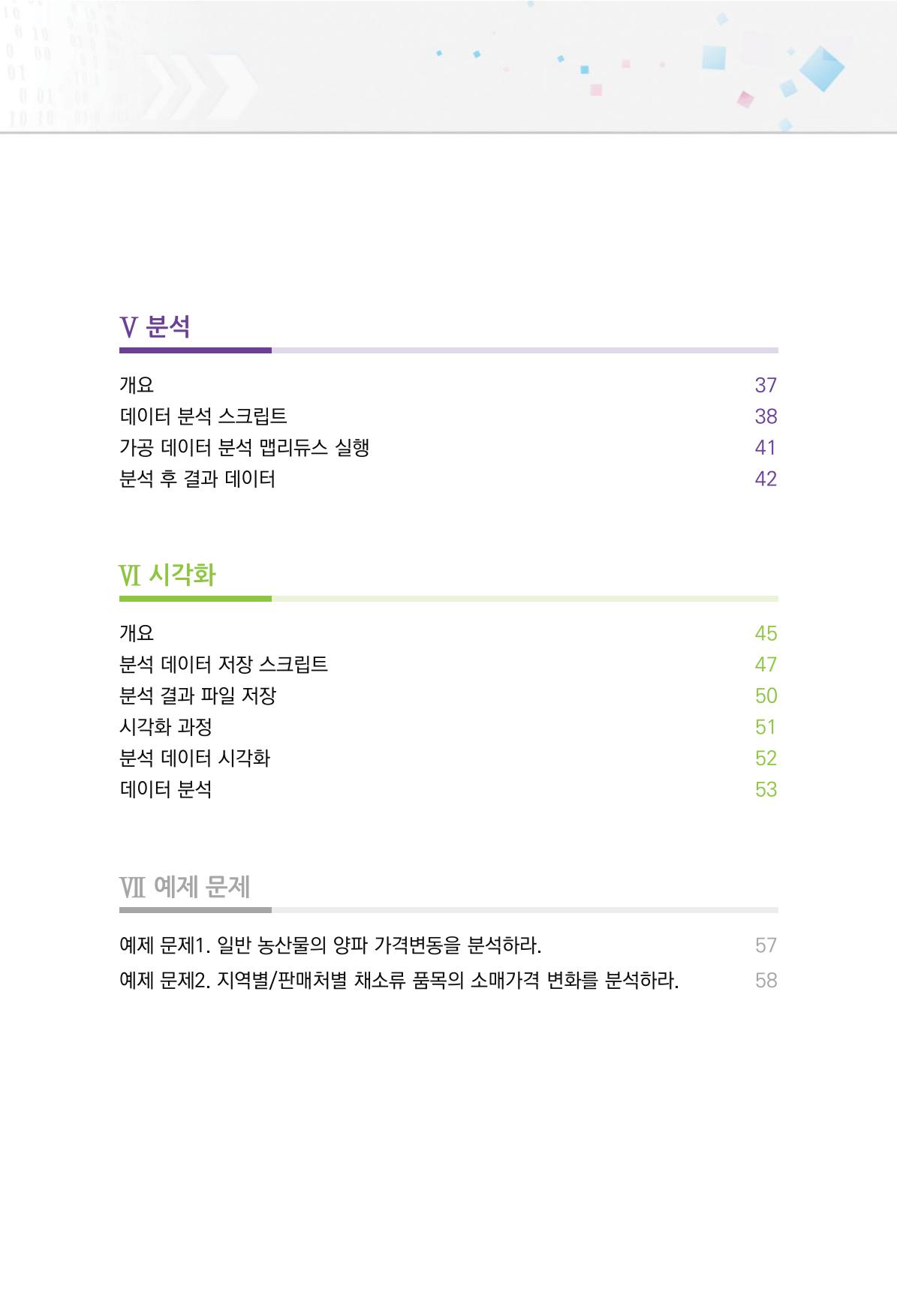
개요	13
교육용 데이터 샘플	14
데이터 수집	15
데이터 작업 영역 이동 스크립트	18

III 가공

개요	23
데이터 가공 스크립트	24

IV 저장

개요	29
가공 데이터 저장	30
몽고DB 저장 데이터 조회	32



V 분석

개요	37
데이터 분석 스크립트	38
가공 데이터 분석 맵리듀스 실행	41
분석 후 결과 데이터	42

VI 시각화

개요	45
분석 데이터 저장 스크립트	47
분석 결과 파일 저장	50
시각화 과정	51
분석 데이터 시각화	52
데이터 분석	53

VII 예제 문제

예제 문제1. 일반 농산물의 양파 가격변동을 분석하라.	57
예제 문제2. 지역별/판매처별 채소류 품목의 소매가격 변화를 분석하라.	58

CONTENTS

Intermediate Level 

I 개요

개요	63
----	----

II 수집

개요	67
교육용 데이터 샘플	68
데이터 수집	69
데이터 작업 영역 이동 스크립트	72

III 가공

개요	77
데이터 가공 스크립트	79

IV 저장

개요	85
가공 데이터 하둡 파일시스템 업로드	86
가공 데이터 하둡 파일시스템 저장	87
하둡 파일시스템 파일 조회	88
하둡 명령어로 파일 조회	89

V 분석

개요	93
데이터 분석 스크립트	95
데이터 분석 맵리듀스 실행	97
분석 데이터 파일 조회	98
분석 후 결과 데이터	99

VI 시각화

개요	103
분석 데이터 저장 방법 1	104
분석 데이터 저장 방법 2	106
시각화 과정	110
분석 데이터 시각화	111
데이터 분석	112

VII 예제 문제

예제 문제1. 친환경농산물의 채소류의 가격 변화와 축산물의 가격 변화를 대도시를 중심으로 비교하라.	115
예제 문제2. 2013년도 일반 농산물의 양파 가격을 월별 비교하여 가격이 폭락 했을 시점의 사회적 이슈를 뉴스 데이터와 매쉬업하여 분석하라.	116



농산물 

Beginning Level

초급과정







I 개요

개요

9

8

I

개요

> 개요

한국농수산식품유통공사에서 제공받은 2011년~2013년 농축산물 데이터를 바탕으로 품목별, 부류별, 지역별 2013년도 농산물 가격 변화를 시계열 분석을 통하여 주이 분석 및 패턴 분석을 하는 방법을 학습한다. 이러한 방법으로 품목별, 부류별, 지역별 계절에 따라 농산물의 가격 변화가 어떻게 영향이 있는지를 알아볼 수 있고, 일반 농산물 채소류와 축산물의 가격을 비교하여 축산물의 가격 변동에 따른 채소류의 가격 변화의 패턴을 알 수 있다. 또한 농산물의 향후 가격 예측과 사회적 이슈나 기상의 요건에 의해서 가격 변동을 예측하는데 활용할 수 있다.

> 활용 데이터

- **product.csv** : 2011~2013년 일반 농축산물 데이터
- **code.csv** : 코드 정보

> 선행학습

- **오플오피스** – 피벗테이블 기능, 차트 사용 방법
- **자바스크립트** – 객체(내장객체, 브라우저객체), 속성, 변수, 연산자(연산자 우선순위), 제어문, 함수(내장함수, 함수정의) 사용 방법
- **몽고 DB** – csv 파일 Import, 맵리듀스 실행 방법
- **D3 차트** – D3 라이브러리 사용법, 차트 설정 방법

▶ 요구사항

- 2011년~2013년 일반 농축산물의 소매가격 정보를 기반으로 농축산물 데이터 중 축산물의 돼지고기 소매가격과 일반 농산물의 상추 소매가격 정보를 가지고 시계열 분석에 따른 가격 변화를 품목별로 시각화하여 변화 추이를 분석하라.

▶ 분석 절차

- 수집된 2011년~2013년 일반 농축산물의 소매가격 정보 데이터를 로드한다.
- 제공된 농산물가격 정보(product.csv)에서 2013년도 전국 평균 축산물의 돼지고기 가격과 채소류의 상추 가격을 시계열 분석에 용이한 데이터 형태로 추출하여 저장한다.
- 2013년 일자별 돼지고기 가격과 상추 가격의 일 평균 가격을 구해서 저장을 한다.
- 2013년 돼지고기 가격과 상추 가격의 전국 가격 변화에 대한 패턴분석을 하기 위하여 하둡 맵리듀스 분석을 실행하고, 그 결과 데이터를 활용하여 시계열 분석의 패턴분석을 할 수 있다.
- 분석된 결과 데이터를 엑셀 형식(csv)이나 D3챠트 형식(json)으로 파일로 저장한다.
- 저장된 파일을 불러와서 엑셀이나 D3챠트로 일 평균 가격의 변화를 시각화해 본다.
- 일반 농산물 채소류와 축산물의 평균 가격 변화를 비교하여 축산물의 가격 변동에 따른 채소류의 가격 변화의 연관성 분석과 패턴 분석이 가능하다.



II 수집

개요	13
교육용 데이터 샘플	14
데이터 수집	15
데이터 작업 영역 이동 스크립트	18



수집

▶ 개요

농축산물 데이터는 한국농수산식품유통공사에서 제공받은 2011년 ~2013년 농축산물 데이터를 수집하여 분석 목적을 달성할 수 있는 한도 내에서 품목, 지역 및 마트를 비식별화 처리를 통해 분석에 용이하게 편집하여 제공한다. 농산물 소매가격정보는 일반 농산물, 친환경 농산물 소매가격으로 일자, 부류, 품목, 지역, 마트 별 구분으로 구성이 되어 있다.

- 부류는 식량작물, 채소작물, 특용작물, 과일류, 축산물 항목을 의미
- 품목은 쌀, 배추, 상추, 호박 등 작물을 의미

일반 농산물에서 축산물 가격을 도매가격을 제공하고 있으며, 친환경 농산물에서는 축산물 가격을 제공하지 않는다. 농축산물 도소매가격 정보는 한국농수산식품유통공사에서 제공해 주었다.

▶ 수집 방법

- **데이터 제공** : 농축산물 데이터는 한국농수산식품유통공사에서 제공해 준 데이터를 OpenAPI, 자료수집기(Crawler)로 데이터를 수집하였고, 실습용 자료는 빅데이터 분석 활용센터에 접속하여 농축산물 데이터 셋을 다운로드 할 수 있도록 원시데이터를 제공하고 있다.



용어정리

- **비식별화** : 데이터 값 삭제, 가명처리, 총계처리, 범주화, 데이터 마스킹 등을 통해 개인정보의 일부 또는 전부를 삭제하거나 대체함으로써 다른 정보와 쉽게 결합하여도 특정 개인을 식별할 수 없도록 하는 조치를 말한다.

◦ *출처: 방송통신위원회, “빅데이터 개인정보보호 가이드라인”, 작성일 2014.12.23

▶ 교육용 데이터 샘플

▶ 일반 농산물 소매가격 데이터(product.csv)

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-02	100	111	1101	110212	45,000
2013-01-02	100	111	3911	390401	45,800
2013-01-02	100	111	1101	110401	46,800
2013-01-02	100	111	1101	110402	45,800
2013-01-02	100	111	1101	110403	45,800

▶ 친환경 일반 농산물 소매가격 데이터(green_product.csv)

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-03	100	111	1101	110403	89,600
2013-01-03	100	111	1101	110406	87,000
2013-01-03	100	111	2100	210401	85,200
2013-01-03	100	111	2100	210402	90,860
2013-01-03	100	111	2200	220401	83,210

▶ 코드 설명 데이터(code.csv)

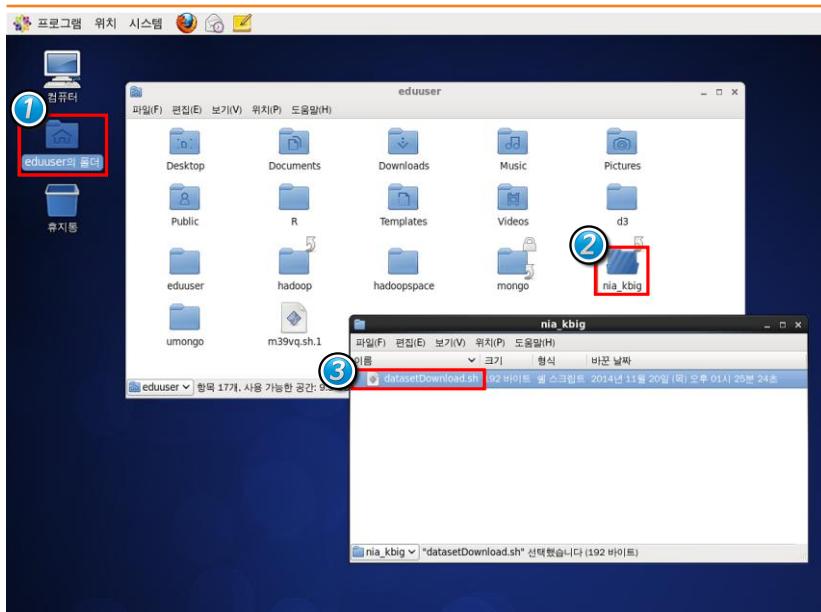
코드구분	코드	코드설명
부류코드	100	식량작물
부류코드	200	채소류
부류코드	300	특용작물
부류코드	400	과일류
부류코드	500	축산물
품목코드	111	쌀
품목코드	211	배추
품목코드	214	상추
품목코드	224	호박
품목코드	245	양파
품목코드	514	돼지고기

II. 수집

▶ 데이터 수집(datasetDownload.sh)

- 데이터 저장소에서 서버 로컬로 일반 농산물 데이터 셋을 복사해 온다.
 - **product.csv** : 소매가격 데이터
 - **code.csv** : 코드 설명 데이터

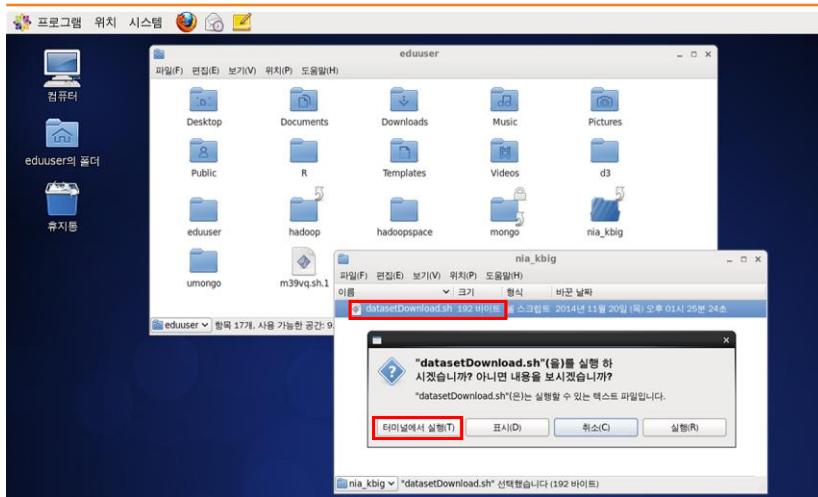
▶ 실습코드 디렉토리로 이동



- ① 로그인 후 바탕화면에서 eduuser 폴더를 오픈한다.
- ② nia_kbig 폴더를 오픈한다.
- ③ datasetDownload.sh를 더블클릭하여 실행한다.

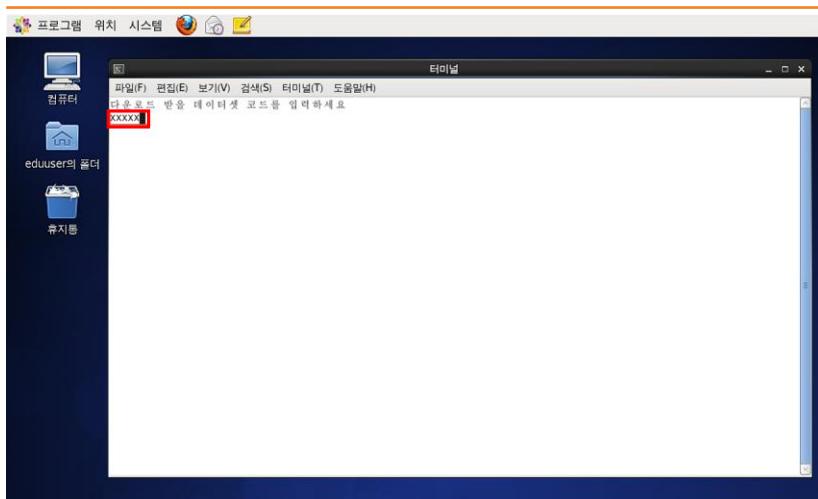
▶ 레파지토리에서 데이터 수집

datasetDownload.sh (원시데이터로 컬서버로 복사)



- ▶ '터미널에서 실행' 버튼을 클릭한다.

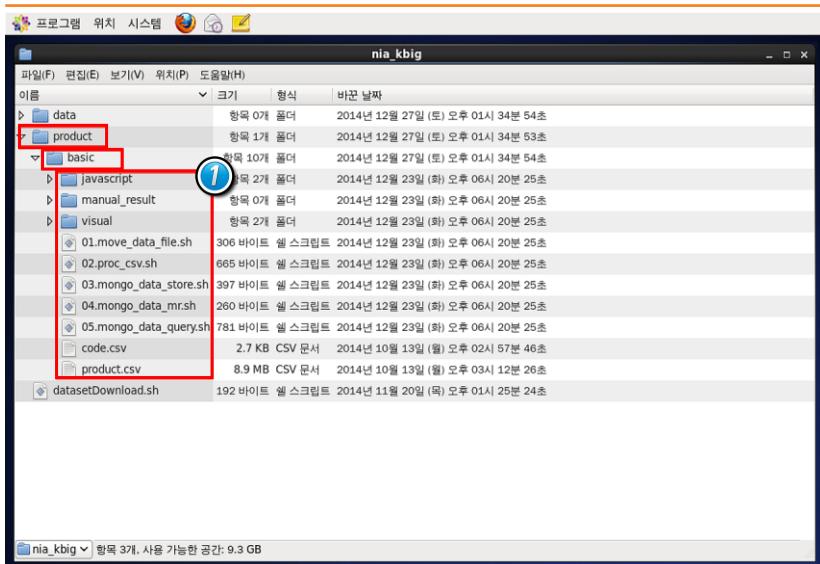
▶ 데이터셋 코드 입력



- ▶ 다운로드 받은 데이터셋 코드를 입력 후 엔터

II. 수집

▶ 데이터셋과 실습용 쉘스크립트



- 실습용 데이터셋과 실습용 스크립트를 확인한다.

▶ ① 데이터 및 스크립트

▪ 01.move_data_file.sh :

작업영역 Data 폴더로 자료 이동하는 스크립트

▪ 02.proc_csv.sh :

원시데이터에서 분석할 대상을 추출하여 저장하는 스크립트

▪ 03.mongo_data_store.sh :

가공데이터를 MongoDB에 저장하는 스크립트

▪ 04.mongo_data_mr.sh :

가공데이터 분석 맵리듀스 실행 스크립트

▪ 05.mongo_data_query.sh :

분석데이터를 저장하는 실행 스크립트

▪ datasetDownload.sh :

레파지토리에서 분석데이터와 실습용 스크립트를 다운로드하는 스크립트

▪ code.csv : 카테고리 분류 코드 데이터

▪ product.csv : 농산물 소매 가격 데이터

> 데이터 작업 영역 이동 스크립트(01.move_data_file.sh)

> 데이터 작업 공간으로 이동

- 로컬로 수집해온 데이터를 작업 영역 Data 폴더로 자료를 이동하는 스크립트

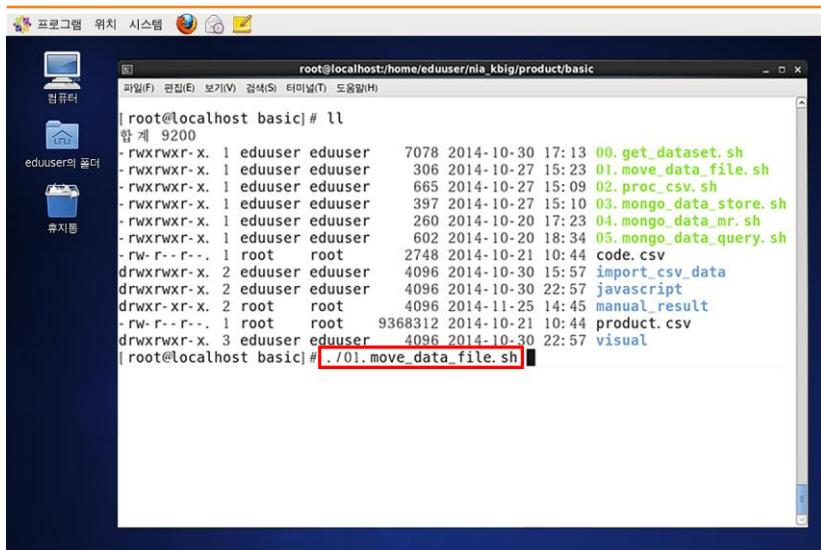
01.move_data_file.sh (작업영역 폴더로 원시데이터 이동)

```

01. #!/bin/bash
02. # 복사 대상 파일 정의
03. #일반농산물 파일
04. TARGET_PRODUCT_PRICE=/home/eduuser/nia_kbig/product/basic/product
05. ↘ .csv
06. TARGET_CODE=/home/eduuser/nia_kbig/product/basic/code.csv
07. # 작업영역 디렉토리 정의
08. LOCAL_DIR=/home/eduuser/nia_kbig/data/
09. # 데이터저장소에서 받은 자료를 작업영역인 /home/eduuser/nia_kbig/data/
10. ↗ 폴더로 자료 이동
11. mv $TARGET_PRODUCT_PRICE $LOCAL_DIR
12. mv $TARGET_GREENPRODUCT_PRICE $LOCAL_DIR
13. mv $TARGET_CODE $LOCAL_DIR
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
146.
147.
148.
149.
150.
151.
152.
153.
154.
155.
156.
157.
158.
159.
160.
161.
162.
163.
164.
165.
166.
167.
168.
169.
170.
171.
172.
173.
174.
175.
176.
177.
178.
179.
180.
181.
182.
183.
184.
185.
186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.
237.
238.
239.
240.
241.
242.
243.
244.
245.
246.
247.
248.
249.
250.
251.
252.
253.
254.
255.
256.
257.
258.
259.
259.
260.
261.
262.
263.
264.
265.
266.
267.
268.
269.
270.
271.
272.
273.
274.
275.
276.
277.
278.
279.
279.
280.
281.
282.
283.
284.
285.
286.
287.
288.
289.
289.
290.
291.
292.
293.
294.
295.
296.
297.
298.
299.
299.
300.
301.
302.
303.
304.
305.
306.
307.
308.
309.
309.
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
339.
340.
341.
342.
343.
344.
345.
346.
347.
348.
349.
349.
350.
351.
352.
353.
354.
355.
356.
357.
358.
359.
359.
360.
361.
362.
363.
364.
365.
366.
367.
368.
369.
369.
370.
371.
372.
373.
374.
375.
376.
377.
378.
379.
379.
380.
381.
382.
383.
384.
385.
386.
387.
388.
389.
389.
390.
391.
392.
393.
394.
395.
396.
397.
398.
399.
399.
400.
401.
402.
403.
404.
405.
406.
407.
408.
409.
409.
410.
411.
412.
413.
414.
415.
416.
417.
418.
419.
419.
420.
421.
422.
423.
424.
425.
426.
427.
428.
429.
429.
430.
431.
432.
433.
434.
435.
436.
437.
438.
439.
439.
440.
441.
442.
443.
444.
445.
446.
447.
448.
449.
449.
450.
451.
452.
453.
454.
455.
456.
457.
458.
459.
459.
460.
461.
462.
463.
464.
465.
466.
467.
468.
469.
469.
470.
471.
472.
473.
474.
475.
476.
477.
478.
479.
479.
480.
481.
482.
483.
484.
485.
486.
487.
488.
489.
489.
490.
491.
492.
493.
494.
495.
496.
497.
498.
499.
499.
500.
501.
502.
503.
504.
505.
506.
507.
508.
509.
509.
510.
511.
512.
513.
514.
515.
516.
517.
517.
518.
519.
519.
520.
521.
522.
523.
524.
525.
526.
527.
528.
529.
529.
530.
531.
532.
533.
534.
535.
536.
537.
538.
539.
539.
540.
541.
542.
543.
544.
545.
546.
547.
548.
549.
549.
550.
551.
552.
553.
554.
555.
556.
557.
558.
559.
559.
560.
561.
562.
563.
564.
565.
566.
567.
568.
569.
569.
570.
571.
572.
573.
574.
575.
576.
577.
578.
579.
579.
580.
581.
582.
583.
584.
585.
586.
587.
588.
589.
589.
590.
591.
592.
593.
594.
595.
596.
597.
598.
599.
599.
600.
601.
602.
603.
604.
605.
606.
607.
608.
609.
609.
610.
611.
612.
613.
614.
615.
616.
617.
618.
619.
619.
620.
621.
622.
623.
624.
625.
626.
627.
628.
629.
629.
630.
631.
632.
633.
634.
635.
636.
637.
638.
639.
639.
640.
641.
642.
643.
644.
645.
646.
647.
648.
649.
649.
650.
651.
652.
653.
654.
655.
656.
657.
658.
659.
659.
660.
661.
662.
663.
664.
665.
666.
667.
668.
669.
669.
670.
671.
672.
673.
674.
675.
676.
677.
678.
679.
679.
680.
681.
682.
683.
684.
685.
686.
687.
688.
689.
689.
690.
691.
692.
693.
694.
695.
696.
697.
698.
699.
699.
700.
701.
702.
703.
704.
705.
706.
707.
708.
709.
709.
710.
711.
712.
713.
714.
715.
716.
717.
717.
718.
719.
719.
720.
721.
722.
723.
724.
725.
726.
727.
728.
729.
729.
730.
731.
732.
733.
734.
735.
736.
737.
738.
739.
739.
740.
741.
742.
743.
744.
745.
746.
747.
748.
749.
749.
750.
751.
752.
753.
754.
755.
756.
757.
758.
759.
759.
760.
761.
762.
763.
764.
765.
766.
767.
768.
769.
769.
770.
771.
772.
773.
774.
775.
776.
777.
778.
779.
779.
780.
781.
782.
783.
784.
785.
786.
787.
788.
789.
789.
790.
791.
792.
793.
794.
795.
796.
797.
798.
799.
799.
800.
801.
802.
803.
804.
805.
806.
807.
808.
809.
809.
810.
811.
812.
813.
814.
815.
816.
817.
818.
819.
819.
820.
821.
822.
823.
824.
825.
826.
827.
828.
829.
829.
830.
831.
832.
833.
834.
835.
836.
837.
838.
839.
839.
840.
841.
842.
843.
844.
845.
846.
847.
848.
849.
849.
850.
851.
852.
853.
854.
855.
856.
857.
858.
859.
859.
860.
861.
862.
863.
864.
865.
866.
867.
868.
869.
869.
870.
871.
872.
873.
874.
875.
876.
877.
878.
879.
879.
880.
881.
882.
883.
884.
885.
886.
887.
888.
889.
889.
890.
891.
892.
893.
894.
895.
896.
897.
898.
898.
899.
900.
901.
902.
903.
904.
905.
906.
907.
908.
909.
909.
910.
911.
912.
913.
914.
915.
916.
917.
918.
919.
919.
920.
921.
922.
923.
924.
925.
926.
927.
928.
929.
929.
930.
931.
932.
933.
934.
935.
936.
937.
938.
939.
939.
940.
941.
942.
943.
944.
945.
946.
947.
948.
949.
949.
950.
951.
952.
953.
954.
955.
956.
957.
958.
959.
959.
960.
961.
962.
963.
964.
965.
966.
967.
968.
969.
969.
970.
971.
972.
973.
974.
975.
976.
977.
978.
979.
979.
980.
981.
982.
983.
984.
985.
986.
987.
988.
989.
989.
990.
991.
992.
993.
994.
995.
996.
997.
998.
999.
999.
1000.
1001.
1002.
1003.
1004.
1005.
1006.
1007.
1008.
1009.
1009.
1010.
1011.
1012.
1013.
1014.
1015.
1016.
1017.
1017.
1018.
1019.
1019.
1020.
1021.
1022.
1023.
1024.
1025.
1026.
1027.
1028.
1029.
1029.
1030.
1031.
1032.
1033.
1034.
1035.
1036.
1037.
1038.
1039.
1039.
1040.
1041.
1042.
1043.
1044.
1045.
1046.
1047.
1048.
1049.
1049.
1050.
1051.
1052.
1053.
1054.
1055.
1056.
1057.
1058.
1059.
1059.
1060.
1061.
1062.
1063.
1064.
1065.
1066.
1067.
1068.
1069.
1069.
1070.
1071.
1072.
1073.
1074.
1075.
1076.
1077.
1078.
1079.
1079.
1080.
1081.
1082.
1083.
1084.
1085.
1086.
1087.
1088.
1089.
1089.
1090.
1091.
1092.
1093.
1094.
1095.
1096.
1097.
1097.
1098.
1099.
1099.
1100.
1101.
1102.
1103.
1104.
1105.
1106.
1107.
1108.
1109.
1109.
1110.
1111.
1112.
1113.
1114.
1115.
1116.
1117.
1118.
1119.
1119.
1120.
1121.
1122.
1123.
1124.
1125.
1126.
1127.
1128.
1129.
1129.
1130.
1131.
1132.
1133.
1134.
1135.
1136.
1137.
1138.
1139.
1139.
1140.
1141.
1142.
1143.
1144.
1145.
1146.
1147.
1148.
1149.
1149.
1150.
1151.
1152.
1153.
1154.
1155.
1156.
1157.
1158.
1159.
1159.
1160.
1161.
1162.
1163.
1164.
1165.
1166.
1167.
1168.
1169.
1169.
1170.
1171.
1172.
1173.
1174.
1175.
1176.
1177.
1178.
1178.
1179.
1180.
1181.
1182.
1183.
1184.
1185.
1186.
1187.
1187.
1188.
1189.
1189.
1190.
1191.
1192.
1193.
1194.
1195.
1196.
1196.
1197.
1198.
1199.
1199.
1200.
1201.
1202.
1203.
1204.
1205.
1206.
1207.
1208.
1209.
1209.
1210.
1211.
1212.
1213.
1214.
1215.
1216.
1217.
1218.
1219.
1219.
1220.
1221.
1222.
1223.
1224.
1225.
1226.
1227.
1228.
1229.
1229.
1230.
1231.
1232.
1233.
1234.
1235.
1236.
1237.
1238.
1239.
1239.
1240.
1241.
1242.
1243.
1244.
1245.
1246.
1247.
1248.
1249.
1249.
1250.
1251.
1252.
1253.
1254.
1255.
1256.
1257.
1258.
1259.
1259.
1260.
1261.
1262.
1263.
1264.
1265.
1266.
1267.
1268.
1269.
1269.
1270.
1271.
1272.
1273.
1274.
1275.
1276.
1277.
1278.
1278.
1279.
1280.
1281.
1282.
1283.
1284.
1285.
1286.
1287.
1287.
1288.
1289.
1289.
1290.
1291.
1292.
1293.
1294.
1295.
1296.
1297.
1297.
1298.
1299.
1299.
1300.
1301.
1302.
1303.
1304.
1305.
1306.
1307.
1308.
1309.
1309.
1310.
1311.
1312.
1313.
1314.
1315.
1316.
1317.
1318.
1319.
1319.
1320.
1321.
1322.
1323.
1324.
1325.
1326.
1327.
1328.
1329.
1329.
1330.
1331.
1332.
1333.
1334.
1335.
1336.
1337.
1338.
1339.
1339.
1340.
1341.
1342.
1343.
1344.
1345.
1346.
1347.
1348.
1349.
1349.
1350.
1351.
1352.
1353.
1354.
1355.
1356.
1357.
1358.
1359.
1359.
1360.
1361.
1362.
1363.
1364.
1365.
1366.
1367.
1368.
1369.
1369.
1370.
1371.
1372.
1373.
1374.
1375.
1376.
1377.
1378.
1378.
1379.
1380.
1381.
1382.
1383.
1384.
1385.
1386.
1387.
1387.
1388.
1389.
1389.
1390.
1391.
1392.
1393.
1394.
1395.
1396.
1397.
1397.
1398.
1399.
1399.
1400.
1401.
1402.
1403.
1404.
1405.
1406.
1407.
1407.
1408.
1409.
1409.
1410.
1411.
1412.
1413.
1414.
1415.
1416.
1416.
1417.
1418.
1418.
1419.
1420.
1421.
1422.
1423.
1424.
1425.
1426.
1427.
1427.
1428.
1429.
1429.
1430.
1431.
1432.
1433.
1434.
1435.
1436.
1437.
1437.
1438.
1439.
1439.
1440.
1441.
1442.
1443.
1444.
1445.
1446.
1447.
1448.
1448.
1449.
1450.
1451.
1452.
1453.
1454.
1455.
1456.
1457.
1458.
1458.
1459.
1460.
1461.
1462.
1463.
1464.
1465.
1466.
1467.
1468.
1468.
1469.
1470.
1471.
1472.
1473.
1474.
1475.
1476.
1477.
1478.
1478.
1479.
1480.
1481.
1482.
1483.
1484.
1485.
1486.
1487.
1488.
1488.
1489.
1490.
1491.
1492.
1493.
1494.
1495.
1496.
1497.
1497.
1498.
1499.
1499.
1500.
1501.
1502.
1503.
1504.
1505.
1506.
1507.
1508.
1508.
1509.
1510.
1511.
1512.
1513.
1514.
1515.
1516.
1517.
1518.
1518.
1519.
1520.
1521.
1522.
1523.
1524.
1525.
1526.
1527.
1528.
1529.
1529.
1530.
1531.
1532.
1533.
1534.
1535.
1536.
1537.
1538.
1538.
1539.
1540.
1541.
1542.
1543.
1544.
1545.
1546.
1547.
1548.
1548.
1549.
1550.
1551.
1552.
1553.
1554.
1555.
1556.
1557.
1558.
1559.
1559.
1560.
1561.
1562.
1563.
1564.
1565.
1566.
1567.
1568.
1568.
1569.
1570.
1571.
1572.
1573.
1574.
1575.
1576.
1577.
1578.
1578.
1579.
1580.
1581.
1582.
1583.
1584.
1585.
1586.
1587.
1588.
1588.
1589.
1590.
1591.
1592.
1593.
1594.
1595.
1596.
1597.
1598.
1598.
1599.
1599.
1600.
1601.
1602.
1603.
1604.
1605.
1606.
1607.
1608.
1609.
1609.
1610.
1611.
1612.
1613.
1614.
1615.
1616.
1617.
1618.
1619.
1619.
1620.
1621.
1622.
1623.
1624.
1625.
1626.
1627.
1628.
1629.
1629.
1630.
1631.
1632.
1633.
1634.
1635.
1636.
1637.
1638.
1638.
1639.
1640.
1641.
1642.
1643.
1644.
1645.
1646.
1647.
1648.
1648.
1649.
1650.
1651.
1652.
1653.
1654.
1655.
1656.
1657.
1658.
1659.
1659.
1660.
1661.
1662.
1663.
1664.
1665.
1666.
1667.
1668.
1668.
1669.
1670.
1671.
1672.
1673.
1674.
1675.
1676.
1677.
1678.
1678.
1679.
1680.
1681.
1682.
1683.
1684.
1685.
1686.
1687.
1688.
1688.
1689.
1690.
1691.
1692.
1693.
1694.
1695.
1696.
1697.
1698.
1698.
1699.
1699.
1700.
1701.
1702.
1703.
1704.
1705.
1706.
1707.
1708.
1709.
1709.
1710.
1711.
1712.
1713.
1714.
1715.
1716.
1717.
1718.
1719.
1719.
1720.
1721.
1722.
1723.
1724.
1725.
1726.
1727.
1728.
1729.
1729.
1730.
1731.
1732.
1733.
1734.
1735.
1736.
1737.
1738.
1738.
1739.
1740.
1741.
1742.
1743.
1744.
1745.
1746.
1747.
1748.
1748.
1749.
1750.
1751.
1752.
1753.
1754.
1755.
1756.
1757.
1758.
1759.
1759.
1760.
1761.
1762.
1763.
1764.
1765.
1766.
1767.
1768.
1768.
1769.
1770.
1771.
1772.
1773.
1774.
1775.
1776.
1777.
1778.
1779.
1779.
1780.
1781.
1782.
1783.
1784.
1785.
1786.
1787.
1788.
1788.
1789.
1790.
1791.
1792.
1793.
1794.
1795.
1796.
1797.
1798.
1798.
1799.
1799.
1800.
1801.
1802.
1803.
1804.
1805.
1806.
1807.
1808.
1809.
1809.
1810.
1811.
1812.
1813.
1814.
1815.
1816.
1817.
1818.
1818.
1819.
1820.
1821.
1822.
1823.
1824.
1825.
1826.
1827.
1828.
1829.
1829.
1830.
1831.
1832.
1833.
1834.
1835.
1836.
1837.
1838.
1838.
1839.
1840.
1841.
1842.
1843.
1844.
1845.
1846.
1847.
1848.
1848.
1849.
1850.
1851.
1852.
1853.
1854.
1855.
1856.
1857.
1858.
1859.
1859.
1860.
1861.
1862.
1863.
1864.
1865.
1866.
1867.
1868.
1868.
1869.
1870.
1871.
1872.
1873.
1874.
1875.
1876.
1877.
1878.
1878.
1879.
1880.
1881.
1882.
1883.
1884.
1885.
1886.
1887.
1888.
1889.
1889.
1890.
1891.
1892.
1893.
1894.
1895.
1896.
1897.
1898.
1898.
1899.
1899.
1900.
1901.
1902.
1903.
1904.
1905.
1906.
1907.
1908.
1909.
1909.
1910.
1911.
1912.
1913.
1914.
1915.
1916.
1917.
1918.
1919.
1919.
1920.
1921.
1922.
1923.
1924.
1925.
1926.
1927.
1928.
1929.
1929.
1930.
1931.
1932.
1933.
1934.
1935.
1936.
1937.
1938.
1939.
1939.
1940.
1941.
1942.
1943.
1944.
1945.
1946.
1947.
1948.
1949.
1949.
1950.
1951.
1952.
1953.
1954.
1955.
1956.
1957.
1958.
1959.
1959.
1960.
1961.
1962.
1963.
1964.
1965.
1966.
1967.
1968.
1969.
1969.
1970.
1971.
1972.
1973.
1974.
1975.
1976.
1977.
1978.
1979.
1979.
1980.
1981.
1982.
1983.
1984.
1985.
1986.
1987.
1988.
1989.
1989.
1990.
1991.
1992.
1993.
1994.
1995.
1996.
1997.
1998.
1999.
1999.
2000.
2001.
2002.
2003.
2004.
2005.
2006.
2007.
2008.
2009.
2009.
2010.
2011.
2012.
2013.
2014.
2015.
2016.
2017.
2018.
2019.
2019.
2020.
2021.
2022.
2023.
2024.
2025.
2026.
2027.
2028.
2029.
2029.
2030.
2031.
2032.
2033.
2034.
2035.
2036.
2037.
2038.
2039.
2039.
2040.
2041.
2042.
2043.
2044.
2045.
2046.
2047.
2048.
2049.
2049.
2050.
2051.
2052.
2053.
2054.
2055.
2056.
2057.
2058.
2059.
2059.
2060.
2061.
2062.
2063.
2064.
2065.
2066.
2067.
2068.
2069.
2069.
2070.
2071.
2072.
2073.
2074.
2075.
2076.
2077.
2078.
2079.
2079.
2080.
2081.
2082.
2083.
2084.
2085.
2086.
2087.
2088.
2089.
2089.
2090.
2091.
2092.
2093.
2094.
2095.
2096.
2097.
2097.
2098.
2099.
2099.
2100.
2101.
2102.
2103.
2104.
2105.
2106.
2107.
2108.
2109.
2109.
2110.
2111.
2112.
2113.
2114.
2115.
2116.
2117.
2118.
2119.
2119.
2120.
2121.
2122.
2123.
2124.
2125.
2126.
2127.
2128.
2129.
2129.
2130.
2131.
2132.
2133.
2134.
2135.
2136.
2137.
2138.
2139.
2139.
2140.
2141.
2142.
2143.
2144.
2145.
2146.
2147.
2148.
2149.
2149.
2150.
2151.
2152.
2153.
2154.
2155.
2156.
2157.
2158.
2159.
2159.
2160.
2161.
2162.
2163.
2164.
2165.
2166.
2167.
2168.
2169.
2169.
2170.
2171.
2172.
2173.
2174.
2175.
2176.
2177.
2178.
2179.
2179.
2180.
2181.
2182.
2183.
2184.
2185.
2186.
2187.
2188.
2189.
2189.
2190.
2191.
2192.
2193.
2194.
2195.
2196.
2197.
2198.
2198.
2199.
2199.
2200.
2201.
2202.
2203.
2204.
2205.
2206.
2207.
2208.
2209.
2209.
2210.
2211.
2212.
2213.
2214.
2215.
2216.
2217.
2218.
2219.
2219.
2220.
2221.
2222.
2223.
2224.
2225.
2226.
2227.
2228.
2229.
2229.
2230.
2231.
2232.
2233.
2234.
2235.
2236.
2237.
2238.
2239.
2239.
2240.
2241.
2242.
2243.
2244.
2245.
2246.
2247.
2248.
2249.
2249.
2250.
2251.
2252.
2253.
2254.
2255.
2256.
2257.
2258.
2259.
2259.
2260.
2261.
2262.
2263.
2264.
2265.
2266.
2267.
2268.
2269.
2269.
2270.
2271.
2272.
2273.
2274.
2275.
2276.
2277.
2278.
2278.
2279.
2280.
2281.
2282.
2283.
2284.
2285.
2286.
2287.
2288.
2288.
2289.
2289.
2290.
2291.
2292.
2293.
2294.
2295.
2296.
2296.
2297.
2298.
2299.
2299.
2300.
2301.
2302.
2303.
2304.
2305.
2306.
2307.
2308.
2309.
2309.
2310.
2311.
2312.
2313.
2314.
2315.
2316.
2317.
2318.
2319.
2319.
2320.
2321.
2322.
2323.
2324.
2325.
2326.
2327.
2328.
2329.
2329.
2330.
2331.
2332.
2333.
2334.
2335.
2336.
2337.
2338.
2339.
2339.
2340.
2341.
2342.
2343.
2344.
2345.
2346.
2347.
2348.
2349.
2349.
2350.
2351.
2352.
2353.
2354.
2355.
235
```

II. 수집

▶ 수집 데이터 셋 작업 영역 폴더 이동



The screenshot shows a terminal window titled "root@localhost:/home/eduuser/nia_kbig/product/basic". The window displays a file listing with the command "ll". The output shows various files like "00.get_dataset.sh", "01.move_data_file.sh", etc., with their permissions, sizes, and modification dates. At the bottom of the terminal, the command "/01.move_data_file.sh" is highlighted with a red box.

```
root@localhost basic] # ll
합계 9200
-rwxrwxr-x. 1 eduuser eduuser 7078 2014-10-30 17:13 00.get_dataset.sh
-rwxrwxr-x. 1 eduuser eduuser 306 2014-10-27 15:23 01.move_data_file.sh
-rwxrwxr-x. 1 eduuser eduuser 665 2014-10-27 15:09 02.proc_csv.sh
-rwxrwxr-x. 1 eduuser eduuser 397 2014-10-27 15:10 03.mongo_data_store.sh
-rwxrwxr-x. 1 eduuser eduuser 260 2014-10-20 17:23 04.mongo_data_mr.sh
-rwxrwxr-x. 1 eduuser eduuser 602 2014-10-20 18:34 05.mongo_data_query.sh
-rw-r--r--. 1 root  root 2748 2014-10-21 10:44 code.csv
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 22:57 javascript
drwxr-xr-x. 2 root  root 4096 2014-11-25 14:45 manual_result
-rw-r--r--. 1 root  root 9368312 2014-10-21 10:44 product.csv
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[ root@localhost basic] # ./01.move_data_file.sh
```

- 로컬에 원시데이터를 작업영역 폴더로 이동 (/home/eduuser/nia_kbig/data/) 시킨다.
./01.move_data_file.sh 입력 후 엔터

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화







가공

> 개요

작업 영역 폴더에 복사한 일반농산물 소매가격 데이터에서 2013년 축산물 부류의 돼지고기, 채소류의 상추 가격을 전 지역의 데이터를 추출하여 CSV 파일(2013_product.csv) 형식으로 저장 한다. 데이터 가공은 셀 스크립트를 사용하여 전체 데이터에서 필터링 기법을 적용하여 데이터를 가공한다.



> 가공 방법

- 일반 농산물 소매가격 데이터에는 2011~2013년간의 3년치 데이터가 저장 되어 있다.
- 분석 대상은 2013년도의 돼지고기와 상추 가격이므로, 일반 농산물 소매 가격 데이터 (product.csv) 파일에서 2013년도 돼지고기 가격과 상추 가격 데이터만 추출하여 2013_product.csv 파일을 생성한다.

> 데이터셋

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-02	100	111	1101	110212	45,000
2013-01-02	100	111	3911	390401	45,800
2013-01-02	100	111	1101	110401	46,800
2013-01-02	100	111	1101	110402	45,800
2013-01-02	100	111	1101	110403	45,800

> 데이터 가공 스크립트(02.proc_csv.sh)

- 셀 스크립트를 이용하여 2013년도 데이터만을 추출한다. 추출한 데이터는 2013_product.csv로 저장한다.

02.proc_csv.sh (원시데이터에서 분석할 대상을 추출 하여 저장)

```
01.#!/bin/bash
02. # 입력 CSV 파일 지정
03. INPUT_FILE='/home/eduuser/nia_kbig/data/product.csv'
04. # 출력결과 CSV 파일 지정
05. OUTPUT_FILE='/home/eduuser/nia_kbig/data/2013_product.csv'
06. # 년도 설정 2013년도 설정
07. TARGET_YEAR='2013'
08. # HEADER컬럼 출력
09. echo "Date,Category,Item,Area,Mart,Price" > $OUTPUT_FILE
10. # ','를 구분자로 해서 파일을 읽어들인다.
11. IFS=','
12. while read DATE CATEGORY ITEM AREA MART PRICE
13. do
14.     # DATA가 TARGET_YEAR로 시작하는 년도인지 체크한다.
15.     if [[ $DATE == ${TARGET_YEAR}* ]]; then
16.         # 해당년도의 데이터만을 CSV로 출력한다.
17.         echo "$DATE,$CATEGORY,$ITEM,$AREA,$MART,$PRICE" >>
18.             $OUTPUT_FILE
19.     fi
20. done < $INPUT_FILE
```



데이터 가공 스크립트 소스(02.proc_csv.sh)

- 라인 03~05 : 가공 대상인 농산물데이터(product.csv) 지정을 하고, 가공 후 가공데이터를 2013_product.csv 파일로 저장하는 라인이다.
- 라인 07 : 가공 년도 설정을 2013년도로 설정하는 라인이다.
- 라인 09 : 가공데이터 파일을 생성시 상단에 Header 정보를 추가하는 라인이다.
- 라인 11~19 : 원시데이터 파일을 1라인씩 읽어서 2013년도 해당하는 데이터만 파일로 저장하는 라인이다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화

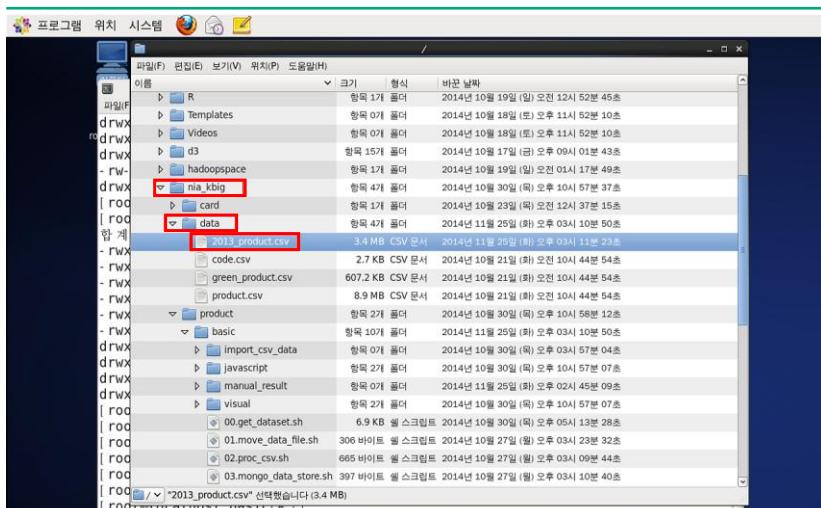
III. 가공

▶ 원시데이터에서 분석 대상 데이터 가공

```
root@localhost:/home/eduuser/nia_kbig/product/basic
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 22:57 javascript
drwxr-xr-x. 2 root root 4096 2014-11-25 14:45 manual_result
-rw-r--r--. 1 root root 9368312 2014-10-21 10:44 product.csv
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[ root@localhost basic]# ./01.move_data_file.sh
[ root@localhost basic]# ll
합계 44
-rwxrwxr-x. 1 eduuser eduuser 7078 2014-10-30 17:13 00.get_dataset.sh
-rwxrwxr-x. 1 eduuser eduuser 306 2014-10-27 15:23 01.move_data_file.sh
-rwxrwxr-x. 1 eduuser eduuser 665 2014-10-27 15:09 02.proc_csv.sh
-rwxrwxr-x. 1 eduuser eduuser 397 2014-10-27 15:10 03.mongo_data_store.sh
-rwxrwxr-x. 1 eduuser eduuser 260 2014-10-20 17:23 04.mongo_data_mr.sh
-rwxrwxr-x. 1 eduuser eduuser 602 2014-10-20 18:34 05.mongo_data_query.sh
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 22:57 javascript
drwxr-xr-x. 2 root root 4096 2014-11-25 14:45 manual_result
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[ root@localhost basic]#
[ root@localhost basic]# ./02.proc_csv.sh
[ root@localhost basic]#
```

- 원시 데이터 셋에서 분석할 데이터를 가공하여 2013_product.csv 파일을 생성한다.
.02.proc_csv.sh 입력 후 엔터

▶ 가공 데이터 작업 영역 폴더에 생성



- /home/eduuser/nia_kbig/data 폴더에 2013_product.csv 파일이 생성된다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화

WIT



IV 저장

개요	29
가공 데이터 저장	30
동고DB 저장 데이터 조회	32

IV

저장

> 개요

시계열 분석의 패턴 분석을 위해서 목표 대상과 분석할 범위가 지정된 가공 데이터(2013_product.csv)를 몽고DB에 products 컬렉션을 만들어 저장을 한다. 몽고DB 컬렉션에 저장된 데이터는 몽고DB에서 자체 제공하는 맵리듀스 분석을 이용할 수 있다. 몽고DB는 메모리 특성을 가지고 있기 때문에 데이터 읽기, 쓰기 연산이 빠른 특징을 가지고 있고 스키마 제약이 없기 때문에 좀 더 유연하게 데이터를 처리할 수 있다.

> 저장 방법

- 2013년도 일반 농산물 소매가격 데이터 파일(2013_product.csv)을 분석하기 위해서 몽고DB에 Import 처리한다.
- 몽고에서 제공하는 CSV Import 툴인 mongoimport를 사용하여 몽고DB에 원시데이터를 Import 한다.
- 몽고DB에 들어가 있는 내용을 파악하기 위해서 /home/eduuser/nia_kbig/tools/umongo/ 폴더에 있는 lauch-umongo.sh 파일을 실행하여 입력된 데이터를 확인한다.

> 가공 데이터 저장(03.mongo_data_store.sh)

> MongoDB에 가공 데이터 저장 스크립트

- MongoDB로 저장할 CSV 파일을 mongoimport 명령어로 저장 처리를 한다.

03.mongo_data_store.sh (가공데이터를 MongoDB에 저장)

```

01.#!/bin/bash
02. # Import 파일 위치 경로
03. LOCAL_TARGET=/home/eduuser/nia_kbig/data/2013_product.csv
04. # mongo DB 접속정보 설정
05. MONGO_HOST=127.0.0.1
06. MONGO_PORT=27017
07. #mongo 데이터베이스명
08. MONGO_DATABASE=bigdata
09. #mongo 컬렉션 명
10. MONGO_COLLECTION=products
11. #mongo Import 파일 형식
12. MONGO_IMPORT_FILE_TYPE=csv
13. # mongo DB로 가공데이터 Import 처리 명령어
14. mongoimport -h $MONGO_HOST --port $MONGO_PORT \
15.           -d $MONGO_DATABASE -c $MONGO_COLLECTION \
16.           --type $MONGO_IMPORT_FILE_TYPE --file $LOCAL_TAR
17.             ↛ GET -headerline

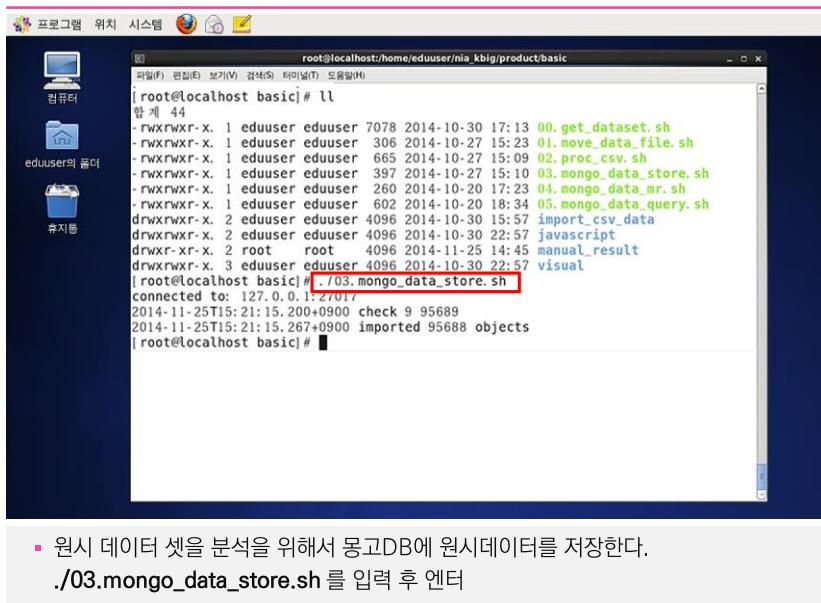
```



- 가공 데이터 저장 스크립트 소스(03.mongo_data_store.sh)
- 라인 03 : MongoDB에 입력할 가공데이터(2013_product.csv)를 지정하는 라인이다.
- 라인 05~06 : 로컬 서버에 있는 MongoDB에 접속을 설정하는 라인이다.
- 라인 08~12 : bigdata 데이터베이스를 정의하고 컬렉션으로 products를 지정한다. MongoDB에 입력되는 데이터 파일 형식이 csv 파일로 지정하는 라인이다.
- 라인 14~16 : mongoimport 명령어에 접속호스트, 포트, 데이터베이스, 컬렉션, 타입을 지정하고 가공된 데이터(2013_product.csv)를 MongoDB에 import 처리하는 라인이다.

IV. 저장

▶ 가공 데이터 MongoDB에 저장

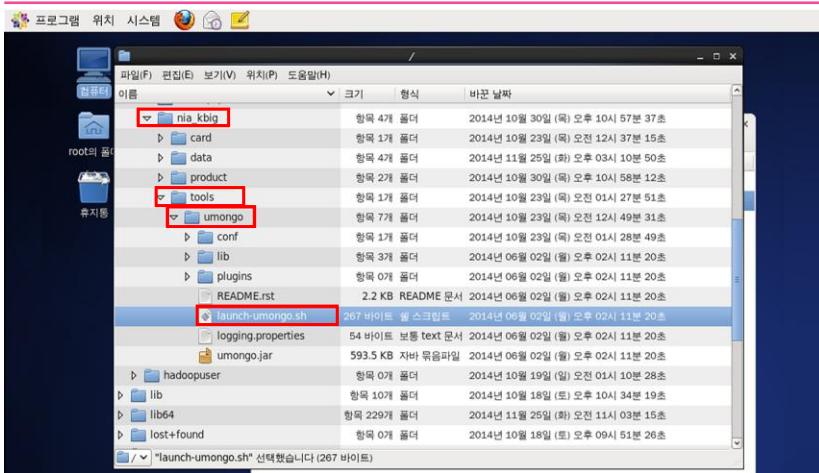


```
root@localhost basic] # ll
total 44
-rwxrwxr-x 1 eduuser eduuser 7078 2014-10-30 17:13 00_get_dataset.sh
-rwxrwxr-x 1 eduuser eduuser 306 2014-10-27 15:23 01_move_data_file.sh
-rwxrwxr-x 1 eduuser eduuser 665 2014-10-27 15:09 02_proc_csv.sh
-rwxrwxr-x 1 eduuser eduuser 397 2014-10-27 15:10 03.mongo_data_store.sh
-rwxrwxr-x 1 eduuser eduuser 260 2014-10-20 17:23 04.mongo_data_mr.sh
-rwxrwxr-x 1 eduuser eduuser 602 2014-10-20 18:34 05.mongo_data_query.sh
drwxrwxr-x 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxrwxr-x 2 eduuser eduuser 4096 2014-10-30 22:57 javascript
drwxr-xr-x 2 root root 4096 2014-11-25 14:45 manual_result
drwxrwxr-x 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[root@localhost basic] # ./03.mongo_data_store.sh
connected to: 127.0.0.1:27017
2014-11-25T15:21:15.200+0900 check 9 95689
2014-11-25T15:21:15.267+0900 imported 95688 objects
[root@localhost basic] #
```

■ 원시 데이터 셋을 분석을 위해서 MongoDB에 원시데이터를 저장한다.
./03.mongo_data_store.sh 를 입력 후 엔터

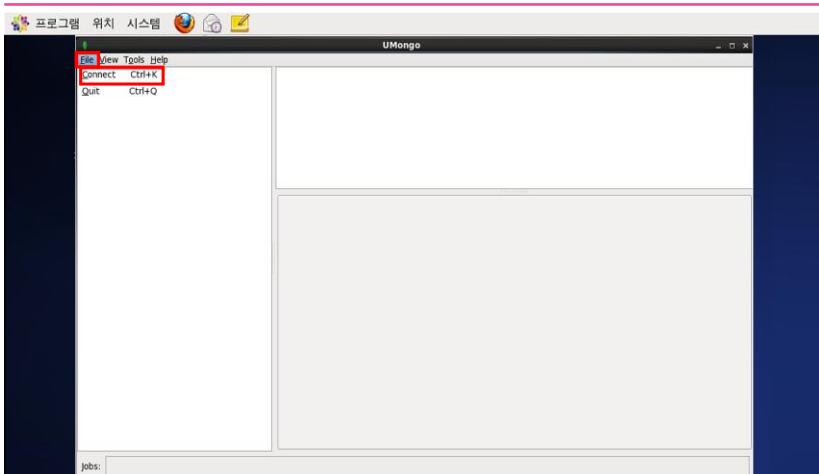
> MongoDB 저장 데이터 조회

> umongo 툴 실행



- /home/eduuser/nia_kbig/tools/umongo/lauch-umongo.sh 더블클릭하여 '터미널에서 실행' 버튼을 클릭하여 MongoDB 관리 툴을 실행한다.

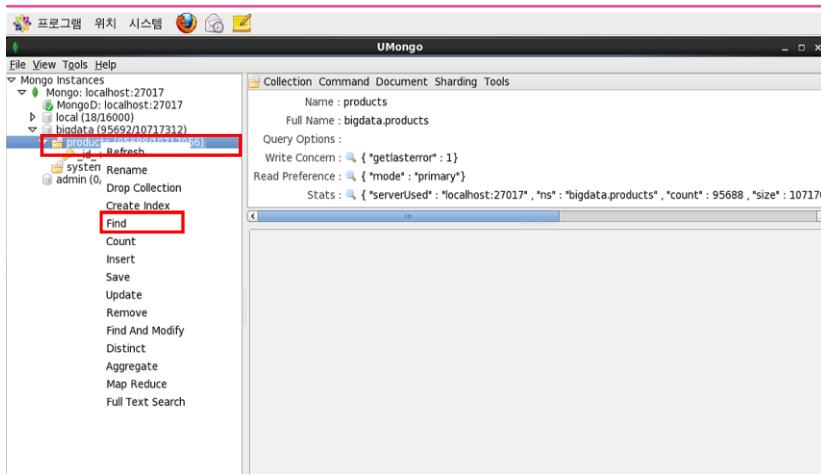
> MongoDB 접속



- File메뉴 > Connect 메뉴 클릭한다.
- Default를 선택하여 OK 버튼을 클릭한다.

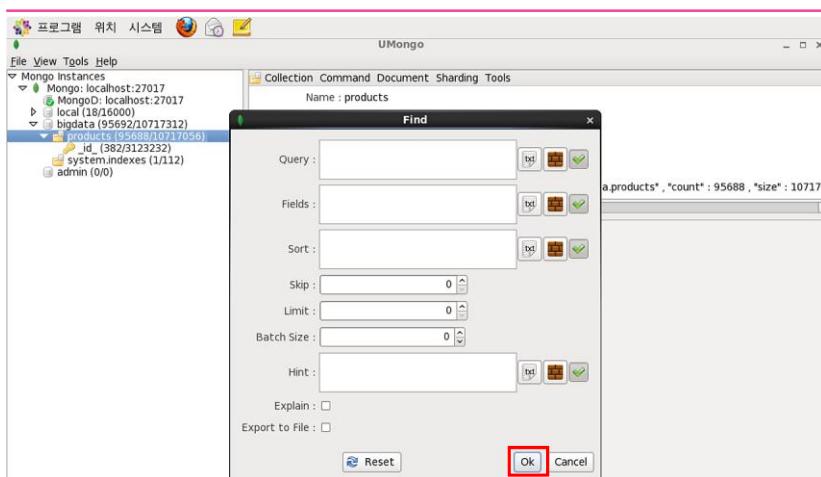
IV. 저장

▶ products 컬렉션 선택



- bigdata DB 하위에 products 컬렉션 선택 후 오른쪽 마우스 클릭 후 Find 메뉴를 클릭한다.

▶ products 컬렉션 Find



- Find 창에서 OK 버튼을 클릭한다.

> products 컬렉션 목록 선택

Collection Command Document Sharding Tools

Name : products
Full Name : bigdata.products

Query Options :

Write Concern : { "w": 1 }
Read Preference : { "mode": "primary" }

Stats : { "serverUsed": "localhost:27017", "ns": "bigdata.products", "count": 95688, "size": 107170 }

bigdata.products / Find [x]

Result View Tools [x]

```
{
  "_id": { "$oid": "54741fd904094fa6aaa20c4f5" },
  "Date": "2013-01-02",
  "Item": 100,
  "Area": 3911,
  "Mart": 390401,
  "Price": 45800
}
```

- bigdata MongoDB에 products 컬렉션에 입력된 데이터 목록이 출력이 되고 목록을 클릭하면 데이터셋 내용을 확인 할 수 있다.

W





V 분석

개요	37
데이터 분석 스크립트	38
가공 데이터 분석 맵리듀스 실행	41
분석 후 결과 데이터	42

V

분석

> 개요

농축산물 데이터의 분석은 MongoDB 컬렉션(products)에 저장되어 있는 가공된 데이터를 가지고 돼지고기 일자별 평균 가격과 상추 일자별 평균 가격을 구하기 위해서 품목별, 일자별 평균 가격을 MongoDB 맵리듀스 분석 스크립트로 작성한다. 돼지고기 가격 변화에 따른 상추 가격 변화를 파악하기 위해서 전 지역의 데이터를 비교 분석 대상으로 하고 계절에 따른 두 품목 간의 연관성을 알아보기 한다. 일자별 돼지고기 가격 변화 추이와 상추의 가격 변화 추이를 비교 분석하기 위해 시계열 분석 방법을 적용한다.



> 분석 방법

- MongoDB의 컬렉션(products)에 JSON 형태로 저장되어 있는 2013년도 일반 농산물 소매가격 데이터를 중심으로 맵리듀스 작업을 통해서 일자별 돼지고기 평균 가격과 일자별 상추 평균 가격의 데이터를 뽑아낸다.

> 가공 데이터 샘플

Date	Category	Item	Area	Mart	Price
2013-01-02	100	111	1101	110212	45,000
2013-01-02	100	111	3911	390401	45,800
2013-01-02	100	111	1101	110401	46,800
2013-01-02	100	111	1101	110402	45,800
2013-01-02	100	111	1101	110403	45,800

> 데이터 분석 스크립트(04.mongo_data_mr.sh)

> 가공 데이터 분석 실행 스크립트

- 몽고는 맵리듀스를 자바스크립트로 처리하여 자바스크립트 파일로 작성한 후 몽고 커맨드로 실행시켜서 결과를 추출한다.

04.mongo_data_mr.sh (가공데이터 분석 맵리듀스 실행)

```

01.#!/bin/bash
02. #Mongo DB 접속
03. MONGO_HOST=127.0.0.1
04. MONGO_PORT=27017
05. MONGO_DATABASE=bigdata
06. # 돼지평균가격과 상주의 평균가격을 구하는 M/R 모듈
07. EXECUTE_JS_MODULE=javascript/products_avg_price.js
08. # Mongo에서 M/R을 실행하는 명령어
09. mongo --host $MONGO_HOST --port $MONGO_PORT $MONGO_DATABASE
   ↳ SE $ EXECUTE_JS_MODULE
10.

```



- 가공데이터 분석 맵리듀스 실행 스크립트 소스(04.mongo_data_mr.sh)
- 라인 03~05 : 로컬 서버에 있는 몽고DB에 접속정보를 설정하는 라인이다.
- 라인 07 : javascript 폴더에 있는 분석스크립트 파일(products_avg_price.js)을 맵리듀스 분석 파일로 지정하는 라인이다.
- 라인 09 : mongo 명령어에 접속호스트, 포트, 데이터베이스를 지정하고 가공된 데이터를 몽고 DB에서 맵리듀스 분석 작업을 수행하는 라인이다.

> 맵리듀스 분석 스크립트(products_avg_price.js)

```

01. /**
02. * [products] 콜렉션을 대상으로 '일자','품명'별로 일자별 평균가격을 구한다.
03. */
04.
05. // MapReduce 대상 콜렉션을 선정한다.
06. var products = db.products;
07. // Map함수를 선언한다.
08. var map = function() {
09. var key = this.Date + ":" + this.Item; // '일자:품명' 을 키로 만든다.
10. var curDoc = new Object;
11. curDoc.date = this.Date;
12. curDoc.item = this.Item;
13. curDoc.totalPrice = this.Price;
14. curDoc.avgPrice = this.Price; // 아이템갯수가 1개인 것은 reduce에 넘어가지 않는다.
    ↪ 따라서 기본으로 평균값을 넣어준다.
15. curDoc.count = 1;
16. emit(key, curDoc);
17. };
18. // Reduce함수를 선언한다.
19. var reduce = function(key, products) {
20. // map에서 생성한 도큐먼트와 동일한 JSON 형태로 만들어주어야 한다.
21. var reduced = {date: "", item: "", totalPrice : 0 ,avgPrice : 0 , count : 0};
22. // 동일한 키를 가지는 아이템들을 일순하면서 가격의 총합을 구한다.
23. products.forEach( function(product) {
24.     reduced.date = product.date;
25.     reduced.item = product.item;
26.     reduced.totalPrice += product.totalPrice;
27.     reduced.count += product.count;
28. });
29. reduced.avgPrice = reduced.totalPrice / reduced.count; // 평균가격을 구한다.
30. return reduced;
31. };
32. // 콜렉션에 M/R 작업을 건다.
33. products.mapReduce(

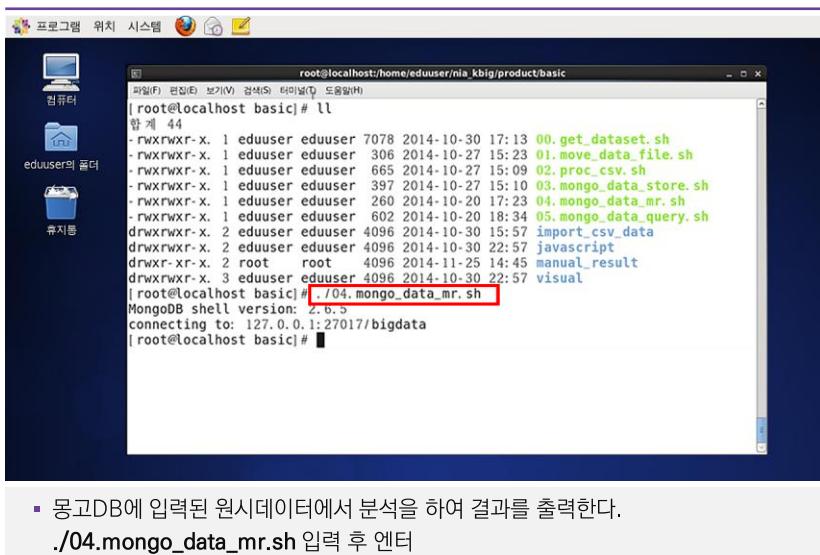
```



- 맵리듀스 분석 스크립트 소스(products_avg_price.js)
- 라인 06 : 몽고DB의 products 컬렉션을 설정하는 라인이다.
- 라인 09~14 : 일자+품명으로 고유 key를 생성하고 일자, 품목, 총합계가격, 평균가격을 객체에 삽입하는 라인이다.
- 라인 16 : map 함수를 실행하는 라인이다.
- 라인 19 : reduce 함수 선언 부분 라인이다.
- 라인 23~28 : 데이터를 읽으면서 동일한 키를 가지고 있는 품목의 총합계를 구하는 라인이다.
- 라인 29 : 가격의 평균을 구해서 reduced 객체인 avgPrice에 값을 설정하는 라인이다.
- 라인 33 : products 컬렉션에 맵리듀스 작업을 실행시키는 라인이다.
- 라인 36 : 분석 결과 데이터 저장을 products_mr_result 컬렉션으로 저장하는 라인이다.

➤ 가공 데이터 분석 맵리듀스 실행

➤ 분석 맵리듀스 실행



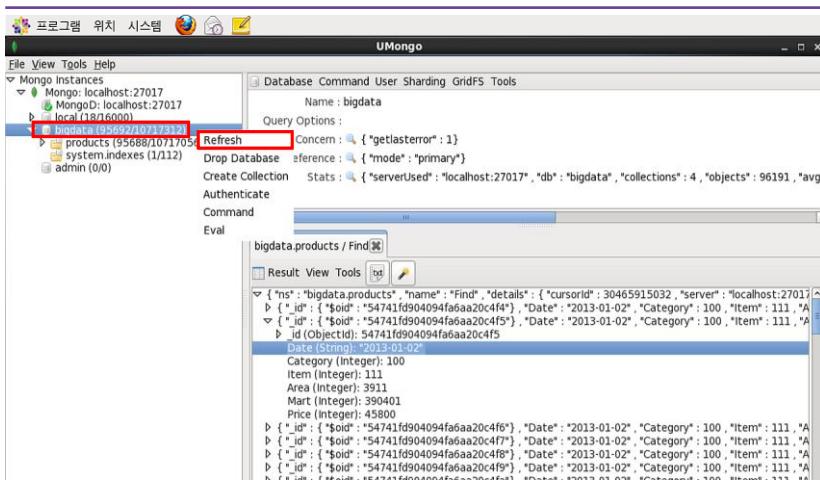
```

root@localhost basic]# ll
total 44
-rwxrwxr-x. 1 eduuser eduuser 7078 2014-10-30 17:13 00.get_dataset.sh
-rwxrwxr-x. 1 eduuser eduuser 306 2014-10-27 15:23 01.move_data_file.sh
-rwxrwxr-x. 1 eduuser eduuser 665 2014-10-27 15:09 02.proc_csv.sh
-rwxrwxr-x. 1 eduuser eduuser 397 2014-10-27 15:10 03.mongo_data_store.sh
-rwxrwxr-x. 1 eduuser eduuser 261 2014-10-20 17:23 04.mongo_data_mr.sh
-rwxrwxr-x. 1 eduuser eduuser 602 2014-10-20 18:34 05.mongo_data_query.sh
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxr-xr-x. 2 root root 4096 2014-11-25 14:45 manual_result
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[root@localhost basic]# ./04.mongo_data_mr.sh
MongoDB shell version: 2.6.5
connecting to: 127.0.0.1:27017/bigdata
[root@localhost basic]#

```

▪ 몽고DB에 입력된 원시데이터에서 분석을 하여 결과를 출력한다.
./04.mongo_data_mr.sh 입력 후 엔터

➤ 분석 결과 데이터 확인



The screenshot shows the MongoDB Management Studio interface with the following details:

- Database:** bigdata
- Collection:** products
- Query Options:** {"getlasterror": 1}
- Result View:** Shows the results of the map-reduce operation. One document is visible:


```

{
  "_id": "54741fd904094fa6aa20c4f6",
  "Category": 100,
  "Item": 111,
  "Area": 3911,
  "Mart": 390401,
  "Price": 45800
}
      
```

▪ products_mr_result 컬렉션 선택 후 마우스 오른쪽 클릭 후 find 팝업 메뉴를 클릭하면 입력된 데이터 목록을 확인할 수 있다.

▶ 분석 후 결과 데이터

▶ product_mr_result 데이터셋

```
01. {
02.   "_id" : "2013-01-02:214",
03.   "value" : {
04.     "date" : "2013-01-02",
05.     "item" : "214",
06.     "totalPrice" : 59335.0,
07.     "avgPrice" : 1023.0172413793104,
08.     "count" : 58.0
09.   }
10. }
11. {
12.   "_id" : "2013-01-02:514",
13.   "value" : {
14.     "date" : "2013-01-02",
15.     "item" : "514",
16.     "totalPrice" : 57500.0,
17.     "avgPrice" : 1554.054054054054,
18.     "count" : 37.0
19.   }
20. }
21. {
22.   "_id" : "2013-01-03:214",
23.   "value" : {
24.     "date" : "2013-01-03",
25.     "item" : "214",
26.     "totalPrice" : 65993.0,
27.     "avgPrice" : 1137.8103448275863,
28.     "count" : 58.0
29.   }
30. }
31. {
32.   "_id" : "2013-01-03:514",
33.   "value" : {
34.     "date" : "2013-01-03",
35.     "item" : "514",
36.     "totalPrice" : 57220.0,
37.     "avgPrice" : 1546.4864864864865,
38.     "count" : 37.0
39.   }
40. }
```

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화



1

2

VI 시각화

개요	45
분석 데이터 저장 스크립트	47
분석 결과 파일 저장	50
시각화 과정	51
분석 데이터 시각화	52
데이터 분석	53

VI

시각화

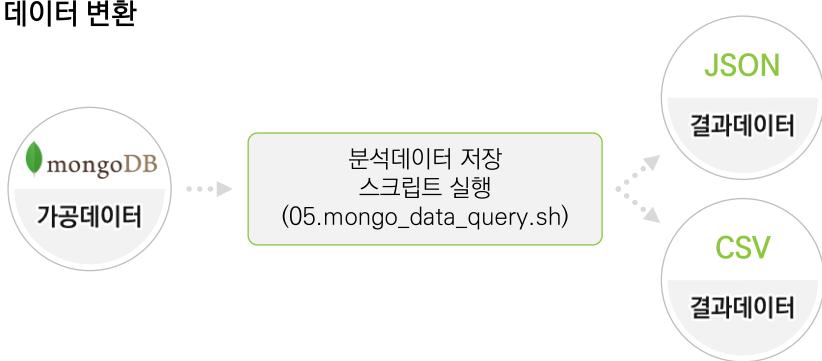
> 개요

농축산물 데이터 분석 과정에서 MongoDB에 저장된 데이터를 시각화하기 위해서 CSV, JSON 형태의 파일로 결과 데이터를 저장한다. 농축산물 데이터에서 분석된 일자별, 품목별 평균 가격 정보 데이터를 D3 챠트 라이브러리를 활용하여 꺾은선 그래프 형태로 시각화하여 2013년도의 돼지고기 가격 변화 패턴과 상추의 가격 변화 패턴을 비교 분석한다. 돼지고기 가격 변동과 상추의 가격 변동을 월별로 파악함으로써 다른 농산물과의 상호 연동하여 가격 변동 분석을 할 수 있다.

> 시각화 방법 및 활용기술

- 2013년도 돼지고기 가격과 상추 가격의 변환된 데이터를 시각화에 사용할 형태로 추출한다. 가공 변환된 데이터를 JSON 파일과 엑셀 CSV로 저장 한다.
- 돼지고기 가격 변화와 상추 가격 변화를 비교 분석하기 위해서 D3 챠트의 꺾은선 그래프를 활용한다.

▶ 데이터 변환



▶ 분석 데이터 저장 스크립트(05.mongo_data_query.sh)

▶ MongoDB 분석 결과 데이터 저장

- MongoDB에 저장된 분석된 데이터를 파일 형식으로 저장한다. 터미널 커맨드 창에서 `./05.mongo_data_query.sh` 입력 후 엔터

05.mongo_data_query.sh (분석데이터를 저장하는 커맨드)

```

01. #!/bin/bash
02. MONGO_HOST=127.0.0.1
03. MONGO_PORT=27017
04. MONGO_DATABASE=bigdata
05. OUTPUT_JS_FILE=/home/eduuser/nia_kbig/product/basic/manual_result/pr
   ↪ oduct_data.js
06. OUTPUT_CSV_FILE=/home/eduuser/nia_kbig/product/basic/manual_result/
   ↪ product_data.csv
07. # M/R 결과 컬렉션에서 돼지고기 , 상추 평균가격변화 데이터 조회 모듈(JSON
   ↪ format)
08. EXECUTE_JS_MODULE=javascript/products_mr_result_query.js
09. mongo --host $MONGO_HOST --port $MONGO_PORT --quiet --eval "var
   ↪ param='json'" $MONGO_DATABASE $EXECUTE_JS_MODULE > $OUTPUT
   _JS_FILE
10. # M/R 결과 컬렉션에서 돼지고기 , 상추 평균가격변화 데이터 조회 모듈(CSV format)
11. mongo --host $MONGO_HOST --port $MONGO_PORT --quiet --eval "var
   ↪ param='csv'" $MONGO_DATABASE $EXECUTE_JS_MODULE > $OUTPUT
   _CSV_FILE
12.

```



- 분석 데이터 저장 스크립트 소스(05.mongo_data_query.sh)
- 라인 02~04 : MongoDB 접속 정보를 설정하는 라인이다.
- 라인 05~06 : MongoDB에서 분석결과 데이터를 저장할 파일 형태를 지정한다. `product_data.js`, `product_data.csv` 2가지 형태 파일로 설정하는 라인이다.
- 라인 08 : 맵리듀스를 실행하는 스크립트(`products_mr_result_query.js`)를 지정하는 라인이다.
- 라인 09~11 : mongo 명령어를 사용하여 호스트, 포트, 데이터형식, 데이터베이스, 저장스크립트 설정하고 저장파일을 지정하여 데이터를 저장하는 라인이다.

▶ 분석 결과 출력 스크립트

javascript/products_mr_result_query.js

```
01. /**
02. * [products_mr_result] 컬렉션을 조회한다.
03. * 결과값은 JSON ARRAY 형태로 출력한다.
04. */
05. // 대상 컬렉션을 설정한다.
06. var products_mr_result = db.products_mr_result;
07. function printResultJSONArray(targetItemCode , targetItemName) {
08.     // item에 따른 총 문서의 갯수를 구한다
09.     var totalCount = products_mr_result.find({ "value.item" : targetItemCode }).count ();
10.     var startIdx = 0;
11.
12.     // item에 따른 문서를 조회한다.
13.     products_mr_result.find({ "value.item" : targetItemCode }).forEach(get_results);
14.     // 조회결과를 callback처리하며 JSON 배열 포맷을 맞춘다.
15.     function get_results (result) {
16.         if (startIdx == 0) {
17.             print("[" );
18.             print(" "item" : " + targetItemName + " ,");
19.             print(" "Data" : [ ");
20.         }
21.         var date = result.value.date;
22.         var avgPrice = result.value.avgPrice;
23.         print(" "Date" : " + date + " ,");
24.         print(" "Value" : " + avgPrice );
25.         print(" " ]");
26.         if (startIdx < totalCount -1) {
27.             print(",");
28.         } else {
29.             print(" ]");
30.         }
31.     }
32. }
```



- 분석 결과 출력 스크립트 소스(javascript/products_mr_result_query.js)
- 라인 06 : 맵리듀스 분석결과를 저장할 컬렉션을 설정하는 라인이다.
- 라인 09 : 품목에 따른 데이터의 개수를 구하는 라인이다.
- 라인 13 : 품목코드에 따른 자료를 조회하는 라인이다.
- 라인 15~35 : 품목코드로 조회한 데이터를 가지고 일자, 평균가격으로 json형태의 파일 구조를 만드는 라인이다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화

VI. 시각화

```
31.         print(']');
32.     }
33.     startIdx++;
34. }
35. }
36. function printResultCSV(targetItemCode) {
37.     // item에 따른 총 문서의 갯수를 구한다
38.     var totalCount = products_mr_result.find({ "value.item" : targetItemCode }).count();
39.
40.     // item에 따른 문서를 조회한다.
41.     products_mr_result.find({ "value.item" : targetItemCode }).forEach(get_results);
42.     // 조회결과를 callback처리하여 CSV 포맷을 맞춘다.
43.     function get_results(result) {
44.         var date = result.value.date;
45.         var avgPrice = result.value.avgPrice;
46.         print(targetItemCode + "," + date + "," + avgPrice);
47.     }
48. }
49. // 복수개의 품목코드를 지정하여 결과 값을 JSON Array로 출력한다.
50. function printAsJSON() {
51.     print("[");
52.     printResultJSONArray("214", "일반농산물 - 상추");
53.     print(",");
54.     printResultJSONArray("514", "돼지고기");
55.     print("]");
56. }
57. // 복수개의 품목코드를 지정하여 결과 값을 CSV로 출력한다.
58. function printAsCSV() {
59.     print("Item,Date,Value");
60.     printResultCSV("214");
61.     printResultCSV("514");
62. }
63.
```



- 48페이지 분석 결과 출력 스크립트 소스(javascript/products_mr_result_query.js)
- 라인 38 : 맵리듀스의 결과데이터에서 품목별 데이터 건수를 구하는 라인이다.
- 라인 41~46 : 품목의 코드에 따라 맵리듀스 분석 결과 데이터를 조회하는 라인이다.
- 라인 50~56 : 품목코드가 214인 상추와 514인 돼지고기 분석결과 데이터를 json Array 형태로 출력하는 라인이다.
- 라인 58~62 : 품목코드가 214인 상추와 514인 돼지고기 분석결과 데이터를 csv 형태로 출력하는 라인이다.

▶ 분석 결과 파일 저장

▶ 분석 결과 데이터 저장 스크립트 실행

```

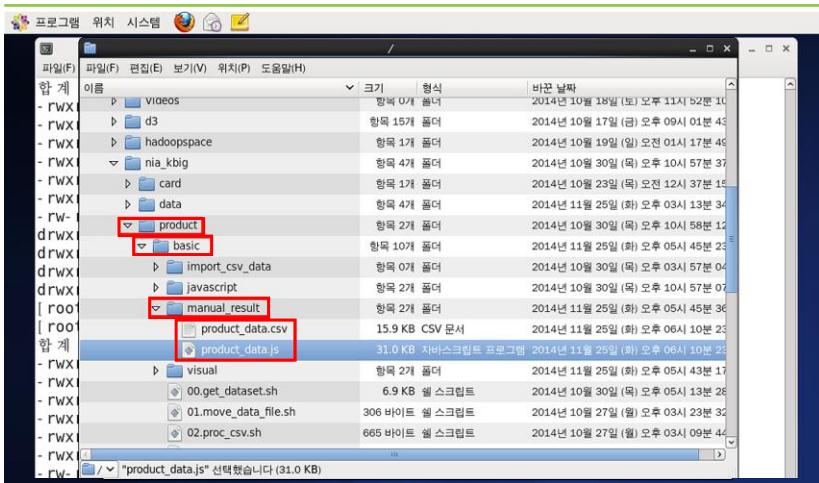
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
- rwxrwxr-x . 1 eduuser eduuser 7078 2014-10-30 17:13 00.get_dataset.sh
- rwxrwxr-x . 1 eduuser eduuser 306 2014-10-27 15:23 01.move_data_file.sh
- rwxrwxr-x . 1 eduuser eduuser 665 2014-10-27 15:09 02.proc_csv.sh
- rwxrwxr-x . 1 eduuser eduuser 397 2014-10-27 15:10 03.mongo_data_store.sh
- rwxrwxr-x . 1 eduuser eduuser 260 2014-10-20 17:23 04.mongo_data_mr.sh
- rwxrwxr-x . 1 eduuser eduuser 602 2014-10-20 18:34 05.mongo_data_query.sh
drwxrwxr-x . 2 eduuser eduuser 4096 2014-10-30 15:57 import_csv_data
drwxrwxr-x . 2 eduuser eduuser 4096 2014-10-30 22:57 javascript
drwxr-xr-x . 2 root root 4096 2014-11-25 14:45 manual_result
drwxrwxr-x . 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[root@localhost basic]# ./04.mongo_data_mr.sh
MongoDB shell version: 2.6.5
connecting to: 127.0.0.1:27017/bigdata
[root@localhost basic]# ./05.mongo_data_query.sh
[root@localhost basic]#

```

- MongoDB에 입력·저장된 분석 결과를 json 파일이나 csv 파일로 저장한다.

`./05.mongo_data_query.sh` 입력 후 엔터

▶ 분석 결과 파일 저장 폴더



- 분석 결과 저장되는 위치는 `/home/eduuser/nia_kbig/product/basic/manual_result/` 폴더 밑으로 `product_data.js`, `product_data.csv` 파일이 생성된다.

> 시각화 과정

> 시각화 절차



> 시각화 과정

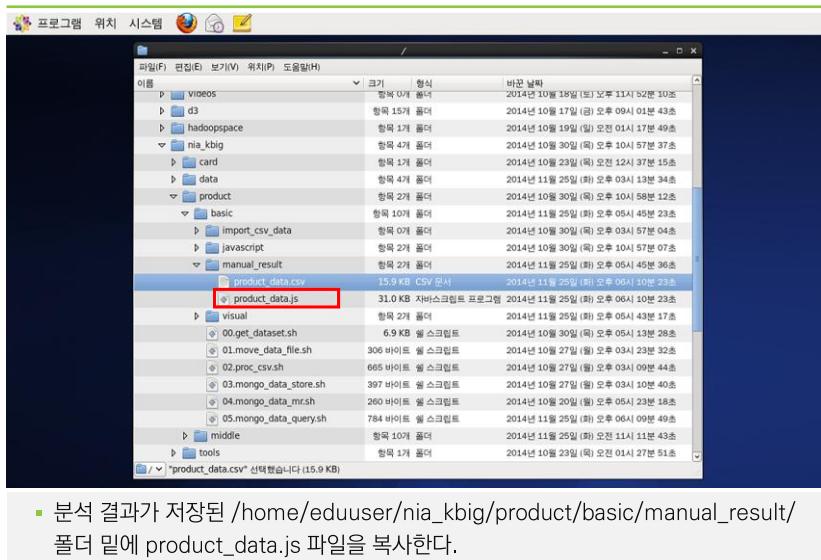
- d3.v3.js 라이브러리 파일을 제공 사이트에서 다운로드하여 저장한다.
- 시각화할 HTML 페이지를 생성한다.
- D3 차트 라이브러리 모듈에 페이지를 삽입한다.
- D3 차트에서 데이터를 읽어 오는 부분(product_data.js)에 결과 데이터를 삽입한다.
- X축과 Y축의 값을 지정한다.
- HTML 페이지를 웹 브라우저에서 실행한다.

> D3 Chart 모듈 삽입과 결과 데이터 삽입

```
01. <!-- d3 모듈을 불러온다 -->
02. <SCRIPT src="d3.v3.js"></SCRIPT>
03. <!-----d3 Chart data 연계 -->
04. <SCRIPT src="js/product_data.js"></SCRIPT>
```

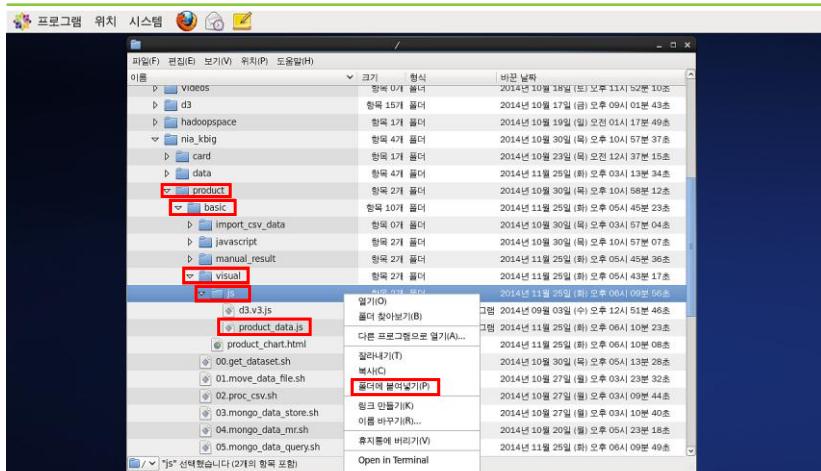
> 분석 데이터 시작화

> 분석 데이터 파일 복사



- 분석 결과가 저장된 `/home/eduuser/nia_kbig/product/basic/manual_result/` 폴더 밑에 `product_data.js` 파일을 복사한다.

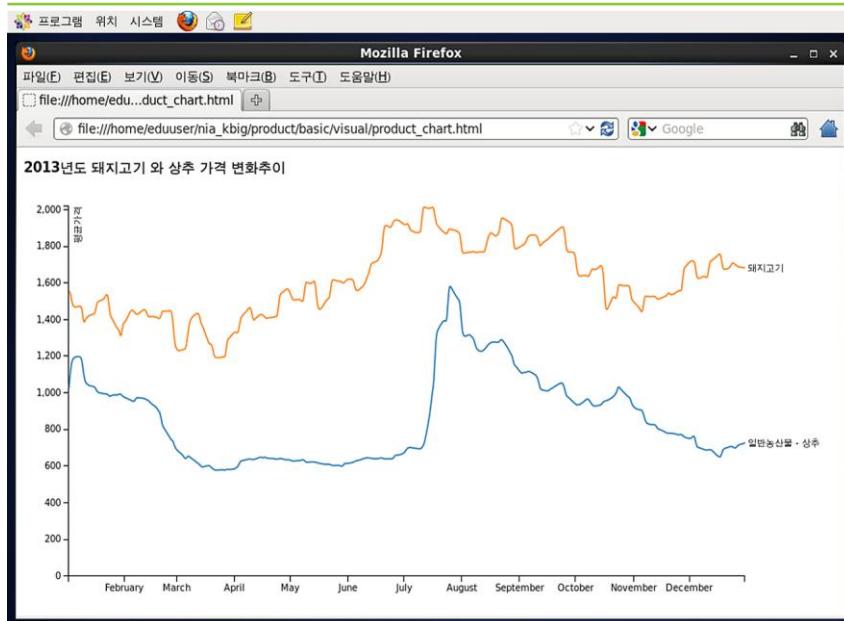
> 시각화 차트 실행



- 바로 밑에 폴더 있는 `product_chart.html` 을 더블 클릭하여 '표시' 버튼을 클릭하여 브라우저로 오픈한다.

> 데이터 분석

> 2013년도 돼지고기와 상추 가격 변화 추이



- 돼지고기와 상추의 관계는 유통 구조상 유사한 가격 패턴을 가지고 있다.
- 돼지고기와 상추의 최고점 가격은 7~8월 사이로 보이는데 이는 일반인들의 휴가 시기와 결합되어 소비가 많아지는 관계로 가격 상승이 보인다.
- 겨울 시기에 돼지고기 가격은 상승하나 상추의 가격이 하락하는 비정상적인 패턴을 보이는 현상을 목격할 수 있다.
- 분석할 때 사회 현상을 두고 예측할 수 있는 분석 결과는 12월 연말 시즌의 도래로 인하여 돼지고기의 소비가 증가하는 현상이 나타나며, 상추의 경우 출하량의 저하로 대체 야채의 유통으로 가격 하락 패턴이 발생한다는 것을 알 수 있다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화



VII 예제문제

예제 문제1

57

예제 문제2

58

예 / 제 / 문 / 제

예제 1

일반 농산물의 양파 가격변동을 분석하라.

- 일반 농산물의 양파 가격의 변동을 2011~2013년도 3년치 데이터를 분석하여 시계열 차트로 출력하라.

- 일반 농산물 원시데이터 셋에서 양파 품목을 추출한다.
- 양파의 소매가격의 일자별 평균을 구한다.
- 연도별, 월별 양파의 도매가격을 분리하여 저장한다.
- 연도별, 월별 양파의 변화 폭을 막대그래프를 사용하여 시각화한다.

예제 2

지역별/판매처별 채소류 품목의 소매가격 변화를 분석하라.

- 지역별 친환경농산물의 채소류 품목의 2013년도 가격 변화를 비교 분석하라.

- 친환경농산물의 원시데이터에서 채소류의 품목 중 2013년도 소매가격만을 추출하여 저장한다.
- 추출된 도매가격 정보에서 품목별, 지역별로 그룹화하여 월별 가격 평균을 구한다.
- 품목별, 지역별로 월별로 구한 가격을 막대그래프를 이용하여 시각화하여 비교 분석한다.



농산물 

Intermediate Level

중급과정







I 개요

개요

63

I

개요

> 개요

한국농수산식품유통공사에서 제공받은 2011년~2013년 농축산물 데이터를 바탕으로 특정 지역의 부류별, 품목별, 지역별 2013년도 농산물 가격 변화를 추출하여 추이와 패턴을 2013년도 특정 지역의 일별 평균 온도와 비교하여 가격 변화가 온도 변화에 따라 데이터 매쉬업을 통한 연관성과 패턴 분석을 하는 방법을 알아보고자 한다. 이러한 방법으로 온도에 변화에 따른 농축산물 판매 및 소비 패턴을 시계열 분석을 통하여 연관성을 파악할 수 있고, 농축산물 출하 시기를 예측 조절할 수 있는 자료로 활용할 수 있다.

> 활용 데이터

- **product.csv** : 2011~2013년 일반 농축산물 데이터
- **green_product.csv** : 2011~2013년 친환경 농산물 데이터
- **code.csv** : 코드 정보

> 선행학습

- **하둡 에코시스템** – 하둡 시작, 종료, 하둡 파일 시스템 명령어, 맵리듀스 실행 방법
- **자바** – 자바코딩, 자바컴파일, JDK 설치, jar 파일 만드는 방법
- **자바스크립트** – 객체(내장객체, 브라우저객체), 속성, 변수, 연산자(연산자 우선순위), 제어문, 함수(내장함수, 함수정의) 사용법
- **D3 차트** – D3 라이브러리 사용법, 차트 설정 방법

> 요구사항

- 분석에 사용되는 데이터는 2013년도 서울지역 돼지고기 소매가격 정보와 2013년도 서울지역 일별 평균 온도를 사용하고자 한다. 온도 변화에 따라 돼지고기의 가격 변화를 시계열로 시각화하여 온도와 돼지고기 가격과의 연관성을 파악하라.

> 분석 절차

- 수집된 2011~2013년 일반 농축산물 데이터와 기상청 데이터를 로드한다.
- 제공된 일반 농산물가격 정보에서 2013년 서울 지역 돼지고기 소매가격을 시계열 분석에 용이한 데이터 형태로 변화하기 위해 추출하여 가공된 데이터를 CSV 파일로 저장한다.
- 기상 데이터 중 서울지역 2013년 일별 평균기온을 시계열 분석에 용이한 데이터 형태로 변화하기 위해 추출하여 가공된 데이터를 CSV 파일로 저장한다.
- 추출한 2013년 서울지역 돼지고기 가격과 2013년 서울지역 평균기온 데이터를 하둡 파일 시스템에 업로드한다.
- 돼지고기 가격과 온도와의 연관성 분석을 위해서 분석 스크립트(맵리듀스)를 실행한다.
- 돼지고기 가격과 평균온도의 분석 데이터를 하둡 파일 시스템에 결과 파일을 CSV, JSON 형태로 저장한다.
- 분석된 데이터를 엑셀 형식이나 D3 챠트 형식으로 보기 위한 데이터를 서버 로컬 폴더로 저장한다.
- 저장된 JSON 파일을 불러와서 D3 챠트 중 꺾은선 그래프로 월별 돼지고기 가격과 평균온도와의 데이터를 시각화하여 가격 변동 패턴을 분석한다.



II 수집

개요	67
교육용 데이터 샘플	68
데이터 수집	69
데이터 작업 영역 이동 스크립트	72



수집

▶ 개요

농축산물 데이터는 한국농수산식품유통공사에서 제공받은 2011년~2013년 농축산물 데이터를 수집하여 분석 목적을 달성할 수 있는 한도 내에서 품목, 지역 및 마트를 비식별화 처리를 통해 분석에 용이하게 편집하여 제공한다.

농산물 소매가격정보는 일반 농산물, 친환경농산물 소매가격으로 일자, 부류, 품목, 지역, 마트 별 구분으로 구성이 되어 있다.

- 부류는 식량작물, 채소작물, 특용작물, 과일류, 축산물 항목을 의미
- 품목은 쌀, 배추, 상추, 호박 등 작물을 의미

일반 농산물에서 축산물 가격을 도매가격을 제공하고 있으며, 친환경 농산물에서는 축산물 가격을 제공하지 않는다. 농축산물 도소매가격 정보는 한국농수산식품유통공사에서 제공해 주었다. 기상 데이터는 기상청 사이트의 날씨 > 기후자료 > 과거자료 메뉴에서 기상대 지점별 기상 데이터를 확보하여 제공한다.

▶ 수집 방법

- **데이터 제공** : 농축산물 데이터는 한국농수산식품유통공사에서 제공해 준 데이터를 OpenAPI, 자료수집기(Crawler)로 데이터를 수집하였고, 실습용 자료는 빅데이터 분석 활용센터에 접속하여 농축산물 데이터 셋을 다운로드 할 수 있도록 원시데이터를 제공하고 있다.

▶ 교육용 데이터 샘플

▶ 일반 농산물 소매가격 데이터(product.csv)

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-02	500	514	1101	110401	1650
2013-01-02	200	214	1101	110402	1480
2013-01-02	300	323	1101	110212	1440
2013-01-02	200	215	1101	110251	1600
2013-01-02	500	514	1101	110403	1380
2013-01-02	500	514	1101	110406	1290
2013-01-02	500	514	1101	110407	1590
2013-01-03	500	515	1101	110212	1440
2013-01-03	500	514	1101	110251	1600
2013-01-03	500	514	1101	110401	1650

▶ 기상 데이터(weather.csv)

지역	기상구분	측정값	일자
서울(청)	평균기온	-6.8	20110101
서울(청)	평균기온	-0.2	20110201
서울(청)	평균기온	0.5	20110301
서울(청)	평균기온	9.1	20110401
서울(청)	평균기온	12.5	20110501
서울(청)	평균기온	18	20110601
서울(청)	평균기온	25.1	20110701
서울(청)	평균기온	25.6	20110801
서울(청)	평균기온	27	20110901
서울(청)	평균기온	12.7	20111001

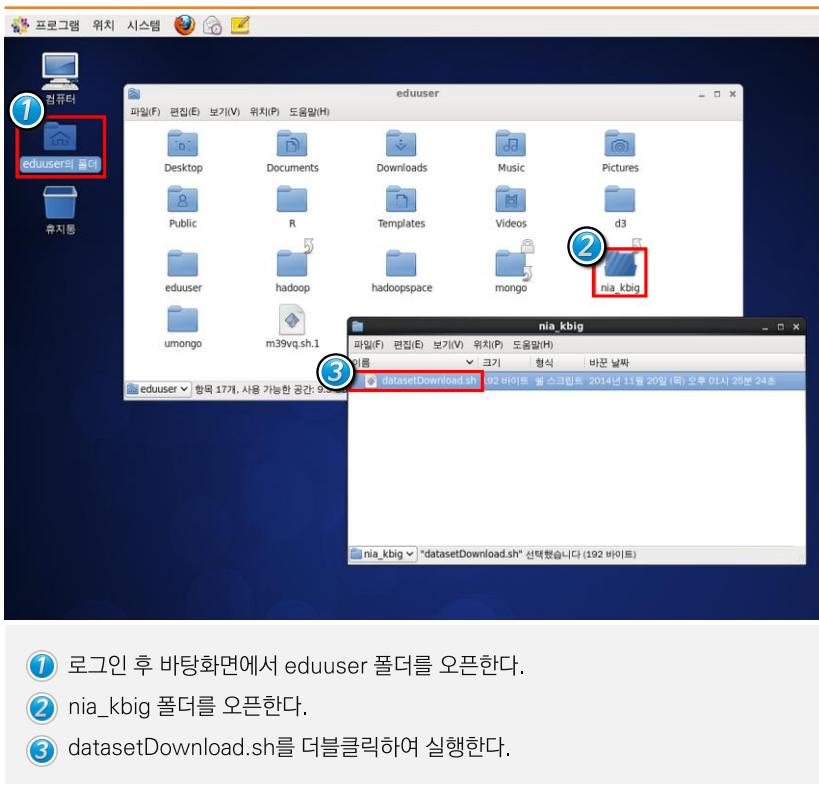
▶ 코드설명 데이터(code.csv)

코드구분	코드	코드설명
부류코드	100	식량작물
부류코드	200	채소류
부류코드	300	축용작물
부류코드	400	과일류
부류코드	500	축산물
품목코드	514	돼지고기
품목코드	515	닭고기
지역코드	1101	서울

▶ 데이터 수집(datasetDownload.sh)

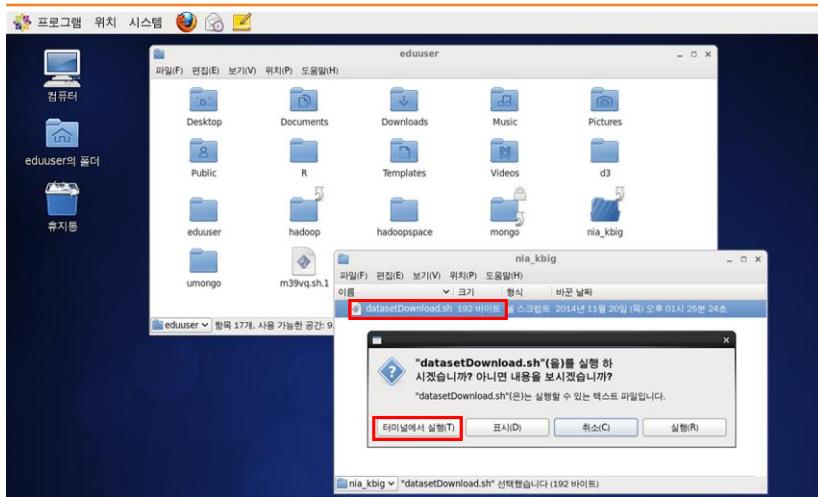
- 데이터 저장소에서 서버 로컬로 일반 농산물 데이터 셋을 복사해 온다.
 - product.csv** : 소매가격 데이터
 - green_product.csv** : 친환경농산물 소매가격 데이터
 - weather.csv** : 기상 데이터
 - code.csv** : 코드 설명 데이터

▶ 실습코드 디렉토리로 이동



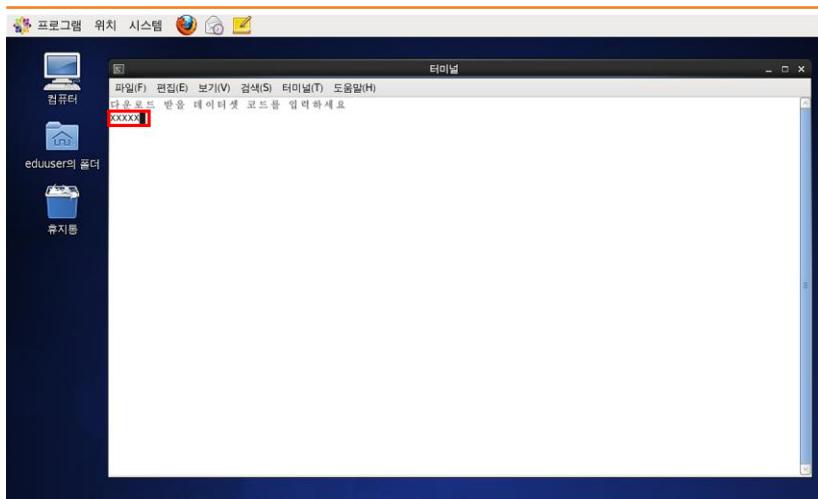
▶ 레파지토리에서 데이터 수집

datasetDownload.sh (원시데이터로 컬서버로 복사)



- '터미널에서 실행' 버튼을 클릭한다.

▶ 데이터셋 코드 입력



- 다운로드 받을 데이터셋 코드를 입력 후 엔터

➤ 데이터셋과 실습용 쉘 스크립트



- 실습용 데이터셋과 실습용 스크립트를 확인한다.

➤ ① 데이터 및 스크립트

▪ 01.move_data_file.sh :

작업영역 Data 폴더로 자료 이동하는 스크립트

▪ 02.proc_csv.sh :

원시데이터에서 분석할 대상을 추출하여 저장하는 스크립트

▪ 03.upload_csv.sh :

하둡파일시스템 가공데이터 파일 업로드 실행 스크립트

▪ 04.run.sh :

가공데이터 분석 맵리듀스 실행 스크립트

▪ datasetDownload.sh :

레파지토리에서 분석데이터와 실습용 스크립트를 다운로드하는 스크립트

▪ code.csv : 카테고리 분류 코드 데이터

▪ green_product.csv : 친환경농산물 데이터

▪ product.csv : 일반농산물 데이터

▪ wheather.csv : 기상데이터

> 데이터 작업 영역 이동 스크립트(01.move_data_file.sh)

> 데이터 이동 스크립트

- 로컬로 수집해온 데이터를 작업 영역 Data 폴더로 자료를 이동하는 스크립트

01.move_data_file.sh (작업영역 폴더로 원시데이터 이동)

```

01.#!/bin/bash
02. # 복사 대상 파일 정의
03. #일반농산물 파일
04. TARGET_PRODUCT_PRICE=/home/eduuser/nia_kbig/product/middle/product.
05. ↪ csv
06. #친환경농산물 파일
07. TARGET_GREENPRODUCT_PRICE=/home/eduuser/nia_kbig/product/middle
08. ↪ /green_product.csv
09. #코드설명 파일
10. TARGET_CODE=/home/eduuser/nia_kbig/product/middle/code.csv
11. # 기상 파일
12. TARGET_WEATHER=/home/eduuser/nia_kbig/product/middle/weather.csv
13. # 작업영역 디렉토리 정의
14. LOCAL_DIR=/home/eduuser/nia_kbig/data/
15. # 작업영역 폴더로 이동
16. mv $TARGET_PRODUCT_PRICE $LOCAL_DIR
17. mv $TARGET_GREENPRODUCT_PRICE $LOCAL_DIR
18. mv $TARGET_CODE $LOCAL_DIR
19. mv $TARGET_WEATHER $LOCAL_DIR

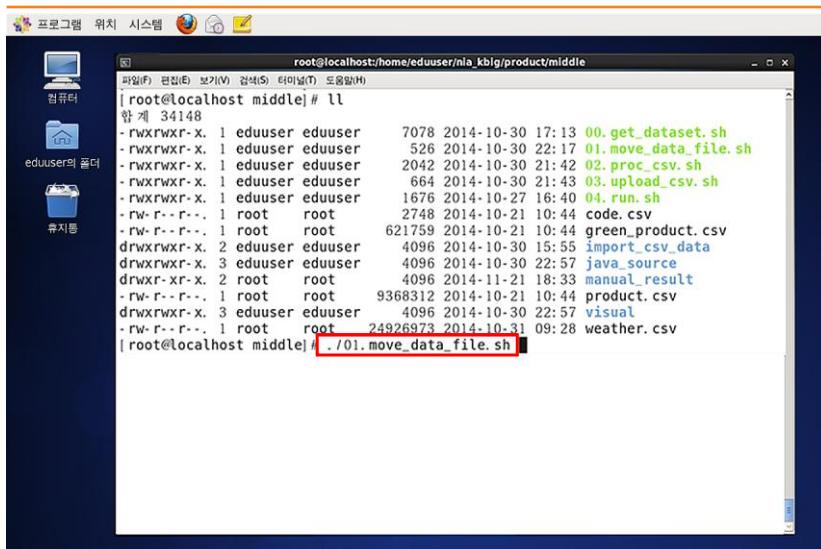
```



- 데이터 작업 영역 이동 스크립트 소스(01.move_data_file.sh)
- 라인 04~10 : 다운로드 받은 원시데이터 파일들의 위치(path)를 변수(TARGET_PRODUCT_PRICE, TARGET_GREENPRODUCT_PRICE, TARGET_CODE, TARGET_WEATHER)로 지정하는 라인이다.
- 라인 12 : 작업영역 디렉토리의 위치(path)를 변수(LOCAL_DIR)로 지정하는 라인이다.
- 라인 14~17 : mv 명령어를 사용하여 다운로드 받은 원시데이터 파일들을 작업영역 디렉토리로 이동시키는 라인이다.

II. 수집

➤ 수집 데이터 셋 작업 영역 폴더 이동



The screenshot shows a terminal window titled 'root@localhost:/home/eduuser/nia_kbig/product/middle'. The window displays a file listing from the command 'll' and then the execution of a shell script './01.move_data_file.sh'. The terminal interface includes a menu bar with Korean text and a sidebar with icons for Computer, Home, and Recycle Bin.

```
root@localhost middle# ll
합계 34148
-rwxrwxr-x. 1 eduuser eduuser      7078 2014-10-30 17:13 00.get_dataset.sh
-rwxrwxr-x. 1 eduuser eduuser      526 2014-10-30 22:17 01.move_data_file.sh
-rwxrwxr-x. 1 eduuser eduuser     2042 2014-10-30 21:42 02.proc_csv.sh
-rwxrwxr-x. 1 eduuser eduuser      664 2014-10-30 21:43 03.upload_csv.sh
-rwxrwxr-x. 1 eduuser eduuser     1676 2014-10-27 16:40 04.run.sh
-rw-r--r--. 1 root   root       2748 2014-10-21 10:44 code.csv
-rw-r--r--. 1 root   root      621758 2014-10-21 10:44 green_product.csv
drwxrwxr-x. 2 eduuser eduuser      4096 2014-10-30 15:55 import_csv_data
drwxrwxr-x. 3 eduuser eduuser      4096 2014-10-30 22:57 java_source
drwxr-xr-x. 2 root   root       4096 2014-11-21 18:33 manual_result
-rw-r--r--. 1 root   root      9368312 2014-10-21 10:44 product.csv
drwxrwxr-x. 3 eduuser eduuser      4096 2014-10-30 22:57 visual
-rw-r--r--. 1 root   root      24926973 2014-10-31 09:28 weather.csv
[root@localhost middle] ./01.move_data_file.sh
```

- 로컬에 원시데이터를 작업 영역 폴더로 이동 (/home/eduuser/nia_kbig/data/) 시킨다.
./01.move_data_file.sh 입력 후 엔터

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화





III 가공

개요

77

데이터 가공 스크립트

79



가공

> 개요

작업 영역 폴더에 복사한 일반 농산물 소매가격 데이터에서 2013년 축산물 부류의 돼지고기 가격 중 지역이 서울 지역인 데이터만을 추출하여 2013_product.csv 파일로 데이터를 저장하고, 기상 데이터는 서울지역 기상대에서 관측한 2013년도 평균온도 데이터를 추출하여 2013_weather.csv 파일로 저장한다. 돼지고기 가격과 기상의 평균 온도와의 시계열 패턴 분석에 유용한 객체 형태로 변환하도록 한다.

> 가공 방법

- 일반농산물 소매가격 데이터(product.csv) 파일에서 2013년도 서울지역 돼지고기 소매가격 데이터만 추출하여 2013_product.csv 파일을 생성한다.
- 기상 데이터(weather.csv) 파일에서 2013년도 서울지역 일별 평균기온 데이터를 추출하여 2013_weather.csv 파일을 생성한다.

▶ 데이터셋

- 2013_product.csv (2103년 서울지역 돼지고기 소매가격)

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-02	500	514	1101	110401	1650
2013-01-02	500	514	1101	110402	1480
2013-01-02	500	514	1101	110212	1440
2013-01-02	500	514	1101	110251	1600
2013-01-02	500	514	1101	110403	1380

- 2013_weather.csv (2103년 서울지역 일별 평균기온)

지역	기상구분	측정값	일자
서울(청)	평균기온	-4.7	20130101
서울(청)	평균기온	-11.7	20130102
서울(청)	평균기온	-13.2	20130103
서울(청)	평균기온	-10.7	20130104
서울(청)	평균기온	-7	20130105
서울(청)	평균기온	-6.3	20130106
서울(청)	평균기온	-5.1	20130107
서울(청)	평균기온	-4.6	20130108
서울(청)	평균기온	-9	20130109

III. 가공

▶ 데이터 가공 스크립트(02.proc_csv.sh)

- 셀 스크립트를 이용하여 2013년도 데이터만을 추출한다. 추출한 농산물 데이터는 2013_product.csv로, 기상정보 데이터는 2013_weather.csv로 저장한다.

02.proc_csv.sh (원시데이터에서 분석할 대상을 추출하여 저장)

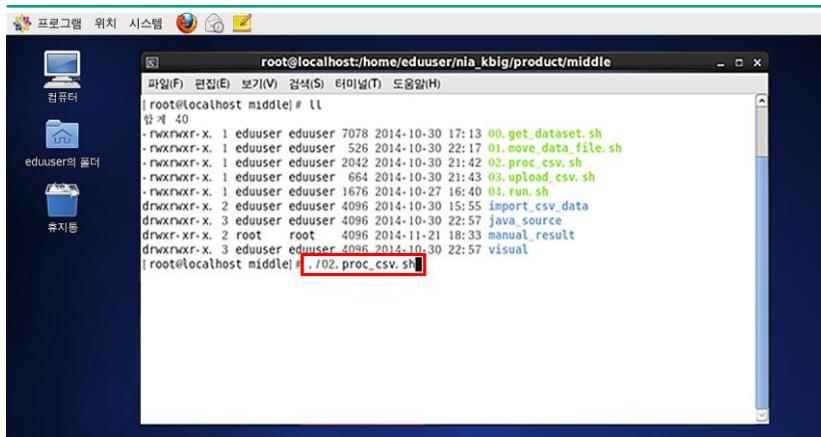
```
01.#!/bin/bash
02. # 농산물가격 정보 입력 CSV 파일 지정
03. PROD_INPUT_FILE='/home/eduuser/nia_kbig/data/product.csv'
04. # 2013년 농산물 출력결과 CSV 파일 지정
05. PROD_OUTPUT_FILE='/home/eduuser/nia_kbig/data/2013_product.csv'
06. # 기상데이터 입력 CSV 파일 지정
07. WEATHER_INPUT_FILE='/home/eduuser/nia_kbig/data/weather.csv'
08. # 2013년 서울지역 평균온도 출력결과 CSV파일 지정
09. WEATHER_OUTPUT_FILE='/home/eduuser/nia_kbig/data/2013_weather.csv'
10. TARGET_YEAR='2013'
11. # 평균기온만을 대상으로 설정
12. TARGET_TYPE='평균기온'
13. # 서울지역의 온도를 대상으로 설정
14. TARGET_AREA='서울'
15. # 2013년 서울지역 평균기온 출력결과 CSV HEADER컬럼 출력
16. echo "Date,Temperature" > $WEATHER_OUTPUT_FILE
17. # ','를 구분자로 해서 파일을 읽어들인다.
18. IFS=','
19. while read AREA TYPE VALUE DATE
20. do
21.     # 측정값이 빈 것은 SKIP처리 한다.
22.     if [ -z $VALUE ]; then
23.         continue;
24.     fi
25.     # TARGET_YEAR로 시작하는 년도인지 체크한다.
26.     # TARGET_TYPE(평균기온)인지 체크한다.
27.     # TARGET_AREA(서울)인지 체크한다.
28.     if [[ ( $DATE == ${TARGET_YEAR}* ) && ( $TYPE == ${TARGET_TYPE}* ) && ( $AREA == ${TARGET_AREA}* ) ]]; then
```



- 데이터 가공 스크립트 소스(02.proc_csv.sh)
- 라인 03 : 가공할 데이터 파일(product.csv)을 설정하는 라인이다.
- 라인 05 : 가공 후 저장할 파일명(2013_product)을 지정하는 라인이다.
- 라인 07 : 가공할 기상 데이터 파일(weather.csv)을 저장하는 라인이다.
- 라인 09 : 가공 후 저장할 기상(2013_weather.csv)을 지정하는 라인이다.
- 라인 10~14 : 추출 대상 년도(2013), 기상(평균기온), 지역(서울)을 설정하는 라인이다.
- 라인 19~28 : 추출 대상 조건에 맞는 데이터만 선정하여 파일에 저장하는 라인이다.
- 라인 34 : 2013년도 농산물 출력결과 CSV HEADER를 출력하는 라인이다.
- 라인 36~44 : 데이터를 읽어서 2013년도 농산물정보만 파일에 저장하는 라인이다.

III. 가공

▶ 원시데이터에서 분석 대상 데이터 가공



- 원시 데이터 셋에서 분석할 데이터를 가공하여 2013_product.csv 파일을 생성한다.
`./02.proc_csv.sh` 입력 후 엔터
 - /home/eduuser/nia_kbig/data/ 폴더에 2013_product.csv,
2013_weather.csv 2개 파일 생성한다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화



IV 저 장

개요	85
가공 데이터 하둡 파일시스템 업로드	86
가공 데이터 하둡 파일시스템 저장	87
하둡 파일시스템 파일 조회	88
하둡 명령어로 파일 조회	89

> 개요

하둡 파일 시스템의 하둡 맵리듀스를 이용하기 위해 분석에 필요한 자료들을 하둡 파일 시스템에 업로드 시킨다. 분석에 사용될 데이터는 원시 데이터에서 가공한 2013년도 서울지역 돼지고기 소매가격과 2013년도 서울지역 평균기온 데이터를 사용한다. 하둡의 put 명령어를 사용하여 하둡 파일시스템의 /user/bigdata/ 폴더로 자료를 업로드를 한다. 하둡 파일 시스템에 저장 시에는 한글이 있는 경우 깨지기 때문에 반드시 utf-8 포맷 형식으로 업로드를 해야 맵리듀스 분석 실행 시 오류가 발생하지 않는다.

> 저장 방법

- 2013년도 일반 농산물 소매가격 데이터 파일(2013_product.csv)과 2013년도 서울 평균기온 데이터 파일(2013_weather.csv)을 하둡에 업로드 한다.
- 하둡 커맨드를 이용해서 가공된 데이터를 하둡 파일 시스템에 업로드한다.

> 가공 데이터 하둡 파일시스템 업로드(03.upload_csv.sh)

> 하둡 파일시스템에 업로드 스크립트

- 원시데이터에서 가공된 2013_product.csv, 2013_weather.csv 파일을 하둡 시스템에 업로드한다.

03.upload_csv.sh (가공데이터를 하둡파일시스템으로 업로드)

```

01.#!/bin/bash
02. # 2013년 서울지역 돼지고기 평균가격 출력결과 CSV 파일 지정
03. PROD_OUTPUT_FILE=' /home/eduuser/nia_kbig/data/2013_product.csv '
04. # 2013년 서울지역 평균기온 출력결과 CSV파일 지정
05. WEATHER_OUTPUT_FILE='/home/eduuser/nia_kbig/data/2013_weather.csv'
06. # 하둡의 2013년 농산물 출력결과 저장 위치
07. HDFS_WEATHER=/user/bigdata/2013_weather.csv
08. # 하둡의 2013년 서울지역 평균기온 출력결과 저장 위치
09. HDFS_PRODUCT=/user/bigdata/2013_product.csv
10. # 돼지고기 가격과 기상 온도파일을 하둡의 파일 시스템에 업로드
11. hadoop fs -put $WEATHER_OUTPUT_FILE $HDFS_WEATHER
12. hadoop fs -put $PROD_OUTPUT_FILE $HDFS_PRODUCT
13.

```

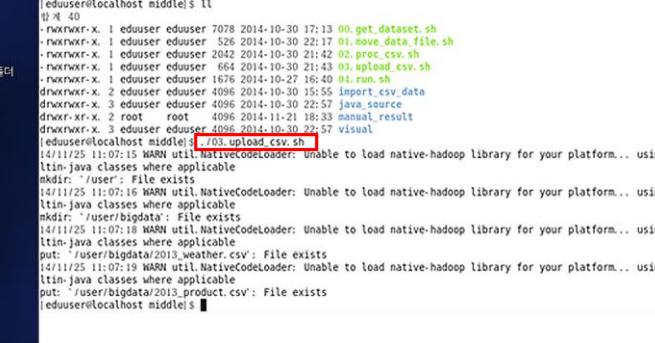


- 가공 데이터 하둡 파일시스템 업로드 스크립트 소스(03.upload_csv.sh)
- 라인 03 : 가공된 데이터 파일(2013_product.csv)을 설정하는 라인이다.
- 라인 05 : 2013년도 평균기온을(2013_weather.json)을 설정하는 라인이다.
- 라인 07 : 하둡 파일 시스템에 가공 농산물 데이터 저장 위치를 지정하는 라인이다.
- 라인 09 : 하둡 파일 시스템에 가공 기상 데이터 저장 위치를 지정하는 라인이다.
- 라인 11~12 : 하둡 파일 시스템에 가공 기상 데이터와 가공 농산물 데이터를 업로드하는 라인이다.

IV. 저장

> 가공 데이터 하둡 파일시스템 저장

> 가공 데이터 하둡 파일시스템 저장



```
edusuuser@localhost:~/nia_kbip/product/middle
```

파일(F) 리近些(ℓ) 보기(V) 검색(S) 터미널(T) 도움말(H)

```
edusuuser@localhost middle$ ll
```

합계 40

```
-rwxrwxr-x 1 edusuuser edusuuser 7078 2014-10-30 17:13 00_get_dataset.sh  
-rwxrwxr-x 1 edusuuser edusuuser 1024 2014-10-30 21:42 01_save_file.sh  
-rwxrwxr-x 1 edusuuser edusuuser 2042 2014-10-30 21:42 02_prccs_csv.sh  
-rwxrwxr-x 1 edusuuser edusuuser 664 2014-10-30 21:43 03_upload_csv.sh  
-rwxrwxr-x 1 edusuuser edusuuser 1676 2014-10-27 16:40 04_rnm.sh  
drwxrwxr-x 2 edusuuser edusuuser 4096 2014-10-30 15:55 import_csv_data  
drwxrwxr-x 3 edusuuser edusuuser 4096 2014-10-30 22:57 java_source  
drwxrwxr-x 2 root root 4096 2014-11-21 18:18 manual_result  
drwxrwxr-x 3 edusuuser edusuuser 4096 2014-10-30 22:57 visual
```

```
edusuuser@localhost middle$ ./03.upload_csv.sh
```

```
14/11/25 11:07:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in Java classes where applicable  
mkdir: '/user': File exists  
14/11/25 11:07:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in Java classes where applicable  
mkdir: '/user/bigdata': File exists  
14/11/25 11:07:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in Java classes where applicable  
put: '/user/bigdata/2013.product.csv': File exists  
14/11/25 11:07:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in Java classes where applicable  
put: '/user/bigdata/2013.product.csv': File exists
```

```
edusuuser@localhost middle$
```

- 가공한 2013_product.csv, 2013_weather.csv 파일을 하둡 파일시스템에 업로드 한다. `./03.upload_csv.sh` 입력 후 엔터

▶ 하둡 파일시스템 파일 조회

▶ 하둡 파일시스템 접속

The screenshot shows a Mozilla Firefox window with the title "HDFS:/user/bigdata - Mozilla Firefox". The address bar contains the URL "localhost:50075/browseDirectory.jsp?dir=%2Fuser%2Fbigdata&namenode". The page content is titled "Contents of directory /user/bigdata". It includes a "Goto" input field with "user/bigdata" and a "go" button. Below is a table with two rows:

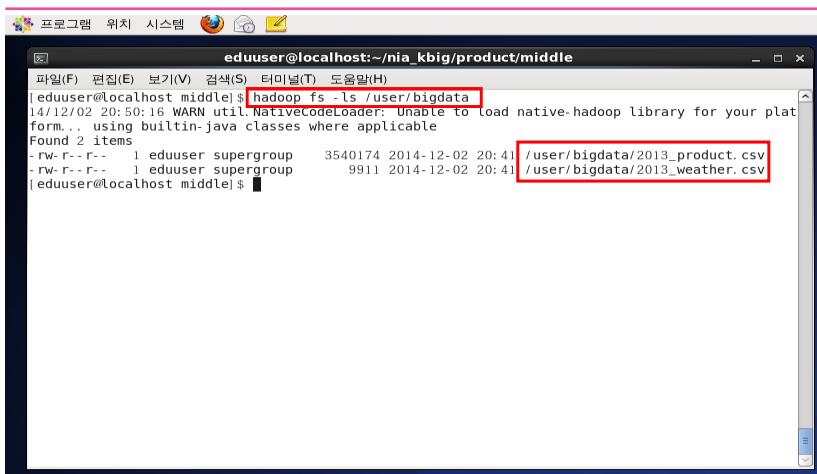
Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
2013_product.csv	file	3.38 MB	1	128 MB	2014-12-02 20:41	rw-r--r--	eduuser	supergroup
2013_weather.csv	file	9.68 KB	1	128 MB	2014-12-02 20:41	rw-r--r--	eduuser	supergroup

Below the table are links: "Go back to DFS home", "Local logs", "Log directory", and "Hadoop, 2014."

- 파이어폭스 브라우저를 클릭하여 오픈 한 후 주소창에 <http://localhost:50070> 입력 후 엔터 치면 하둡 파일 시스템에 접속할 수 있다.
- Browse the filesystem 링크를 클릭하고 user 폴더 / bigdata 폴더를 클릭하면 업로드한 가공 데이터 목록을 볼 수 있다.

> 하둡 명령어로 파일 조회

> 하둡 파일시스템 조회



```
eduuser@localhost:~/nia_kbig/product/middle
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[eduuser@localhost middle]$ hadoop fs -ls /user/bigdata
14/12/02 20:50:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 eduuser supergroup 3540174 2014-12-02 20:41 /user/bigdata/2013_product.csv
-rw-r--r-- 1 eduuser supergroup 9911 2014-12-02 20:41 /user/bigdata/2013_weather.csv
[eduuser@localhost middle]$
```

- 터미널 창에서 **hadoop fs -ls /user/bigdata** 입력 후 엔터를 치면 하둡 파일시스템에 올라간 파일을 조회할 수 있다.
- 업로드한 2013_product.csv, 2013_weather.csv 파일 목록을 확인할 수 있다.

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화

W





V 분석

개요	93
데이터 분석 스크립트	95
데이터 분석 맵리द스 실행	97
분석 데이터 파일 조회	98
분석 후 결과 데이터	99

V 분석

> 개요

농축산물 데이터의 분석은 하둡 파일 시스템의 맵리듀스 함수를 활용하여 시계열 분석에 필요한 일자별 돼지고기 평균 가격을 분석한다. 2013년도 서울지역 돼지고기 일자별 평균 가격과 2013년 서울지역 일자별 평균기온을 매쉬업하여 온도 변화에 따른 돼지고기 소매가격의 변화를 비교 분석하기 위해서 시계열분석 방법을 적용하여 패턴을 분석해 연관성을 검증해 본다. 이러한 방법으로 타 작물과 평균기온, 강우량의 기상정보를 기초로 계절별 가격 변화 추이를 분석하여 농산물의 가격 변화를 예측할 수 있다.

> 분석 방법

- 자바로 하둡 파일 시스템에 올라가 있는 2013_product.csv, 2013_weather.csv를 대상으로 맵리듀스 작업을 통해서 일자별로 돼지고기 평균 가격 데이터와 일별 평균기온 데이터를 구한다.
- 결과값은 하둡 파일 시스템의 '/user/bigdata/product/out/2013' 경로에 파일로 출력하도록 한다.
- 맵리듀스를 실행하는 프로그램은 자바를 이용해서 구현하고 products.jar로 만들어서 실행한다.
- 콘솔에 로그인해서 하둡의 yarn 커맨드로 실행하고 결과 파일을 구한다.

➤ 가공 데이터 샘플

- 2013년 서울지역 돼지고기 소매가격

일자	부류코드	품목코드	지역코드	마트코드	가격
2013-01-02	500	514	1101	110401	1650
2013-01-02	500	514	1101	110402	1480
2013-01-02	500	514	1101	110212	1440
2013-01-02	500	514	1101	110251	1600
2013-01-02	500	514	1101	110403	1380

- 2013년 서울지역 일별 평균기온

지역	기상구분	측정값	일자
서울(청)	평균기온	-4.7	20130101
서울(청)	평균기온	-11.7	20130102
서울(청)	평균기온	-13.2	20130103
서울(청)	평균기온	-10.7	20130104
서울(청)	평균기온	-7	20130105
서울(청)	평균기온	-6.3	20130106
서울(청)	평균기온	-5.1	20130107
서울(청)	평균기온	-4.6	20130108
서울(청)	평균기온	-9	20130109

➤ 데이터 분석 스크립트(04.run.sh)

➤ 가공 데이터 분석 실행

- 맵리듀스를 처리하는 프로그램은 products.java에 구현되어 있다.
- 자바 프로그램을 컴파일하여 products.jar 파일로 만든 후 yarn 커맨드를 이용해서 products.jar 파일로 맵리듀스 작업을 수행한다.
- 분석 결과는 하둡 파일시스템의 지정한 디렉토리에 저장한다.

04.run.sh (맵리듀스 실행)

```

01.#!/bin/bash
02. # 현재 위치를 지정한다.
03. CURRENT_DIR=/home/eduuser/nia_kbig/product/middle
04. # 컴파일하여 생성할 프로그램(jar) 경로를 지정한다.
05. TARGET_JAR=$CURRENT_DIR/products.jar
06. # 컴파일할 소스를 지정한다.
07. TARGET_SOURCE=$CURRENT_DIR/java_source/com/nia/hadoop/*.java
08. # jar를 생성하는데 필요한 class 파일을 지정한다.
09. TARGET_CLASSES=$CURRENT_DIR/com/nia/hadoop/*.class
10. # Hadoop상에 존재하는 농수산물 가격정보 파일을 지정한다.
11. INPUT_PRODUCT_DATA=/user/bigdata/2013_product.csv
12. # Hadoop상에 존재하는 기상(온도)정보 파일을 지정한다.
13. INPUT_WEATHER_DATA=/user/bigdata/2013_weather.csv
14. # MapReduce로 처리한 결과 데이터파일을 생성할 디렉토리를 지정한다.
15. OUTPUT_DIR=/user/bigdata/product/out/2013
16. #컴파일에 필요한 hadoop 라이브러리 패스와 함께 source를 컴파일한다.
javac -classpath /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapr
    ↪ educe-client-core-2.2.0.jar:/usr/local/hadoop/share/hadoop/common/li
        b/commons-cli-1.2.jar:/usr/local/hadoop/share/hadoop/common/hadoo
            p-common-2.2.0.jar $TARGET_SOURCE
17.
18. # 컴파일한 *.class 파일을 jar로 압축한다.
19. jar cf $TARGET_JAR $TARGET_CLASSES
20. # yarn 커맨드로 Hadoop에서 TARGET_JAR 프로그램을 돌려서 Map/Reduce를
    ↪ 실행한다.
21. yarn jar $TARGET_JAR com.nia.hadoop.products $INPUT_PRODUCT_DATA
    ↪ $INPUT_WEATHER_DATA $OUTPUT_DIR
22.

```



- 데이터 분석 스크립트 소스(04.run.sh)
- 라인 03~09 : 자바 소스를 컴파일하여 클래스 파일을 jar 파일로 압축하는 라인이다.
- 라인 11~13 : 하둡 시스템에 있는 농산물 가공 데이터와 기상 가공 데이터 파일을 지정하는 라인이다.
- 라인 17 : 자바 컴파일을 실행하는 라인이다.
- 라인 19 : jar 명령어를 이용하여 클래스 파일을 jar 파일로 압축하는 라인이다.
- yarn jar jar 압축파일명 실행 클래스명 가공농산물데이터 가공기상데이터 결과 저장 위치

Tip ↵

- 데이터 분석 스크립트(04.run.sh) 실행시, 맵리듀스 분석 실행 중 멈춤 현상 해결 방법
 - Ctrl+C 를 눌러 스크립트 실행 종료.
 - 하둡 종료 : 터미널 입력창에 stop-all.sh 입력 후 엔터.
 - 하둡 재실행 : 터미널 입력창에 start-all.sh 입력 후 엔터.
 - 하둡 실행 상태 확인 : 터미널 입력창에 jps 입력 후 엔터.
(목록 중에 NodeManager가 존재하는지 확인한다.)
- 데이터 분석 스크립트(04.run.sh) 재실행 : 터미널 입력창에 ./04.run.sh 입력 후 엔터.

I. 개요

II. 수집

III. 가공

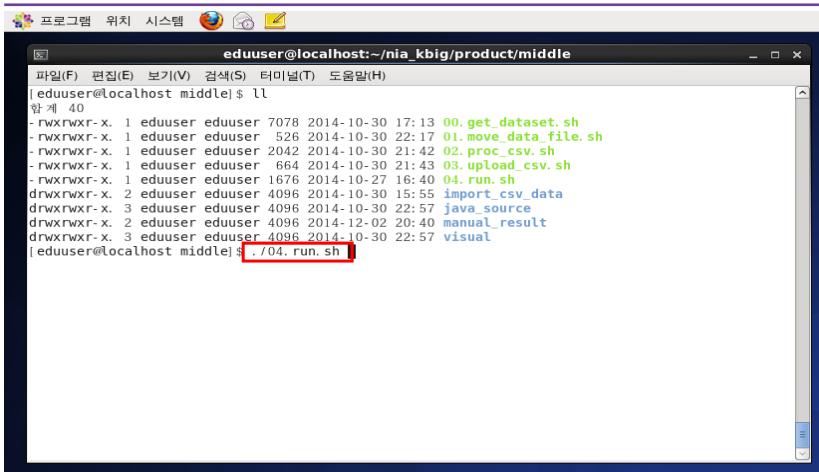
IV. 저장

V. 분석

VI. 시각화

> 데이터 분석 맵리듀스 실행

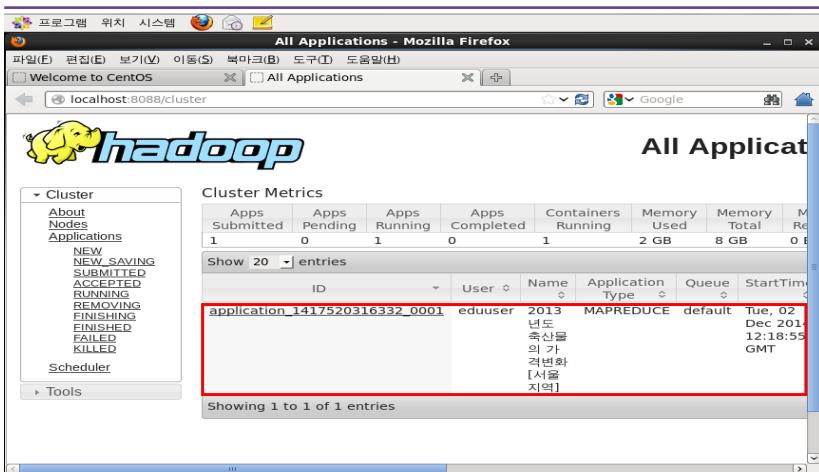
> 분석 맵리듀스 실행



```
eduuser@localhost:~/nia_kbig/product/middle
[파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
eduuser@localhost [middle]$ ll
합계 40
-rwxrwxr-x. 1 eduuser eduuser 7078 2014-10-30 17:13 00.get_dataset.sh
-rwxrwxr-x. 1 eduuser eduuser 526 2014-10-30 22:17 01.move_data_file.sh
-rwxrwxr-x. 1 eduuser eduuser 2042 2014-10-30 21:42 02.proc_csv.sh
-rwxrwxr-x. 1 eduuser eduuser 664 2014-10-30 21:43 03.upload_csv.sh
-rwxrwxr-x. 1 eduuser eduuser 1676 2014-10-27 16:40 04.run.sh
drwxrwxr-x. 2 eduuser eduuser 4096 2014-10-30 15:55 import_csv_data
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 java_source
drwxrwxr-x. 2 eduuser eduuser 4096 2014-12-02 20:49 manual_result
drwxrwxr-x. 3 eduuser eduuser 4096 2014-10-30 22:57 visual
[eduuser@localhost [middle]$ ./04.run.sh
```

- 하둡의 맵리듀스를 실행한 뒤 데이터를 분석하여 결과 파일을 하둡 파일시스템에 생성한다. ./04.run.sh 입력 후 엔터

> 맵리듀스 실행 현황 조회



All Applications - Mozilla Firefox

파일(F) 편집(E) 보기(V) 이동(S) 북마크(B) 도구(I) 도움말(H)

Welcome to CentOS

localhost:8088/cluster

hadoop

All Application

Cluster Metrics							
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Remaining
1	0	1	0	1	2 GB	8 GB	0 B

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime
application_1417520316332_0001	eduuser	2013년도 축산물의 가격변화 [서울 지역]	MAPREDUCE	default	Tue, 02 Dec 2013 12:18:55 GMT

Showing 1 to 1 of 1 entries

- 파이어폭스 브라우저를 클릭한 후 주소 입력창에 <http://localhost:8088>을 입력 후 엔터를 치면 맵리듀스 진행과정을 볼 수 있다.

▶ 분석 데이터 파일 조회

▶ 맵리듀스 분석 결과 파일 조회

NameNode 'localhost:9000' (active)

Started:	Tue Nov 25 11:06:11 KST 2014
Version:	2.2.0.15207080
Compiled:	2013-10-07T06:28Z by hortonmu from branch-2.2.0
Cluster ID:	CID-97aafe3-2002-4dad-9652-1bde0b3d55ee
Block Pool ID:	BP-1924143028-127.0.0.1-1413649470850

[Browse the filesystem](#) [NameNode Logs](#)

Cluster Summary

Security is OFF
30 files and directories, 14 blocks = 44 total.
Heap Memory used 32.23 MB is 21% of Committed Heap Memory 147 MB. Max Heap Memory is 889 MB.
Non Heap Memory used 27.21 MB is 94% of Committed Non Heap Memory 28.94 MB. Max Non Heap Memory is 214 MB.

Configured Capacity	:	66.50 GB
DFS Used	:	3.89 MB
Non DFS Used	:	12.40 GB
DFS Remaining	:	54.10 GB
DFS Used%	:	0.01%
DFS Remaining%	:	81.35%
Block Pool Used	:	3.89 MB

- 맵리듀스의 분석 결과 파일을 확인하기 위해서 파이어폭스 브라우저 창에 **localhost:50070** 입력 후 엔터
- **Browse the filesystem** 링크 클릭하여 하위 폴더로 접근하면 출력 결과물 파일이 나온다.

Contents of directory /user/bigdata/product/out/2013

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
PKCSV-r-00000	file	4.53 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
PKJSON-r-00000	file	10.38 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
WT-r-00000	file	8.13 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
.SUCCESS	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
part-r-00000	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup

[Go back to DFS home](#)

Local logs

[Log directory](#)
[Hadoop](#), 2014.

- 출력 결과 파일 /user/bigdata/product/out/2013 폴더에 PKCSV-r-00000 (돼지고기 가격 – csv 파일), PKJSON-r-00000 (돼지고기 가격 – json 파일), WT-r-00000 (평균온도 – json 파일) 파일을 조회할 수 있다.

> 분석 후 결과 데이터

> product

```

01. var data =
02. [
03.     {
04.         "item": "돼지고기", "Data": [
05.             { "Date": "2013-01-02", "Value": "1490.0" },
06.             { "Date": "2013-01-03", "Value": "1511.4285714285713" },
07.             { "Date": "2013-01-04", "Value": "1440.0" },
08.             { "Date": "2013-01-07", "Value": "1440.0" },
09.             { "Date": "2013-01-08", "Value": "1440.0" },
10.             { "Date": "2013-01-09", "Value": "1440.0" },
11.             { "Date": "2013-01-10", "Value": "1345.7142857142858" },
12.             { "Date": "2013-01-11", "Value": "1317.142857142857" },
13.             { "Date": "2013-01-14", "Value": "1374.2857142857142" },
14.             { "Date": "2013-01-15", "Value": "1374.2857142857142" },
15.             { "Date": "2013-01-16", "Value": "1365.7142857142858" },
16.             ...
17.         ]
18.     }
19. ]

```

> weather

```

01. var temperatures =
02. [
03.     ["2013-01-01", -4.7],
04.     ["2013-01-02", -11.7],
05.     ["2013-01-03", -13.2],
06.     ["2013-01-04", -10.7],
07.     ["2013-01-05", -7],
08.     ["2013-01-06", -6.3],
09.     ["2013-01-07", -5.1],
10.     ["2013-01-08", -4.6],
11.     ["2013-01-09", -9],
12.     ["2013-01-10", -8.3],
13.     ["2013-01-11", -3.2],
14.     ["2013-01-12", 0],
15.     ["2013-01-13", -0.5],
16. ]

```

I. 개요

II. 수집

III. 가공

IV. 저장

V. 분석

VI. 시각화



1

2



VI 시각화

개요	103
분석 데이터 저장 방법 1	104
분석 데이터 저장 방법 2	106
시각화 과정	110
분석 데이터 시각화	111
데이터 분석	112

VI

시각화

> 개요

2013년 서울 농축산물 데이터와 2013년 서울 기상청 평균온도 데이터를 하둡 맵리듀스로 분석한 데이터를 시각화하기 위해서 서버에 CSV, JSON 형태의 결과 파일을 저장해야 한다. 2013년 서울지역의 돼지고기 일자별 평균 가격 정보 데이터를 D3 채트 라이브러리를 활용하여 꺾은선 그래프 형태로 시각화하여 출력하고 기상 데이터의 평균 온도를 D3 채트에 같이 시각화하여 온도 변화에 따른 돼지고기 가격 변화를 비교 분석한다. 기상 데이터의 강우량과도 비교를 하여 강우량에 따른 농산물 가격 변화의 패턴을 비교 분석할 수 있다.

> 시각화 방법 및 활용기술

- 하둡의 파일시스템에 출력 결과 파일은 out/2013으로 지정한 디렉토리 아래에 PKJSON-r-00000(돼지고기, JSON포맷), PKCSV-r-00000(돼지고기, CSV포맷), WT-r-00000(평균기온)이란 이름으로 존재한다.
- 서버 로컬에 다운로드한 파일을 이용하여 product.js (JSON 배열), weather.js (자바스크립트 배열) 파일에 저장 후 product/middle/visual/js/ 폴더로 파일을 복사한 후 D3 Chart의 꺾은선 그래프를 활용하여 시각화 한다.

> 분석 데이터 저장 방법 1(05.hadoop_filecopy.sh)

> 데이터 저장(05.hadoop_filecopy.sh)

- 변환된 데이터를 저장하기 위해서 아래와 같이 저장 스크립트를 실행한다.
- 하둡 파일시스템에서 /home/eduuser/nia_kbig/product/middle/visual/js/ 폴더로 파일을 가져온다.

05.hadoop_filecopy.sh (하둡 파일시스템에서 서버 로컬로 파일 복사)

```

01. # js 빈 파일을 삭제 후 다운로드 한다.
02. rm -r visual/js/product.js
03. rm -r visual/js/product.csv
04. rm -r visual/js/weather.js
05. # 하둡 파일 시스템에서 로컬 파일로 다운로드 한다.
06. #돼지고기 가격 데이터 JSON 형태 다운로드
07. $ hadoop fs -get /user/bigdata/product/out/2013/PKJSON-r-00000
   ↪ /home/eduuser/nia_kbig/product/middle/visual/js/product.js
08. #돼지고기 가격 데이터 CSV 형태 다운로드
09. $ hadoop fs -get /user/bigdata/product/out/2013/PKCSV-r-00000
   ↪ /home/eduuser/nia_kbig/product/middle/visual/js/product.csv
10. #기상데이터 다운로드
11. $ hadoop fs -get /user/bigdata/product/out/2013/WT-r-00000
   ↪ /home/eduuser/nia_kbig/product/middle/visual/js/weather.js

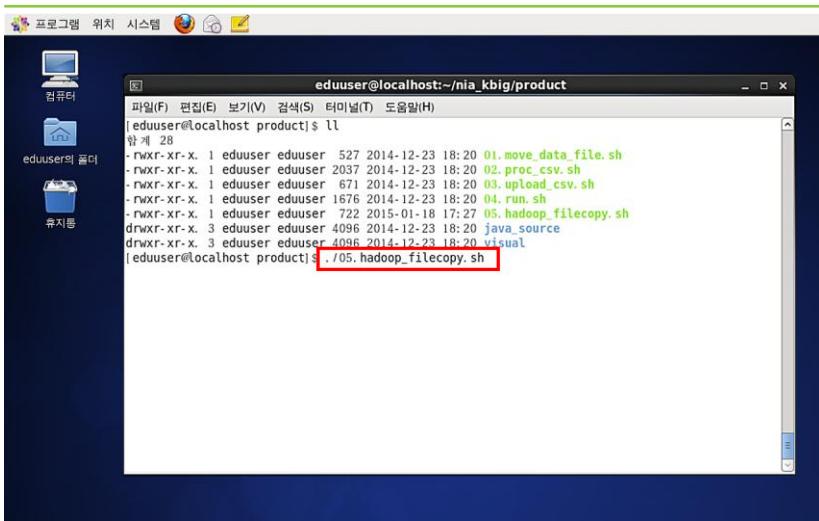
```



- 부연설명
- 분석 데이터 저장 스크립트 소스(05.hadoop_filecopy.sh)
 - 라인 02~04 : 기존에 존재하는 js, csv 빈파일을 삭제한다.
 - 라인 07 : 돼지고기 맵리듀스 분석 결과를 product.js 파일로 /home/eduuser/nia_kbig/product/middle/visual/js 폴더 위치에 저장한다.
 - 라인 09 : 돼지고기 맵리듀스 분석 결과를 product.csv 파일로 /home/eduuser/nia_kbig/product/middle/visual/js 폴더 위치에 저장한다.
 - 라인 11 : 기상 맵리듀스 분석 결과를 weather.js 파일로 /home/eduuser/nia_kbig/product/middle/visual/js 폴더 위치에 저장한다.

VI. 시각화

▶ 데이터 저장 스크립트 실행



```
eduuser@localhost product]$ ll
total 28
-rwxr-xr-x 1 eduuser eduuser 527 2014-12-23 18:20 01.move_data_file.sh
-rwxr-xr-x 1 eduuser eduuser 2037 2014-12-23 18:20 02.proc_csv.sh
-rwxr-xr-x 1 eduuser eduuser 671 2014-12-23 18:20 03.upload_csv.sh
-rwxr-xr-x 1 eduuser eduuser 1676 2014-12-23 18:20 04.run.sh
-rwxr-xr-x 1 eduuser eduuser 722 2015-01-18 17:27 05.hadoop_filecopy.sh
drwxr-xr-x 3 eduuser eduuser 4096 2014-12-23 18:20 java_source
drwxr-xr-x 3 eduuser eduuser 4096 2014-12-23 18:20 visual
[eduuser@localhost product]$ ./05.hadoop_filecopy.sh
```

- 하둡 파일 시스템에서 맵리듀스 분석 데이터를 로컬 서버 폴더 (/home/eduuser/nia_kbig/product/middle/visual/js)로 데이터를 저장한다.
- **./05.hadoop_filecopy.sh** 입력 후 엔터

> 분석 데이터 저장 방법 2

- 웹 브라우저에서 분석 데이터 확인 및 저장

> product.js 파일 만들기

Contents of directory /user/bigdata/product/out/2013

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
PKCSV-r-00000	file	4.53 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
PKJSON-r-00000	file	10.38 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
WT-r-00000	file	8.13 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
SUCCESS	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
part-r-00000	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup

Go back to DFS home

Local logs

Log directory
Hadoop, 2014.

- 맵리듀스의 결과 파일 목록에서 PKJSON-r-00000 파일 클릭한다.

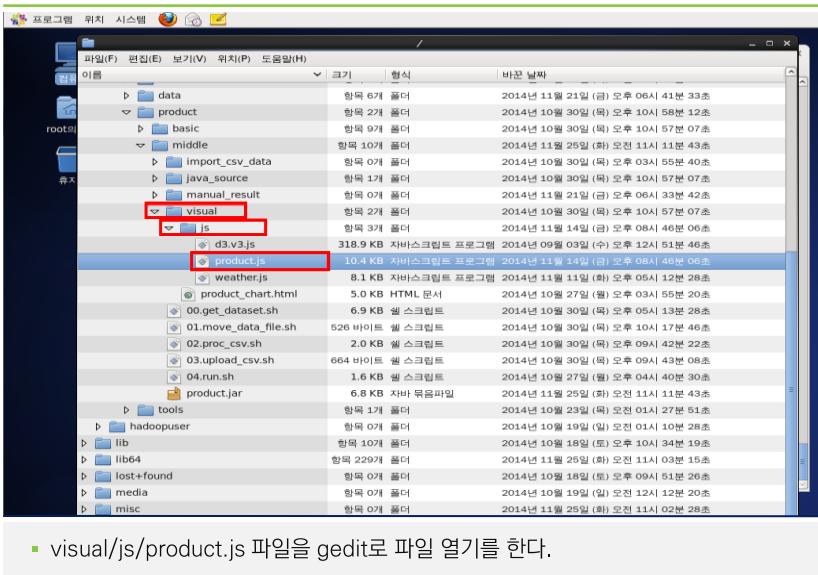
```

var data = [
    {"Date": "2013-01-02", "Value": 1490 },
    {"Date": "2013-01-03", "Value": 1511.43 },
    {"Date": "2013-01-04", "Value": 1440 },
    {"Date": "2013-01-07", "Value": 1440 },
    {"Date": "2013-01-08", "Value": 1440 },
    {"Date": "2013-01-09", "Value": 1440 },
    {"Date": "2013-01-10", "Value": 1345.71 },
    {"Date": "2013-01-11", "Value": 1317.14 },
    {"Date": "2013-01-14", "Value": 1374.29 },
    {"Date": "2013-01-15", "Value": 1374.29 },
    {"Date": "2013-01-16", "Value": 1366.73 },
    {"Date": "2013-01-17", "Value": 1345.71 },
    {"Date": "2013-01-18", "Value": 1445.71 },
    {"Date": "2013-01-21", "Value": 1458.57 },
    {"Date": "2013-01-22", "Value": 1458.57 },
    {"Date": "2013-01-23", "Value": 1458.57 },
    {"Date": "2013-01-24", "Value": 1362.86 },
    {"Date": "2013-01-25", "Value": 1294.29 },
    {"Date": "2013-01-28", "Value": 1294.29 },
    {"Date": "2013-01-29", "Value": 1294.29 },
    {"Date": "2013-01-30", "Value": 1234.29 }
]

```

- PKJSON-r-00000 파일 데이터가 출력이 되면 전체 선택 후 복사를 하여 /home/eduuser/nia_kbig/product/middle/visual/js/ 폴더 밑에 product.js 파일을 gedit로 파일 열기를 한다.

VI. 시각화



- `visual/js/product.js` 파일을 gedit로 파일 열기를 한다.

```

var data = [ { "item": "01.Jan.", "Date": "2013-01-02", "Value": 1490 }, { "item": "01.Jan.", "Date": "2013-01-03", "Value": 1511.43 }, { "item": "01.Jan.", "Date": "2013-01-04", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-05", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-06", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-07", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-08", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-09", "Value": 1440 }, { "item": "01.Jan.", "Date": "2013-01-10", "Value": 1345.71 }, { "item": "01.Jan.", "Date": "2013-01-11", "Value": 1374.14 }, { "item": "01.Jan.", "Date": "2013-01-12", "Value": 1374.14 }, { "item": "01.Jan.", "Date": "2013-01-13", "Value": 1374.29 }, { "item": "01.Jan.", "Date": "2013-01-14", "Value": 1374.29 }, { "item": "01.Jan.", "Date": "2013-01-15", "Value": 1374.29 }, { "item": "01.Jan.", "Date": "2013-01-16", "Value": 1365.71 }, { "item": "01.Jan.", "Date": "2013-01-17", "Value": 1445.71 }, { "item": "01.Jan.", "Date": "2013-01-18", "Value": 1445.71 }, { "item": "01.Jan.", "Date": "2013-01-19", "Value": 1458.57 }, { "item": "01.Jan.", "Date": "2013-01-20", "Value": 1458.57 }, { "item": "01.Jan.", "Date": "2013-01-21", "Value": 1458.57 }, { "item": "01.Jan.", "Date": "2013-01-22", "Value": 1458.57 }, { "item": "01.Jan.", "Date": "2013-01-23", "Value": 1458.57 }, { "item": "01.Jan.", "Date": "2013-01-24", "Value": 1386.86 }, { "item": "01.Jan.", "Date": "2013-01-25", "Value": 1371.11 }, { "item": "01.Jan.", "Date": "2013-01-26", "Value": 1294.29 }, { "item": "01.Jan.", "Date": "2013-01-27", "Value": 1294.29 }, { "item": "01.Jan.", "Date": "2013-01-28", "Value": 1294.29 }, { "item": "01.Jan.", "Date": "2013-01-29", "Value": 1294.29 }, { "item": "01.Jan.", "Date": "2013-01-30", "Value": 1234.29 }, { "item": "01.Jan.", "Date": "2013-01-31", "Value": 1234.29 }, { "item": "01.Jan.", "Date": "2013-02-01", "Value": 1312.86 }, { "item": "01.Jan.", "Date": "2013-02-02", "Value": 1312.86 }, { "item": "01.Jan.", "Date": "2013-02-03", "Value": 1312.86 }, { "item": "01.Jan.", "Date": "2013-02-04", "Value": 1381.43 }, { "item": "01.Jan.", "Date": "2013-02-05", "Value": 1367.14 }, { "item": "01.Jan.", "Date": "2013-02-06", "Value": 1367.14 }, { "item": "01.Jan.", "Date": "2013-02-07", "Value": 1328.0 }, { "item": "01.Jan.", "Date": "2013-02-08", "Value": 1338.57 }, { "item": "01.Jan.", "Date": "2013-02-09", "Value": 1338.57 }, { "item": "01.Jan.", "Date": "2013-02-10", "Value": 1367.14 }, { "item": "01.Jan.", "Date": "2013-02-11", "Value": 1318.57 }, { "item": "01.Jan.", "Date": "2013-02-12", "Value": 1318.57 }, { "item": "01.Jan.", "Date": "2013-02-13", "Value": 1317.14 }, { "item": "01.Jan.", "Date": "2013-02-14", "Value": 1317.14 }, { "item": "01.Jan.", "Date": "2013-02-15", "Value": 1317.14 }, { "item": "01.Jan.", "Date": "2013-02-16", "Value": 1317.14 }, { "item": "01.Jan.", "Date": "2013-02-17", "Value": 1311.43 }
]

```

- gedit로 열기 한 파일에 맵리듀스에서 분석한 JSON 파일 데이터를 복사한 것을 붙여넣기하고 저장한다.

▶ weather.js 파일 만들기

Contents of directory /user/bigdata/product/out/2013

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
PKCSV-r-00000	file	4.53 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
PKJSON-r-00000	file	10.38 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
WT-r-00000	file	8.13 KB	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
SUCCESS	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup
part-r-00000	file	0 B	1	128 MB	2014-11-25 11:12	rw-r--r--	eduuser	supergroup

Go back to DFS home

Local logs

Log directory

Hadoop, 2014.

- 맵리듀스의 결과 파일 목록에서 목록에서 WT-r-00000 파일 클릭한다.

File: /user/bigdata/product/out/2013/WT-r-00000

Goto : /user/bigdata/product/out/2[go]

Go back to dir listing
Advanced view/download options

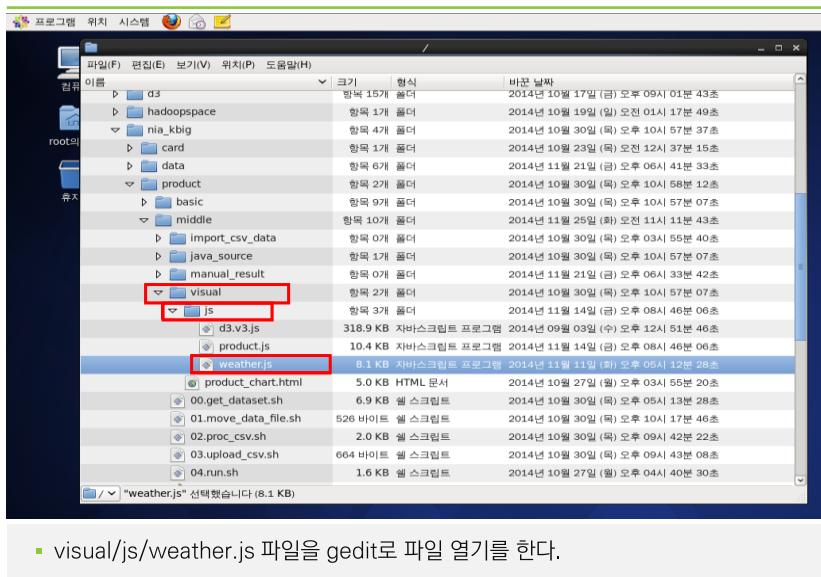
```
weather temperatures = [
  ["2013-01-01", -4.7], [
  ["2013-01-02", 11.7], [
  ["2013-01-03", 13.2], [
  ["2013-01-04", [
  ["2013-01-05", [
  ["2013-01-06", [
  ["2013-01-07", [
  ["2013-01-08", [
  ["2013-01-09", [
  ["2013-01-10", [
  ["2013-01-11", [
  ["2013-01-12", [
  ["2013-01-13", 6.0], [
  ["2013-01-14", -2.1], [
  ["2013-01-15", -2.1], [
  ["2013-01-16", -2.6], [
  ["2013-01-17", -5.8], [
  ["2013-01-18", -4.4], [
  ["2013-01-19", 0.5], [
  ["2013-01-20", 1.6], [

```

실행 취소(U)
질의나기(I)
복사(C)
붙여넣기(B)
삭제(D)
모두 선택(A)
요소 검사(Q)

- WT-r-00000 폴더에 데이터가 출력이 되면 전체 선택 후 복사하여 /home/eduuser/nia_kbig/product/middle/visual/js/ 폴더 밑에 weather.js 파일을 gedit로 파일 열기를 한다.

VI. 시각화



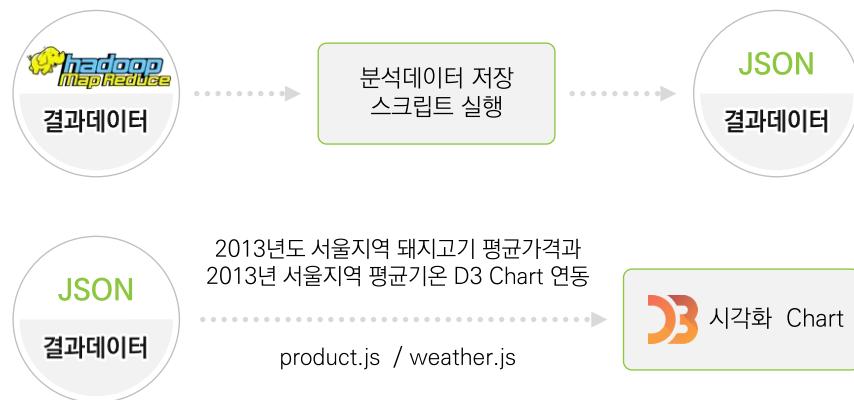
- visual/js/weather.js 파일을 gedit로 파일 열기를 한다.

```
var temperatures = [
    "2013-01-01", "-10.0",
    "2013-01-02", "-11.3",
    "2013-01-03", "-13.2",
    "2013-01-04", "-10.7",
    "2013-01-05", "-7.0",
    "2013-01-06", "-6.3",
    "2013-01-07", "-1.0",
    "2013-01-08", "-4.6",
    "2013-01-09", "-9.0",
    "2013-01-10", "-8.3",
    "2013-01-11", "-3.2",
    "2013-01-12", "0.0",
    "2013-01-13", "3.5",
    "2013-01-14", "-2.1",
    "2013-01-15", "-2.1",
    "2013-01-16", "-2.6",
    "2013-01-17", "-5.6",
    "2013-01-18", "-4.4",
    "2013-01-19", "-1.0",
    "2013-01-20", "1.6",
    "2013-01-21", "2.2",
    "2013-01-22", "3.6",
    "2013-01-23", "1.0",
    "2013-01-24", "-9.0",
    "2013-01-25", "-9.1",
    "2013-01-26", "-8.2",
    "2013-01-27", "-7.3",
    "2013-01-28", "-3.5",
    "2013-01-29", "2.2",
    "2013-01-30", "5.6",
    "2013-01-31", "7.4
];
```

- gedit로 열기 한 파일에 맵리듀스에서 분석한 JSON 파일 데이터를 복사한 것을 붙여넣기하고 저장한다.

> 시각화 과정

> 시각화 절차



> 시각화 과정

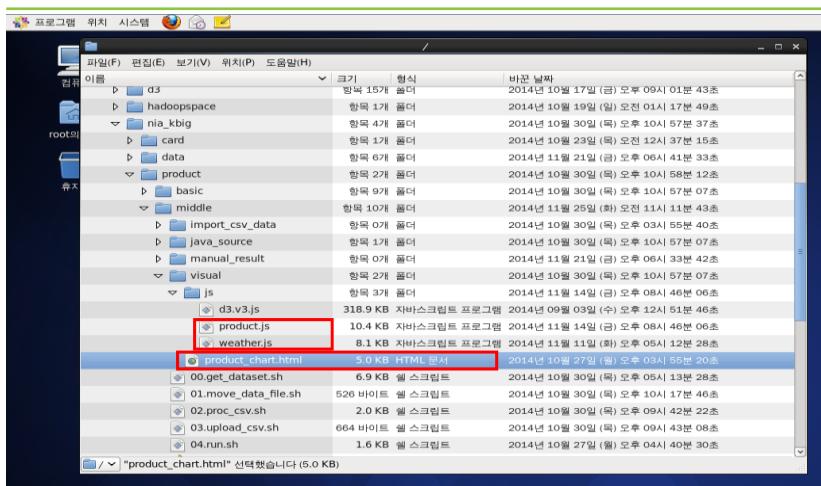
- d3.v3.js 라이브러리 파일을 제공 사이트에서 다운로드하여 저장한다.
- 시각화할 HTML 페이지를 생성한다.
D3 채트 라이브러리 모듈을 페이지에 삽입한다.
- D3 채트 데이터를 읽어 오는 부분(product_data.js)에 결과 데이터를 삽입 한다.
- X축과 Y축의 값을 지정한다.
- HTML 페이지를 웹 브라우저에서 실행한다.

> D3 Chart 모듈 삽입과 결과 데이터 삽입

01. <!-- d3 모듈을 불러온다 -->
02. <script src="[js/d3.v3.js](#)"></script>
03. <!----d3 Chart data 연계 -->
04. <script src="[js/prudct.js](#)"></script>
05. <script src="[js/weather.js](#)"></script>

> 분석 데이터 시각화

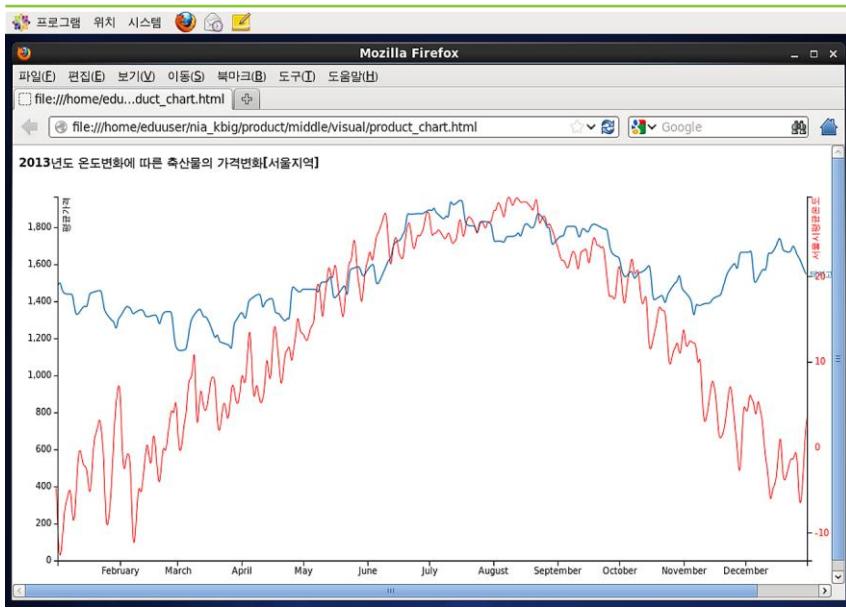
> 시각화 차트 실행



- visual 폴더에 있는 product_chart.html을 더블클릭하여 ‘표시’ 버튼을 클릭하면 브라우저로 오픈한다.

> 데이터 분석

> 2013년도 온도 변화에 따른 축산물의 가격 변화[서울지역]



- 서울지역의 돼지고기 가격과 평균온도의 분석을 통해 기온이 올라가는 4월(봄)부터 돼지고기의 가격이 상승하는 패턴을 보인다.
- 분석을 통해 평균 온도가 가장 높은 시점인 7 ~ 8월 사이의 돼지고기 가격이 높게 분포되어져 있는 것을 볼 수 있다. 이는 일반인들의 휴가 시기와 겹쳐지는 연관관계로 보이며, 이 결과로 휴가시기에 돼지고기 소비가 많아지고, 더불어 판매량 증가로 인한 가격 상승효과로 이어져 보인다.
- 9 ~ 10월 온도가 낮아지면서 돼지고기 가격이 하락하고 있는 패턴을 보이고 있다.
- 12월 가격 상승효과는 연말연시로 돼지고기의 소비량이 증가하여 가격이 상승한 것으로 보인다.



VII 예제문제

예제 문제1

115

예제 문제2

116

예 / 제 / 문 / 제

예제 1

친환경농산물 중 채소류 가격 변화와 축산물의 가격 변화를 대도시를 중심으로 비교하라.

- 친환경농산물 중 채소류의 2013년 가격 변화와 축산물의 가격 변화를 대도시를 중심으로 평균 가격을 구하고 지역 간 가격 변화를 비교하라.

- 친환경농산물(green_product)에서 2013년도 채소류의 가격을 추출한다.
- 추출된 가격에서 지역별로 그룹화하여 품목별 월별 가격을 평균을 구한다.
- 광역시와 지방도시를 구분하여 지역별 가격 변화의 편차를 비교한다.
- 공업도시와 일반도시의 가격 변화 추이를 비교한다.

예제 2

2013년도 일반 농산물의 양파 가격을 월별 비교하여 가격이 폭락했을 시점의 사회적 이슈를 뉴스 데이터와 매쉬업하여 분석하라.

- 일반 농산물 중 2013년도의 양파 가격을 추출하고 뉴스 데이터와 매쉬업하여 분석하라.

- 2013년도 양파 가격을 추출하여 일별 가격 평균을 구한다.
- 가격이 하락했을 경우의 시점을 찾아본다.
- 양파 가격 하락 시점의 지역별 가격 편차를 구해본다.
- 2013의 농산물 사회적 이슈 키워드를 추출한다.
- 뉴스 데이터에서 사회적 이슈 키워드를 분석하여 키워드 랭킹과 가격 변화의 연관성을 비교한다.

데이터 분석 콘텐츠 활용 매뉴얼

2014년 12월 인쇄

2015년 1월 발행

발 행 처 한국정보화진흥원 빅데이터전략센터

집 필 신신애, 김성현, 박재원, 김현태, 김지홍, 정다운,
이승하, 신은비

주 소 서울시 중구 청계천로 14

연 락 처 (02) 2131-0114

인 쇄 HNJ Printing

〈비매품〉

[데 이 터 분 석 콘 텐 츠]

활용 매뉴얼

NIA  한국정보화진흥원

(100-775) 서울시 종구 청계천로 14 한국정보화진흥원
TEL 02-2131-0114 FAX 02-2131-0109
www.nia.or.kr

