
بسم الله الرحمن الرحيم

**پروژه درس داده‌کاوی**

قوانین انجمنی

استاد محترم درس

جناب آقای دکتر پاینده

دانشجو

آیدا اعلابیکی

در ابتدا لازم است از زحمات بی دریغ استاد ارجمندم، جناب آقای دکتر پاینده تشکر کنم.

## فهرست مطالب

## قوانین انجمنی

قوانین انجمنی، از تکنیک‌های اصلی در داده کاوی می‌باشد که تقریبا مهمترین شکل کشف و استخراج الگوها در سیستم‌های یادگیری می‌باشد. قوانین انجمنی ارتباطات جذاب در میان مجموعه عظیمی از داده‌ها را کشف می‌نمایند که این ارتباطات می‌تواند به تصمیم گیرندگان کمک کننده باشد. قوانین انجمنی در واقع شرایطی را نشان می‌دهند که در یک مجموعه داده، به صورت مکرر با هم اتفاق می‌افتند. قوانین استخراج شده در حقیقت حضور برخی ویژگی‌ها را براساس دیگر ویژگی‌ها شرح می‌دهند.

الگوهای تکرار شونده، یکی از انواع الگوهای جذاب در مجموعه داده‌ها می باشد که شامل ترکیبی از اقلام یا اشیا است که به صورت مکرر، با هم یا در طول هم اتفاق می‌افتند. مانند مجموعه‌ای از اقلام در فروشگاه که به صورت مکرر با هم در سبد خرید مشتریان قرار می‌گیرد.

الگوهای تکرار شونده به معنی وجود وابستگی در میان داده‌ها می باشد و به قوانینی که چنین روابطی را نشان می‌دهند، قوانین وابستگی یا قوانین انجمنی گفته می‌شود.

**مثال:** در صورت خرید تلفن همراه، با احتمال 80٪ محافظ صفحه نمایش هم خریداری می‌شود.

شناسایی الگوهای تکرار شونده، با جستجو در تراکنش‌های ثبت شده در پایگاه داده، به دنبال روابط تکراری بین اقلام تراکنش‌ها می‌باشد. معمولا تراکنش‌ها، به صورت برداری از اقلام مورد بررسی با مقادیر بولین (Boolean) نمایش داده می‌شود و هدف الگوریتم، یافتن روابط تکراری در وقوع همزمان زیرمجموعه‌ای از اقلام و استخراج قوانین انجمنی می‌باشد.

بنابراین شناسایی الگوها در این مسئله شامل دو گام اصلی می‌باشد:

- تعیین مجموعه اقلام مکرر **(Frequent Itemset)**
- استخراج قانون **(Rule Extraction)**

ساده‌ترین روش تولید مجموعه اقلام مکرر این است که تمامی ترکیب‌های ممکن بین اقلام در مجموعه داده‌ها، اسکن شده و فراوانی تکرار آن‌ها مورد بررسی قرار گیرد. بدین معنی که برای n قلم کالا، بایستی دو به توان n حالت ممکن در مجموعه داده‌ها جستجو شود. طبیعی است هزینه محاسباتی این فرآیند در تعداد زیاد اقلام بسیار بالاست و نیاز به روش‌های سریع‌تر وجود دارد.

پیدا کردن چنین قوانینی می‌تواند در حوزه‌های مختلف مورد توجه بوده و کاربردهای متفاوتی داشته باشد. به‌عنوان‌مثال کشف روابط وابستگی میان حجم عظیمی از تراکنش‌های خرید می‌تواند در تشخیص تقلب، در حوزه پزشکی و شخصی‌سازی مورد استفاده قرار گیرد. در طراحی کاتالوگ، بازاریابی و دیگر مراحل فرایند تصمیم‌گیری مدیران مؤثر باشد.

مثال: جهت روشن شدن مطلب یک فروشگاه خرده‌فروشی را در نظر بگیرید. مشخصات اجناس خریداری‌شده توسط هر مشتری در یک واحد پایگاه داده ذخیره شده و به هر واحد یک شناسه نسبت داده می‌شود. فرض کنید مجموعه $I$ شامل تمام محصولات فروشگاه است.

اگر مجموعه محصولات $x,y \in I$ باشد به‌طوری‌که $x \cap y=\emptyset$ باشد. آنگاه $x \rightarrow y$ یک قانون وابستگی است که بیان می‌کند: اگر یک مشتری اجناس مجموعه $x$ را بخرد، اجناس مجموعه $y$ را نیز می‌خرد. چنین قوانینی، تأثیر مهمی در تعیین استراتژی‌های فروش و بخش‌بندی مشتریان دارد.

در اکثر اوقات تنها قوانینی برای ما جالب و مفیداند که شامل اقلامی باشند که با دفعات تکرار زیاد، نه اقلامی که به ندرت درانباره داده‌ها یافت می‌شوند . به عنوان مثال، استراتژی چیدمان اجناس یک فروشگاه با دخیل کردن اجناسی که به ندرت مشتری دارند، استراتژی موفقی نخواهد بود. بنا براین اغلب روش‌ها فرض‌شان بر این است که ما به دنبال مجموعه اقلامی هستیم که حداقل در کسر قابل قبولی از تراکنش‌ها با هم رخ دهند، به عبارتی دیگر پشتیبانی آن‌ها از معیار حداقل پشتیبانی مورد نظر ما کمتر نباشد.

اصطلاح مجموعه اقلام پر تکرار برای اقلام با پشتیبانی بالا به کار می‌رود . درجه اطمینان یک قانون $x \rightarrow y$ هم به صورت نسبت تعداد دفعات تکرار همزمان $X , Y$ به تعداد تکرار $X$ به تنهایی تعریف می‌شود، یعنی کسری از تراکنش‌های شامل $X$ که $Y$ را نیز شامل می‌شوند. در مجموع قوانینی مورد قبول واقع می‌شوند که مقادیر قابل قبولی برای هر دو معیار فوق داشته باشند.در مسائل داده کاوی، حجم داده‌ها معمولا آنقدرزیاد است که قابل بار شدن در حافظه اصلی نمی‌باشد. بنابراین در ارزیابی عملکرد روش‌های مختلف کاوش قوانین، معیارهایی از قبیل زمان مورد نیاز برای خواندن داده‌ها از دیسک و یا تعداد دفعاتی که هرجزء داده باید خوانده شود، بکار می‌روند.

می‌توان برای تمامی قوانین انجمنی تولیدشده فاکتورهای پشتیبانی و قابلیت اطمینان را محاسبه نمود. هدف کلی در کشف قوانین انجمنی، استخراج قانون‌هایی از جدول اطلاعات است که دارای میزان پوشش اعتبار قابل‌قبول باشند. میزان فاکتورهای پوشش اطمینان مورد نظر توسط کارشناسان مربوطه تعیین می‌شوند. بنابراین تمامی قانون‌ها باید دارای حداقل پوشش و حداقل اطمینان تعیین‌شده باشند. برای شناسایی این قانون‌ها روش‌های مختلفی وجود دارد که سه مورد از آن‌ها عبارتند از:

- استخراج قوانین انجمنی با محاسبه مقدار lift
- استخراج حداقل قوانین انجمنی غیر زائد
- استخراج قوانین انجمنی غیر زائد Top-K

## الگوریتم‌های کاوش الگوهای مکرر

- Apriori
- FP-tree
- DIC
- ECLAT
- Tree-projection
- H-mine
- Partition
- Sampling-based
- CHARM

الگوریتم‌های زیادی برای کشف قوانین انجمنی تاکنون ارائه شده‌اند. بخش عمده و نسبتا زمان‌گیر در اکثر الگوریتم‌های موجود از جمله روش پایه‌ای و معروف Apriori جستجو اقلام پر تکرار است. بعد از کشف کلیه اقلام پر تکرار از مجموعه داده‌ها گام بعد یعنی تولید به روش مستقیم و سریع صورت می‌گیرد. بنابراین روش‌های مختلفی که ارائه می‌شوند درواقع تفاوتشان در نحوه کشف اقلام پرتکرار است.

برای این منظور بعضی روش‌ها با یافتن راه‌های مستقیم جهت به دست آوردن پشتیبانی بعضی اقلام از مراجعات بیهوده به دیسک خودداری می‌کنند و برخی با ساختن ساختمان داده‌های خاصی در حافظه اصلی تا حدی توانسته‌اند به این منظور دست یابند. از کاراترین روش‌های موجود می‌توان از روش‌های VIPER، ARMORو FP-Growth نام برد. VIPER روشی است که به دلیل مراجعات نسبتا زیاد به دیسک کارایی‌اش نسبت به چند مورد از روش‌های دیگر پایین‌تر است. ARMOR نسخه بهبود یافته الگوریتم دیگری به نامOracle است که از ساختمان داده‌های به خصوصی به نام DAG استفاده میکند و تاکیدش بر بهینه کردن عمل شمارش دفعات تکراراست.

بر خلاف روش‌های VIPER و ARMOR که الگوی عملکردشان کاملا بر پایه Apriori است، روش FP-Growth در فرآیند جستجوی اقلام پرتکرارهیچ قلم کاندیدی تولید نمی‌کند. در این الگوریتم درمجموع داده‌ها 3 بار پیمایش می‌شوند و بعد از آن ساختمان داده خاصی از جنس درخت Hash در حافظه ساخته می‌شود که تمام اقلام پر تکرار را می توان مستقیما با پیمایش‌های خاصی بر روی این درخت به دست آورد، بدون اینکه نیاز به تولید اقلام کاندید باشد. اشکال عمده این روش نیاز به حافظه زیاد در ارتباط با مجموعه داده‌های بسیار حجیم است که عملا در بعضی مواقع الگوریتم را غیر عملی می‌سازد. روش‌های متعدد دیگری نیز وجود دارند که هریک با دیدگاهی نسبتا متفاوت به کاوش اقلام پر تکرارمی‌پردازند.

## الگوریتم Apriori

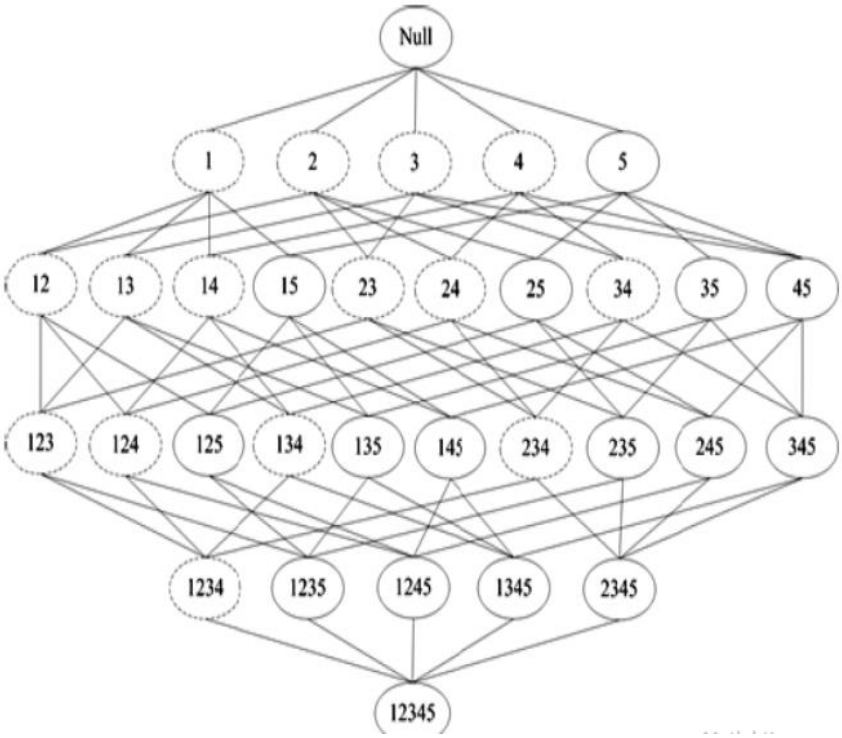
الگوریتم اپریوری (Apriori) از اولین الگوریتم‌هایی است که جهت یافتن مجموعه اقلام مکرر از آن استفاده می‌شود. نام آن برگرفته از شیوه‌های است که از آن استفاده می‌کند، یعنی استفاده از دانش مرحله قبل که در ادامه آن را شرح می‌دهیم. الگوریتم اپریوری توسط اگراوال (Agrawal) و همکاران، در مرکز تحقیقات IBM Almaden کشف شد و می‌توان آن را برای تولید کلیه مجموعه اقلام مکرر بکار برد.

الگوریتم Apriori یک الگوریتم جستجوی سطحی است، که با پایان کاوش در مرحله k ام به مرحله بعدی یعنی k+1 می‌رود. این عمل تا محقق شدن شرط یا شروط پایانی تکرار می‌شود. در مرحله k ام مجموعه اقلام k تایی تولید خواهند شد. پس از محاسبه مقدار پشتیبان برای هرکدام و مقایسه آن با مقدار minsup الگوی‌های مکرر k تایی شناسایی می‌شود.

در مرحله بعدی الگوریتم با کمک الگوهای مکرر k تایی، مجموعه اقلام (k+1) تایی کاندید که بالقوه می‌توانند مکرر باشند را تولید می‌کند. به همین ترتیب با توجه به مقدار minsup برخی حذف شده و مجموعه اقلام مکرر (k+1) تایی تشکیل خواهند شد. این عمل تا یافتن آخرین مجموعه قلم مکرر ادامه پیدا می‌کند.

این الگوریتم در حین اجرا از قاعده‌ای موسوم به قاعده Apriori استفاده می کند که بدین صورت بیان می‌شود: "اگر یک الگوی مکرر داشته باشیم، کلیه زیرمجموعه های آن نیز مکرر هستند." به عبارت دیگر اگر مجموعه اقلام I مکرر نباشد، هر مجموعه‌ای که شامل I است نیز نمی‌تواند مکرر باشد.

با کمک قاعده Apriori فضای جستجو کاهش می‌یابد. شکل زیر کل فضای جستجو را برای مجموعه اقلامی که از {۱,۲,۳,۴,۵} استفاده کرده است را نشان می‌دهد. برای سادگی الگوها به شکل یک عدد نمایش داده شده است. برای مثال الگوی {۱,۲,۳} را به صورت ۱۲۳ نشان داده‌ایم. دقت کنید که در شکل الگوهای مکرر با دایره‌های خط‌چین مشخص شده‌اند. با نگاه به الگوی مکرر {۱,۲,۳,۴} که در شکل به صورت ۱۲۳۴ نشان داده‌ایم، متوجه خواهید شد که همه زیرمجموعه‌های آن نیز مکرر هستند. یا اینکه کلیه‌ی مجموعه اقلامی که شامل {۵} هستند، نمی‌توانند مکرر باشند. چون {۵} مکرر نیست. بدین ترتیب استفاده از این استراتژی باعث کاهش فضای جستجو درتولید اقلام مکرر می‌شود.




قاعده Apriori به گروه خاصی از قواعد که دارای خاصیت پادیکنواختی هستند، تعلق دارد. این خاصیت به صورت خلاصه بدین صورت بیان می‌شود که اگر مجموعه نتواند در یک آزمون موفق باشد، کلیه اَبر مجموعه های آن نیز در همان آزمون با شکست مواجه می‌شوند.

فرض کنید J می تواند هر نمونه‌ای از مجموعه اقلامی باشد که از مجموعه I منتج می‌شود. یک مقیاس f دارای خاصیت یکنواختی است اگر:

$$\forall X, Y \in J : (X \subseteq Y) \Rightarrow f(X) \leq f(Y)$$

که نشان می دهد اگر X زیر مجموعه Y باشد، بنابراین f(X) نباید بزرگتر از f(Y) باشد. در مقابل f دارای خاصیت پاد یکنواختی است اگر:

$$\forall X, Y \in J : (X \subseteq Y) \Rightarrow f(Y) \leq f(X)$$

هر مقیاس و قاعد های همانند اپریوری (Apriori) که دارای خاصیت پاد یکنواختی است، می تواند برای الگوریتم‌های داده کاوی مثل تولید مجموعه اقلام مکرر موثر باشد. جدول زیر یک پایگاه داده تراکنشی با ۵ تراکنش و ۱۱ قلم داده را نشان می‌دهد. با تنظیم مقدار minsup برابر با 0/6 درصد و با کمک قاعده اپریوری می‌خواهیم الگوهای مکرر را در این پایگاه داده تولید کنیم.

| TID | Items |
|---|---|
| *1* | *{I1,I2,I3,I4,I5,I6}* |
| *2* | *{I2,I3,I4,I5,I6,I7}* |
| *3* | *{I1,I4,I5,I8}* |
| *4* | *{I1,I4,I6,I9,I10}* |
| *5* | *{I2,I4,I5,I10,I11}* |

با پیمایش پایگاه داده‌ها و با توجه به مقدار minsup که برابر با 0/6 درصد (معادل ۳ تکرار از میان ۵ تراکنش) است، الگوهای مکرر ۱تایی یا یک عضوی بدست می‌آیند (شکل زیر). همانطور که در شکل زیر مشخص است از میان ۱۱ قلم داده فقط ۵ قلم مکرر هستند، که در ستون سوم جدول علامت خورده‌اند. در مرحله دوم با کمک این مجموعه اقلام مکرر و الحاق آن‌ها مجموعه اقلام کاندیدی تولید خواهند شد که می‌توانند بالقوه مکرر باشد.

| 1-itemsets | Support | Frequent 1-itemset |
|---|---|---|
| *I1* | *3 (60%)* | √ |
| *I2* | *3 (60%)* | √ |
| *I3* | *2 (40%)* | – |
| *I4* | *5 (100%)* | √ |
| *I5* | *4 (80%)* | √ |
| *I6* | *3 (60%)* | √ |
| *I7* | *1 (20%)* | – |
| *I8* | *1 (20%)* | – |
| *I9* | *1 (20%)* | – |
| *I10* | *2 (40%)* | – |
| *I11* | *1 (20%)* | – |

چنانچه الگوریتم بدون توجه به اقلام مکرر ۱ تایی قصد تولید اقلام ۲تایی را داشت، باید ۵۵ مجموعه قلم ۲تایی را ایجاد می‌کرد.

| 2-itemsets | Support | Frequent 2-itemset |
|---|---|---|
| *{I1,I2}* | *1 (20%)* | – |
| *{I1,I4}* | *3 (60%)* | √ |
| *{I1,I5}* | *2 (40%)* | – |
| *{I1,I6}* | *2 (40%)* | – |
| *{I2,I4}* | *3 (60%)* | √ |
| *{I2,I5}* | *3 (60%)* | √ |
| *{I2,I6}* | *2 (40%)* | – |
| *{I4,I5}* | *4 (80%)* | √ |
| *{I4,I6}* | *3 (60%)* | √ |
| *{I5,I6}* | *2 (40%)* | – |

این در حالی است که با کمک الگوهای مکرر ۱ تایی فقط تعداد ۱۶ مجموعه اقلام ۲ تایی ساخته شده است و این نکته کاهش فضای جستجو را نشان می‌دهد. در این مرحله نیز پایگاه داده برای محاسبه مقدار پشتیبان مجموعه اقلام ۲ تایی موجود در شکل بالا پیمایش می‌شود. پس از حذف مجموعه اقلام ۲ تایی که مقدار پشتیبان آن‌ها از حد آستانه (مقدار ۳) کمتر است، مجموعه اقلام مکرر شناسایی می‌شوند. بعد از این با الحاق الگوهای مکرر ۲ تایی باید مجموعه اقلام ۳ تایی که مستعد مکرر بودن هستند، تولید شوند. دو نکته مهم در پیاده‌سازی این مرحله به بعد باید رعایت شود. در این مرحله شما مجاز به الحاق دو الگوی ۲ تایی هستید که نتیجه یک مجموعه اقلام ۳ تایی باشد.

برای مثال با پیوند الگوهای مکرر {I1,I4} و {I2,I4} به مجموعه اقلام ۳ تایی {I1,I2,4} می رسیم. در حالی‌که با پیوند دادن {I1,I4} و {I2,I5} یک مجموعه قلم ۴ تایی {I1,I2,I4,I5} تولید می شود. فراموش نکنید که ترتیب قرار گرفتن اقلام در الگو مهم نیستند. نکته دوم اینکه در حین ایجاد مجموعه اقلام ۳ تایی از قانون Apriori استفاده می‌کنیم. باید مجموعه اقلامی تولید شود که تمام زیر مجموعه های آن مکرر هستند. برای مثال از الحاق {I2,I4} و {I4,I6} که هر دوی آنها مکرر هستند، مجموعه {I2,I4,I6} بدست می‌آید. اما از آنجا که الگوی {I4,I6} مکرر نیست، بدون محاسبه مقدار پشتیبان می‌توان فهمید که {I2,I4,I6} نیز نمی‌تواند مکرر باشد.

راه حل ساده‌ی دیگر برای تولید مجموعه اقلام کاندید با طول ۳، الحاق الگوهای مکرر ۲ تایی و الگوهای مکرر ۱تایی است. این کار می‌تواند برای ساخت کاندیداهایی با طول بالاتر نیز استفاده شود، به نحوی که جهت ساخت کاندیدی با طول n کافی است الگوهای مکرر (n-1) تایی با الگوهای مکرر ۱تایی ترکیب شوند. شکل زیر نتایج مرحله سوم الگوریتم را نشان می‌دهد. در شکل زیر کلیه مجموعه اقلام ۳ تایی که از پیوند الگوهای مکرر ۲ تایی بدست آمده‌اند، نشان داده شده است. به جز برای {I2,I4,I5} لازم نیست الگوریتم مقدار پشتیبان را برای دیگر مجموعه ها محاسبه کند.

زیرا برای هر یک از آن‌ها حداقل یک زیر مجموعه وجود دارد که مکرر نیست. به همین دلیل لزومی ندارد مقدار پشتیبان برای آن‌ها محاسبه شود و با استناد به قاعده Apriori کنار گذارده می‌شوند و نیازی به پیمایش داده‌ها نیست. پس از این مرحله مجموعه اقلام بزرگتری یافت نمی‌شود تا الگوریتم به کار خود ادامه دهد، لذا الگوریتم متوقف می‌شود. بنابراین بزرگترین الگوی مکرر برای مثال مزبور برابر با ۳ خواهد بود.

در این مثال ساده با کمک قانون Apriori فقط تعداد (۶+۱۰+۱۱)=۲۷ مجموعه اقلام تولید شد. در حالی که اگر از این قانون استفاده نمی‌شد، مجبور به تولید

$$\binom{11}{1}+\binom{11}{2}+\binom{11}{3}=231$$

مجموعه قلم با حداکثر طول ۳ خواهیم بود. حتی این مثال ساده نشان می‌دهد که چگونه فضای جستجو با کمک این قانون می‌تواند بطور قابل ملاحظه‌ای کاهش یابد. الگوریتم Apriori در هر مرحله دو عملیات انجام می‌دهد. ابتدا الگوریتم با الحاق الگوهای مکرر k تایی به تولید مجموعه اقلام (k+1) تایی می‌پردازد. همانطور که قبل از این نیز گفته شد، در این مرحله می‌توان برای ساخت مجموعه اقلام (k+1) تایی، هر الگوی مکرر k تایی را با الگو های مکرر ۱ تایی الحاق نمود. سپس با کمک قاعده Apriori برخی از مجموعه اقلام حذف و با محاسبه مقدار پشتیبان برای مابقی، الگوهای مکرر تشخیص داده می‌شوند.

## الگوریتم (FP-Growth)

این الگوریتم بر خلاف Apriori ابتدا تراکنش‌ها در یک ساختار درختی به نام درخت تراکنش‌های غربال شده ذخیره می‌کند و سپس این ساختار را برای یافتن الگوهای تکرار شونده توسط این الگوریتم را نشان می‌دهد. فرض کنید که پایگاه داده تراکنش‌ها ستون‌های دوم و سوم جدول 1 باشد و آستانه تکرار شوندگی نیزسه (3) باشد.

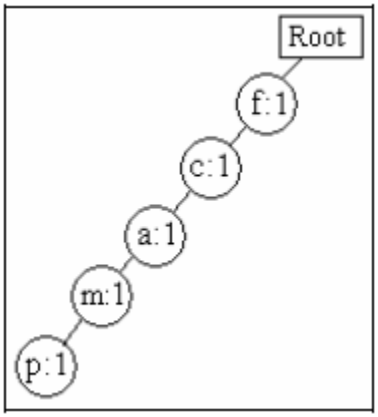
**جدول تراکنش های پایگاه داده**

| TID | Items | Ordered Frequent Items |
|---|---|---|
| 100 | f, a, c, d, g, i, m, p | f, c, a, m, p |
| 200 | a, b, c, f, l, m, o | f, c, a, b, m |
| 300 | b, f, h, j, o | f, b |
| 400 | b, c, k, s, p | c, b, p |
| 500 | a, f, c, e, l, p, m, n | f, c, a, m, p |

در این الگوریتم ابتدا عناصر تکرارشونده به طول یک به صورت مجموعه‌ی [(f:4),(c;4),(a:3),(b:3),(m:3),(p:3)] پیدا می‌شوند. در این مجموعه اعداد کنار عناصر میزان تکرار شوندگی آن‌ها را نشان می‌دهد . عناصر تکرار شونده یافت شده را به ترتیب نزولی در جدولی به نام جدول سر آیند قرار می‌دهیم که ورودی های آن به گره‌های متناظر در درخت تراکنش‌های غربال شده تراکنش‌های غربال شده اشاره می‌کند. از این جدول که نمونه ای از آن در شکل آمده برای پیمایش درخت تراکنش‌ها استفاده می‌شود .

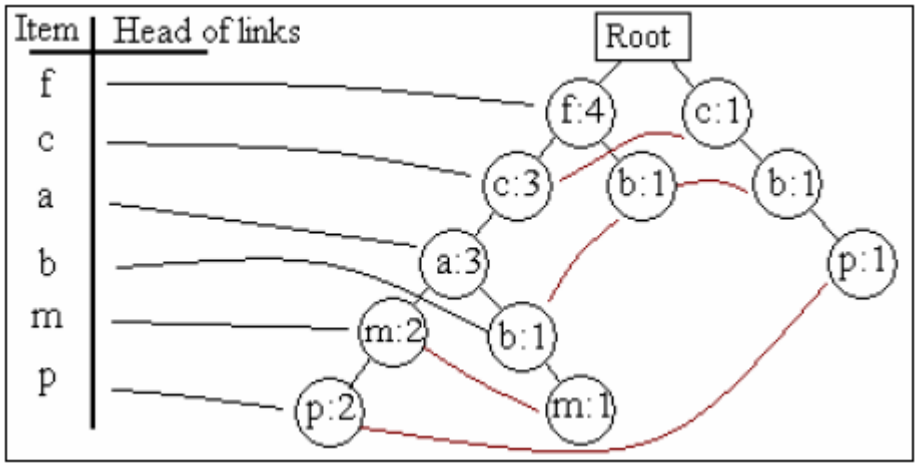
پس از یافتن عناصر تکرار شونده نوبت به ساخت درخت می‌رسد . ابتدا یک ریشه برای درخت به نام ROOT ساخته می‌شود و پایگاه داده را برای بار دوم پویش می‌کنیم تا تراکنش‌ها را استخراج کنیم . پویش اولین تراکنش که پس از مرتب‌سازی به صورت[(f:1),(c;1),(a:1),(m:1),(p:1)] است منجر به ساخت اولین شاخه درخت می‌شود (شکل زیر). توجه کنید که معیار مرتب سازی عناصر ترتیب وقوعشان در جدول سر آیند می‌باشد .

**درخت تراکنش های غربال شده با یک شاخه:**




به همین ترتیب بقیه تراکنش های موجود در پایگاه داده استخراج شده و به درخت اضافه می‌شوند و نهایتا درخت تراکنش‌ها به صورت شکل دوم ساخته می‌شود.

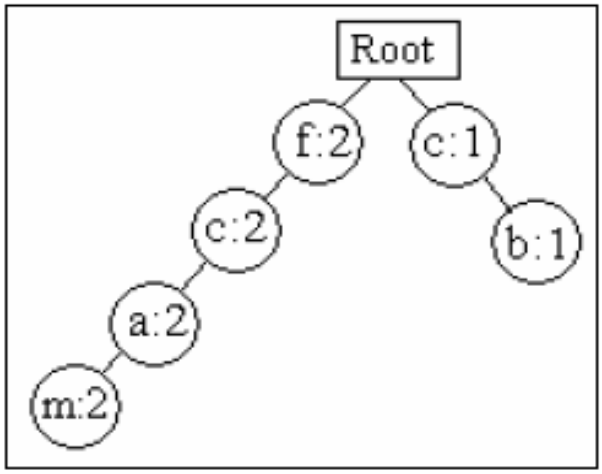
**جدول سرآیند و درخت تراکنش های غربال شده:**




مرحله‌ی بعدی کاوش درخت و یافتن الگوهای تکرار شونده با استفاده از جدول سر آیند می‌باشد . با انتخاب عنصر p به عنوان پایین‌ترین عنصر در جدول، کار یافتن الگوهای تکرار شونده‌ای که p هم جزو آن‌هاست آغاز می‌شود. بدین منظور با شروع اشاره‌گر p همه مسیرهایی که به p ختم می‌شوند را پیدا کنیم . برای گره p دو مسیر در درخت وجود دارد(f:4,c:3,a:3,m:2,p:2) و(c:1,b:1,p:1) شمارنده مربوط به هر عنصر پیشوندی برابر شمارنده عنصر مورد بررسی خواهد بود. برای مسیر (f:4,c:3,a:3,m:2,p:2) چون شمارنده P عدد دو می‌باشد، شمارنده همه عناصر رشته دو می‌شود و به طور مشابه در مسیر (c:1,b:1,p:1) چون شمارنده ی P عدد یک می باشد شمارنده‌ی همه عناصر رشته 1 می‌شود. لذا مجموعه مسیرهای پیشوندی حاصل به صورت {(c:1,b:1),(f:2,c:2,a:2,m:2)} خواهد بود.

حال می‌توان با داشتن این مجموعه از مسیرهای پیشوندی درخت تراکنش‌های مربوط به P را ساخت . پیش از شروع کاوش درخت تراکنش‌های غربال شده مربوط به P مجموع شمارنده‌های P که 3 می‌باشد را به عنوان میزان تکرارشوندگی P در نظر گرفته، P:3 را به عنوان اولین عنصر پیدا شده نگه می‌داریم . درخت ساخته شده مربوط به P به صورت شکل زیر خواهد بود.

درخت مربوط به عنصر P




این بار عمل یافتن الگوهای تکرار شونده را روی این درخت انجام می‌دهیم. به این منظور بایستی الگوریتم را روی همه عنصرهای این درخت اعمال کنیم و تمام مسیرهای ممکن را به صورت بازگشتی پیمایش کنیم.

اگر C را به عنوان نمونه در نظر بگیریم، دو مسیر ( F:2 ، C:2 ) و (C:1 ) از درخت انتخاب می‌شوند. با روی هم قرار دادن شمارنده‌های مربوط به C عدد سه (3) به دست می‌آید. همانند قبل c:3 را نگه داشته وکار را با مسیرهای پیشوندی C که در اینجا فقط (f:2) می‌باشد ادامه می‌دهیم. از آنجا که (F:2) تنها مسیر پیشوندی می‌باشد، احتیاج به ساخت درخت برای آن نیست و با توجه به اینکه میزان تکرار شوندگی آن کمتر از آستانه تکرار شوندگی یعنی 3 می‌باشد عملیات داده‌کاوی در این مسیر پایان یافته و با پیوند عناصر پیدا شده الگوی تکرار شونده (CP:2) ساخته می‌شود . برای یافتن بقیه عناصر تکرار شونده می‌توان به همین طریق عمل نمود.

# پیاده سازی الگوریتم Apriori در نرم افزار R

ما در این بخش از مجموعه داده market basket استفاده می‌کنیم. تجزیه و تحلیل سبد بازار یکی از تکنیک‌های کلیدی است که توسط خرده فروشان بزرگ برای کشف ارتباط بین اقلام مورد استفاده قرار می‌گیرد. با جستجوی ترکیبی از مواردی که اغلب در معاملات با هم اتفاق می‌افتند، کار می‌کند. به بیان دیگر، به خرده فروشان اجازه می‌دهد تا روابط بین اقلامی را که مردم می‌خرند شناسایی کنند.

این مجموعه داده شامل 38765 مشاهده می‌باشد . این مشاهدات شامل تاریخ، زمان و خریدهای انجام شده است.

ابتدا پکیج‌های مورد نیازمان را فراخوانی می‌کنیم.

```
if(sessionInfo()['basePkgs']=="dplyr" | sessionInfo()['otherPkgs']=="dplyr"){
 detach(package:dplyr, unload=TRUE)}
if(sessionInfo()['basePkgs']=="tm" | sessionInfo()['otherPkgs']=="tm"){
 detach(package:sentiment, unload=TRUE)
 detach(package:tm, unload=TRUE)}
library(plyr)
library(arules)
library(arulesViz)
```

حالا مجموعه داده را فراخوانی می‌کنیم.

```
groceries <- read.csv("C:/Users/acer/Desktop/Groceries_dataset.csv")
class(groceries)
[1] "data.frame"
```

در ادامه به منظور پاکسازی داده‌ها داریم:

```
str(groceries)
> str(groceries)
'data.frame':   38765 obs. of  3 variables:
 $ Member_number  : int  1808 2552 2300 1187 3037 4941 4501 3803 2762 4119 ...
 $ Date           : chr  "21-07-2015" "05-01-2015" "19-09-2015" "12-12-2015" ...
 $ itemDescription: chr  "tropical fruit" "whole milk" "pip fruit" "other vegetables" ..
```

```
head(groceries)
```

```
  Member_number       Date  itemDescription
1          1808 21-07-2015   tropical fruit
2          2552 05-01-2015       whole milk
3          2300 19-09-2015        pip fruit
4          1187 12-12-2015 other vegetables
5          3037 01-02-2015       whole milk
6          4941 14-02-2015       rolls/buns
```

```
sum(is.na(groceries))
```

```
> sum(is.na(groceries))
[1] 0
```

به منظور آماده‌سازی داده‌ها به تغییر شکل داده‌ها می‌پردازیم.

```
sorted <- groceries[order(groceries$Member_number),]

sorted$Member_number <- as.numeric(sorted$Member_number)

str(sorted)
```

```
> str(sorted)
'data.frame':   38765 obs. of  3 variables:
 $ Member_number  : num  1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...
 $ Date           : chr  "27-05-2015" "24-07-2015" "15-03-2015" "25-11-2015" .
 $ itemDescription: chr  "soda" "canned beer" "sausage" "sausage" ...
```

اکنون به گروه‌بندی همه اقلامی که توسط یک مشتری در یک تاریخ با هم خریداری شده اند، می‌پردازیم.

```
itemList <- ddply(sorted,
c("Member_number","Date"),function(df1)paste(df1$itemDescription,collapse = ","))

head(itemList,15)
```

```
   Member_number       Date                                              V1
1           1000 15-03-2015 sausage,whole milk,semi-finished bread,yogurt
2           1000 24-06-2014                 whole milk,pastry,salty snack
3           1000 24-07-2015                   canned beer,misc. beverages
4           1000 25-11-2015                      sausage,hygiene articles
5           1000 27-05-2015                       soda,pickled vegetables
6           1001 02-05-2015                               frankfurter,curd
7           1001 07-02-2014                 sausage,whole milk,rolls/buns
8           1001 12-12-2014                                whole milk,soda
9           1001 14-04-2015                               beef,white bread
10          1001 20-01-2015          frankfurter,soda,whipped/sour cream
11          1002 09-02-2014           frozen vegetables,other vegetables
12          1002 26-04-2014                             butter,whole milk
13          1002 26-04-2015                        tropical fruit,sugar
14          1002 30-08-2015              butter milk,specialty chocolate
15          1003 10-02-2015                               sausage,rolls/buns
```

شماره و تاریخ را حذف می‌کنیم.

```
itemList$Member_number <- NULL

itemList$Date <- NULL

colnames(itemList) <- c("itemList")
```

```
                                      itemList
1  sausage,whole milk,semi-finished bread,yogurt
2                  whole milk,pastry,salty snack
3                    canned beer,misc. beverages
4                       sausage,hygiene articles
5                        soda,pickled vegetables
6                                frankfurter,curd
7                  sausage,whole milk,rolls/buns
8                                 whole milk,soda
9                               beef,white bread
10           frankfurter,soda,whipped/sour cream
11            frozen vegetables,other vegetables
12                              butter,whole milk
13                         tropical fruit,sugar
14               butter milk,specialty chocolate
15                             sausage,rolls/buns
```

```
write.csv(itemList,"ItemList.csv", quote = FALSE, row.names = TRUE)

head(itemList)
```

```
                                      itemList
1 sausage,whole milk,semi-finished bread,yogurt
2                  whole milk,pastry,salty snack
3                    canned beer,misc. beverages
4                       sausage,hygiene articles
5                        soda,pickled vegetables
6                                frankfurter,curd
```

فامیل csv را به فرمت سبد خرید تبدیل می‌کنیم.

```
txn = read.transactions(file="ItemList.csv", rm.duplicates= TRUE,
format="basket",sep=",",cols=1);

print(txn)
```

```
transactions in sparse format with
 14964 transactions (rows) and
 168 items (columns)
```

در مجموع 14964 تراکنش با 168 محصول متمایز وجود دارد.

حال الگوریتم Apriori را بر روی مجموعه داده مورد نظرمان پیاده سازی می‌کنیم. بدین منظور داریم:

```
basket_rules <- apriori(txn, parameter = list(minlen=2, sup = 0.001, conf = 0.05, target="rules"))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
       0.05    0.1    1 none FALSE            TRUE       5   0.001      2     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 14

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[168 item(s), 14964 transaction(s)] done [0.01s].
sorting and recoding items ... [149 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [450 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

مجموع قوانین ایجاد شده

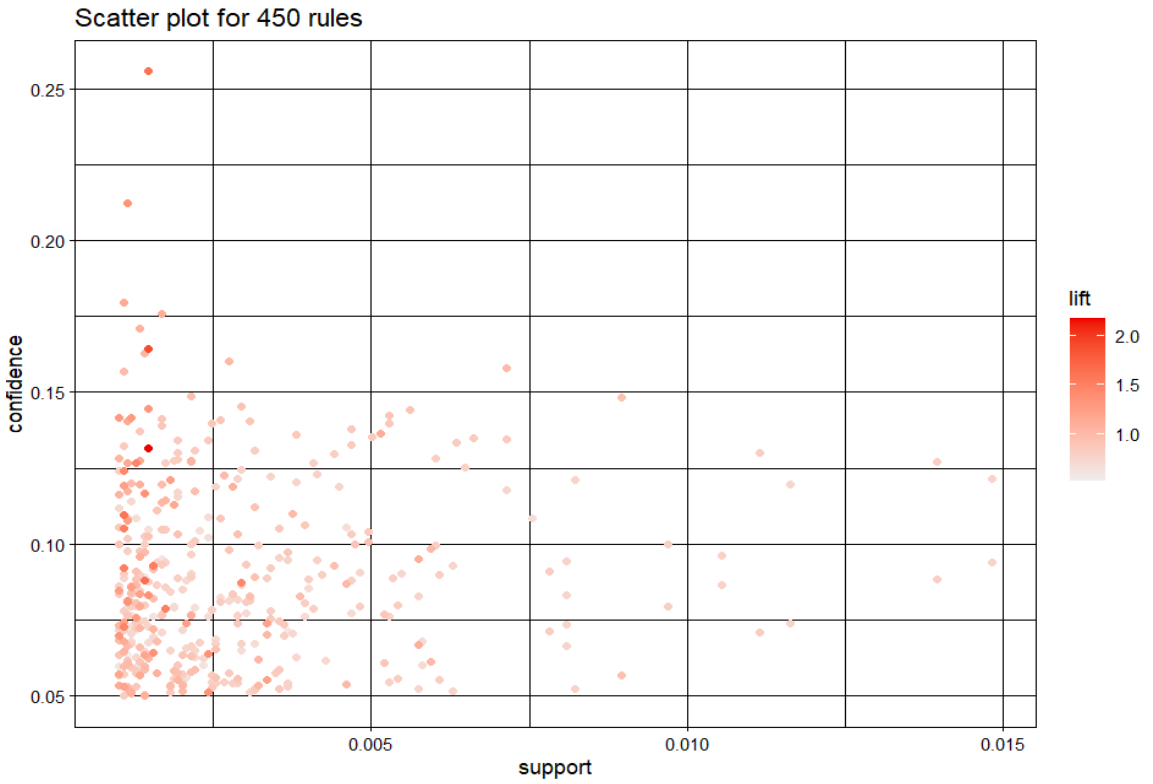
```
print(length(basket_rules))

[1] 450

summary(basket_rules)

set of 450 rules

rule length distribution (lhs + rhs):sizes
  2   3
423  27

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.00    2.00    2.00    2.06    2.00    3.00

summary of quality measures:
    support           confidence        coverage             lift           count
 Min.   :0.001002   Min.   :0.05000   Min.   :0.005346   Min.   :0.5195   Min.   : 15.0
 1st Qu.:0.001270   1st Qu.:0.06397   1st Qu.:0.015972   1st Qu.:0.7673   1st Qu.: 19.0
 Median :0.001938   Median :0.08108   Median :0.023590   Median :0.8350   Median : 29.0
 Mean   :0.002760   Mean   :0.08759   Mean   :0.033723   Mean   :0.8859   Mean   : 41.3
 3rd Qu.:0.003341   3rd Qu.:0.10482   3rd Qu.:0.043705   3rd Qu.:0.9601   3rd Qu.: 50.0
 Max.   :0.014836   Max.   :0.25581   Max.   :0.157912   Max.   :2.1831   Max.   :222.0

mining info:
 data ntransactions support confidence
  txn         14964   0.001       0.05
                                                                                      call
 apriori(data = txn, parameter = list(minlen = 2, sup = 0.001, conf = 0.05, target = "rules"))
```

inspect(basket_rules[1:20])

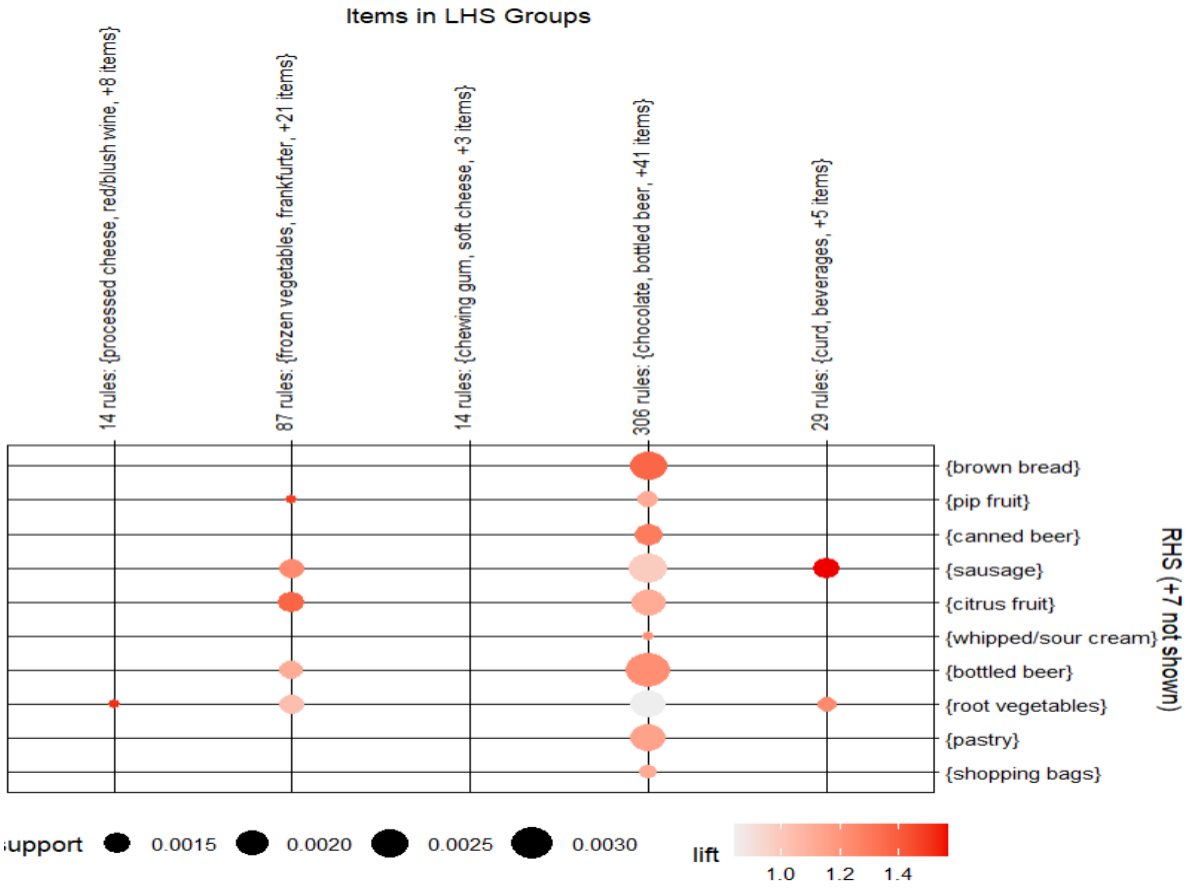
```
> inspect(basket_rules[1:20])
     lhs                              rhs                support     confidence coverage    lift
[1]  {frozen fish}                 => {whole milk}       0.001069233 0.1568627  0.006816359 0.9933534
[2]  {seasonal products}           => {rolls/buns}       0.001002406 0.1415094  0.007083667 1.2864807
[3]  {pot plants}                  => {other vegetables} 0.001002406 0.1282051  0.007818765 1.0500611
[4]  {pot plants}                  => {whole milk}       0.001002406 0.1282051  0.007818765 0.8118754
[5]  {pasta}                       => {whole milk}       0.001069233 0.1322314  0.008086073 0.8373723
[6]  {pickled vegetables}          => {whole milk}       0.001002406 0.1119403  0.008954825 0.7088763
[7]  {packaged fruit/vegetables}   => {rolls/buns}       0.001202887 0.1417323  0.008487036 1.2885066
[8]  {detergent}                   => {yogurt}           0.001069233 0.1240310  0.008620690 1.4443580
[9]  {detergent}                   => {rolls/buns}       0.001002406 0.1162791  0.008620690 1.0571081
[10] {detergent}                   => {whole milk}       0.001403368 0.1627907  0.008620690 1.0308929
[11] {semi-finished bread}         => {other vegetables} 0.001002406 0.1056338  0.009489441 0.8651911
[12] {semi-finished bread}         => {whole milk}       0.001670676 0.1760563  0.009489441 1.1148993
[13] {red/blush wine}              => {rolls/buns}       0.001336541 0.1273885  0.010491847 1.1581057
[14] {red/blush wine}              => {other vegetables} 0.001136060 0.1082803  0.010491847 0.8868668
[15] {flour}                       => {tropical fruit}   0.001069233 0.1095890  0.009756750 1.6172489
[16] {flour}                       => {whole milk}       0.001336541 0.1369863  0.009756750 0.8674833
[17] {herbs}                       => {yogurt}           0.001136060 0.1075949  0.010558674 1.2529577
[18] {herbs}                       => {whole milk}       0.001136060 0.1075949  0.010558674 0.6813587
[19] {processed cheese}            => {root vegetables}  0.001069233 0.1052632  0.010157712 1.5131200
[20] {processed cheese}            => {rolls/buns}       0.001470195 0.1447368  0.010157712 1.3158214

     count
[1]  16
[2]  15
[3]  15
[4]  15
[5]  16
[6]  15
[7]  18
[8]  16
[9]  15
[10] 21
[11] 15
[12] 25
[13] 20
[14] 17
[15] 16
[16] 20
[17] 17
[18] 17
[19] 16
[20] 22
```

برای درک بهتر به صورت مصورسازی داریم:

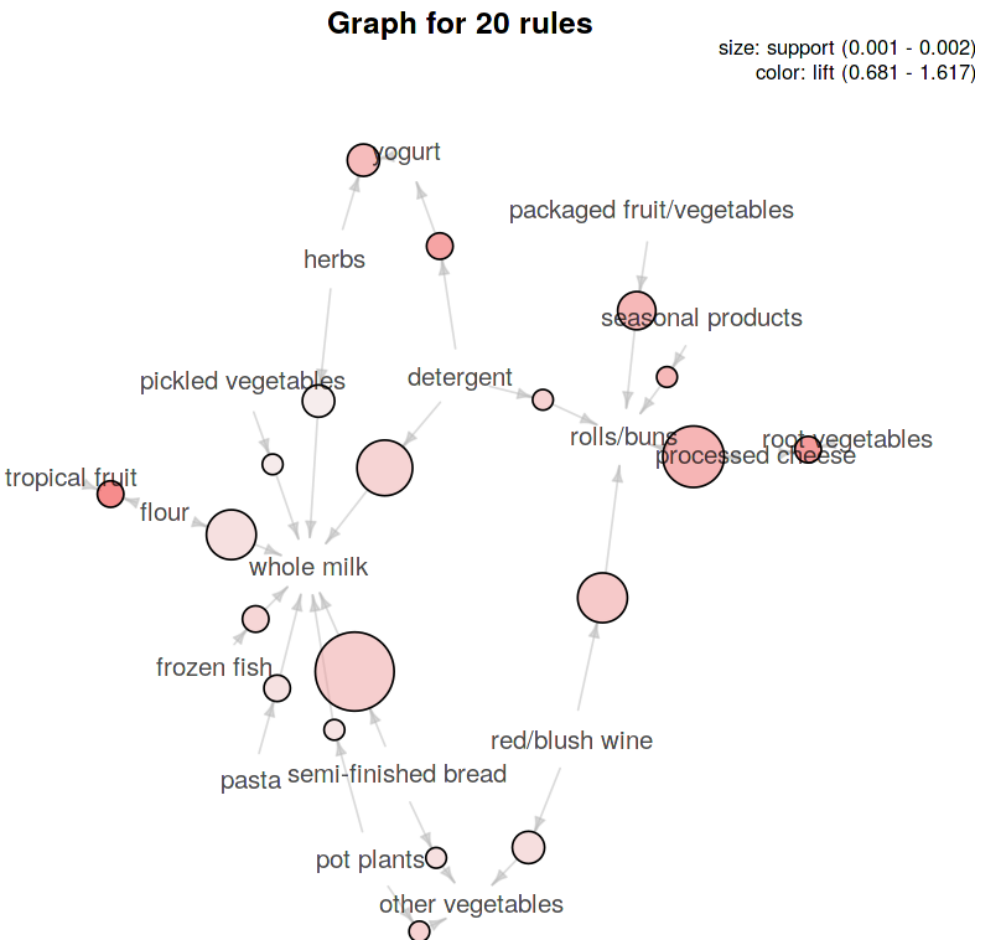
```
plot(basket_rules, jitter = 0)
```




```
plot(basket_rules, method = "grouped", control = list(k = 5))
```

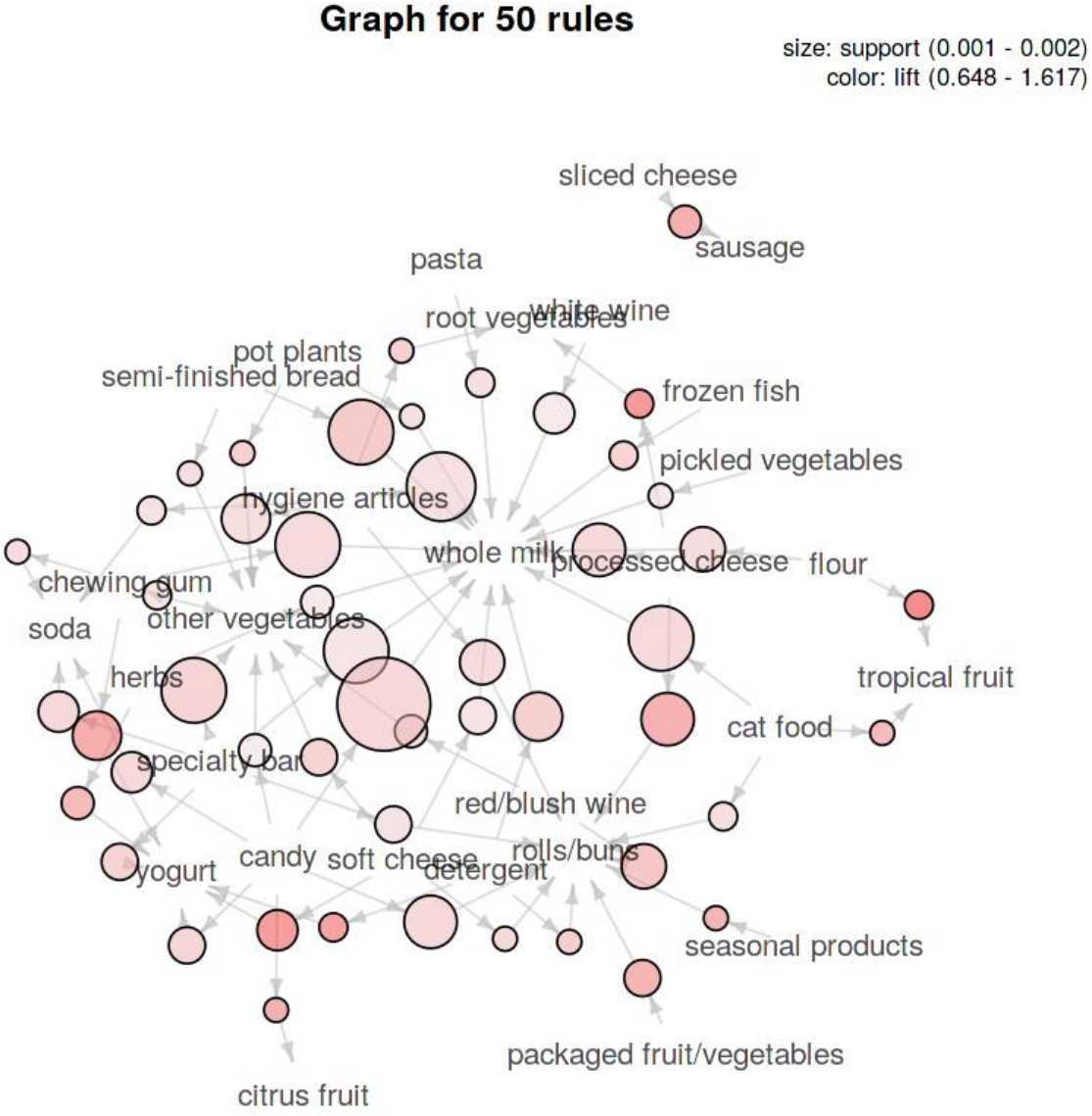



```
plot(basket_rules[1:20], method="graph")
```



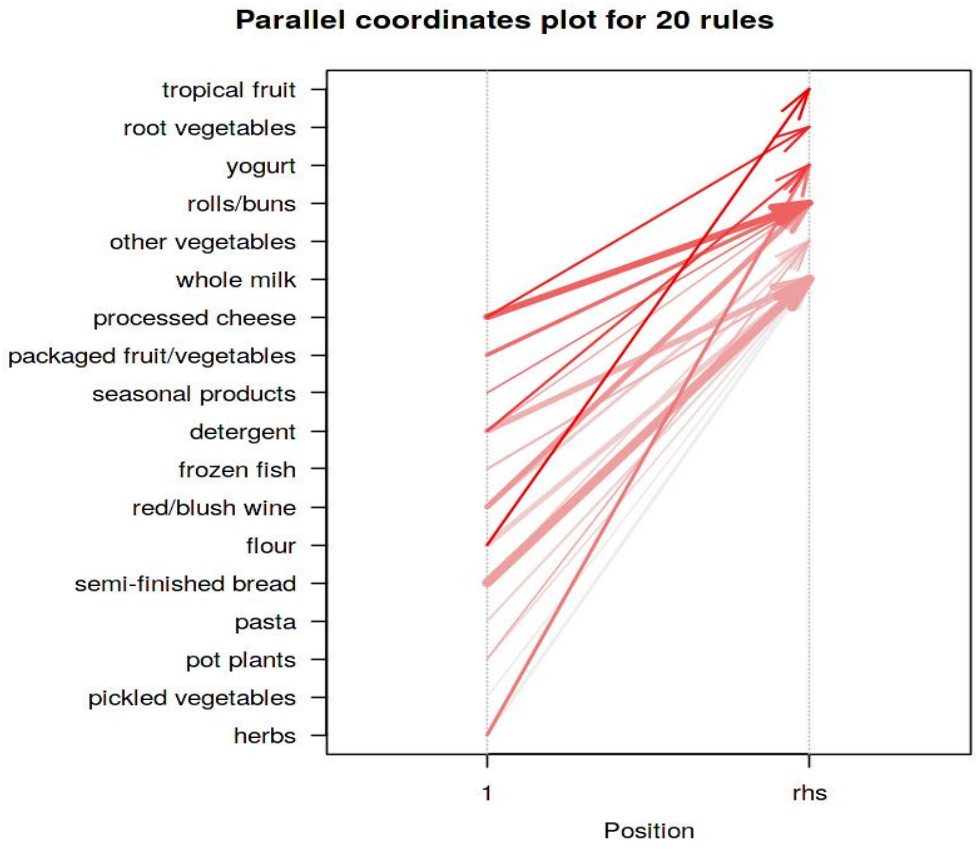

نمودار 50 قانون اول

```
plot(basket_rules[1:50], method="graph")
```



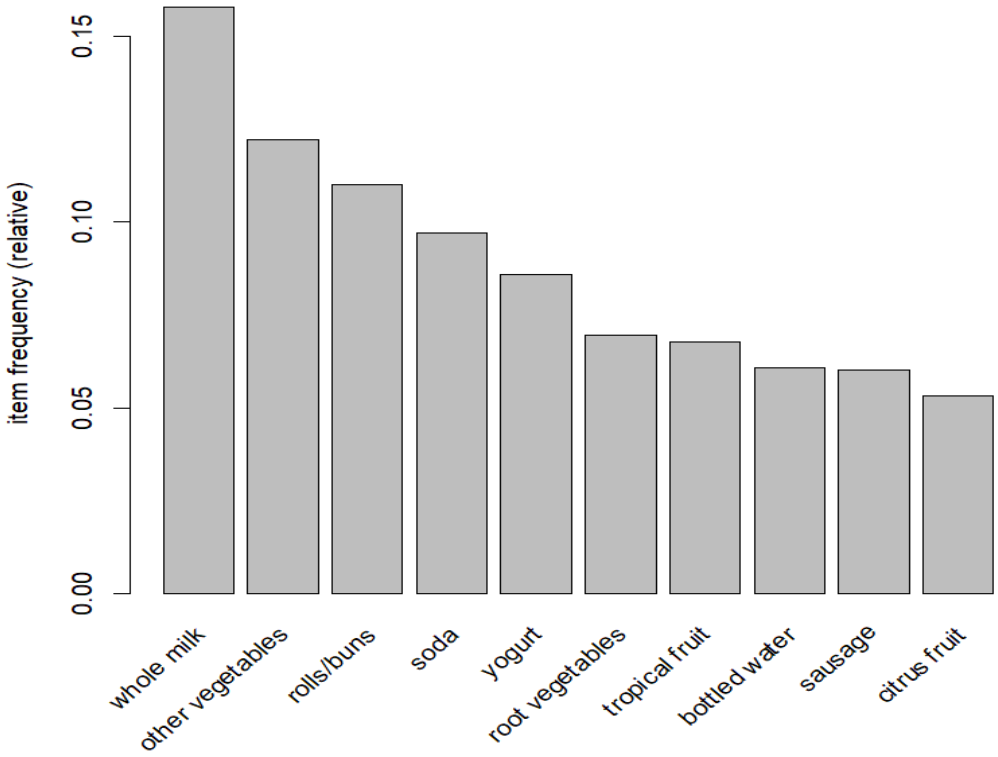

نمودار مختصات موازی

```
plot(basket_rules[1:20], method="paracoord")
```




متداول ترین محصولات

```
itemFrequencyPlot(txn, topN = 10)
```



تغییر هایپرپارامترها

```
basket_rules2 <- apriori(txn, parameter = list(minlen=3, sup = 0.001, conf = 0.1, target="rules"))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.1    0.1    1 none FALSE            TRUE       5   0.001      3     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 14

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[168 item(s), 14964 transaction(s)] done [0.01s].
sorting and recoding items ... [149 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [17 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

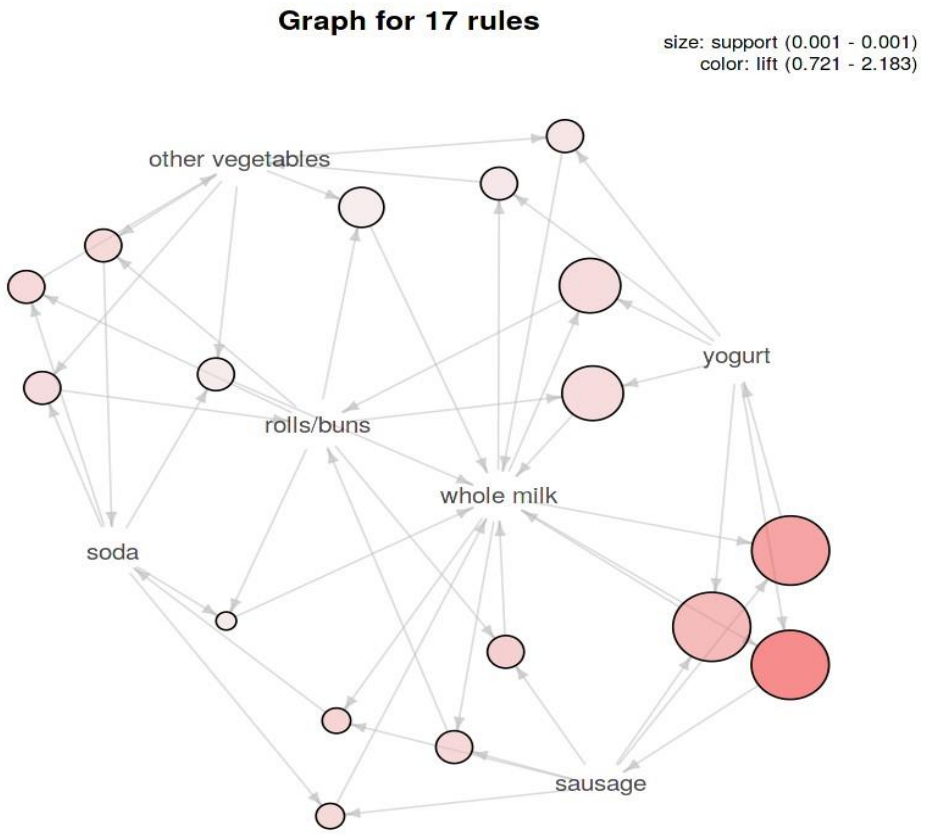
```
print(length(basket_rules2))

summary(basket_rules2)
```

```
[1] 17
> summary(basket_rules2)
set of 17 rules

rule length distribution (lhs + rhs):sizes
 3
17

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      3       3       3       3       3       3

summary of quality measures:
    support           confidence        coverage             lift            count
 Min.   :0.001002   Min.   :0.1018   Min.   :0.005346   Min.   :0.7214   Min.   :15.00
 1st Qu.:0.001136   1st Qu.:0.1172   1st Qu.:0.008086   1st Qu.:0.8897   1st Qu.:17.00
 Median :0.001136   Median :0.1269   Median :0.008955   Median :1.1081   Median :17.00
 Mean   :0.001207   Mean   :0.1437   Mean   :0.008821   Mean   :1.1794   Mean   :18.06
 3rd Qu.:0.001337   3rd Qu.:0.1642   3rd Qu.:0.010559   3rd Qu.:1.2297   3rd Qu.:20.00
 Max.   :0.001470   Max.   :0.2558   Max.   :0.011160   Max.   :2.1831   Max.   :22.00

mining info:
 data ntransactions support confidence
  txn         14964   0.001        0.1
                                                                                    call
 apriori(data = txn, parameter = list(minlen = 3, sup = 0.001, conf = 0.1, target = "rules"))
```

```
inspect(basket_rules2)
```

```
> inspect(basket_rules2)
     lhs                                rhs                support     confidence coverage    lift
[1]  {sausage, yogurt}               => {whole milk}       0.001470195 0.2558140  0.005747126 1.6199746
[2]  {sausage, whole milk}           => {yogurt}           0.001470195 0.1641791  0.008954825 1.9118880
[3]  {whole milk, yogurt}            => {sausage}          0.001470195 0.1317365  0.011160118 2.1830624
[4]  {sausage, soda}                 => {whole milk}       0.001069233 0.1797753  0.005947608 1.1384500
[5]  {sausage, whole milk}           => {soda}             0.001069233 0.1194030  0.008954825 1.2296946
[6]  {rolls/buns, sausage}           => {whole milk}       0.001136060 0.2125000  0.005346164 1.3456835
[7]  {sausage, whole milk}           => {rolls/buns}       0.001136060 0.1268657  0.008954825 1.1533523
[8]  {rolls/buns, yogurt}            => {whole milk}       0.001336541 0.1709402  0.007818765 1.0825005
[9]  {whole milk, yogurt}            => {rolls/buns}       0.001336541 0.1197605  0.011160118 1.0887581
[10] {other vegetables, yogurt}      => {whole milk}       0.001136060 0.1404959  0.008086073 0.8897081
[11] {whole milk, yogurt}            => {other vegetables} 0.001136060 0.1017964  0.011160118 0.8337610
[12] {rolls/buns, soda}              => {other vegetables} 0.001136060 0.1404959  0.008086073 1.1507281
[13] {other vegetables, soda}        => {rolls/buns}       0.001136060 0.1172414  0.009689922 1.0658566
[14] {other vegetables, rolls/buns}  => {soda}             0.001136060 0.1075949  0.010558674 1.1080872
[15] {rolls/buns, soda}              => {whole milk}       0.001002406 0.1239669  0.008086073 0.7850365
[16] {other vegetables, soda}        => {whole milk}       0.001136060 0.1172414  0.009689922 0.7424460
[17] {other vegetables, rolls/buns}  => {whole milk}       0.001202887 0.1139241  0.010558674 0.7214386
     count
[1]  22
[2]  22
[3]  22
[4]  16
[5]  16
[6]  17
[7]  17
[8]  20
[9]  20
[10] 17
[11] 17
[12] 17
[13] 17
[14] 17
[15] 15
[16] 17
```

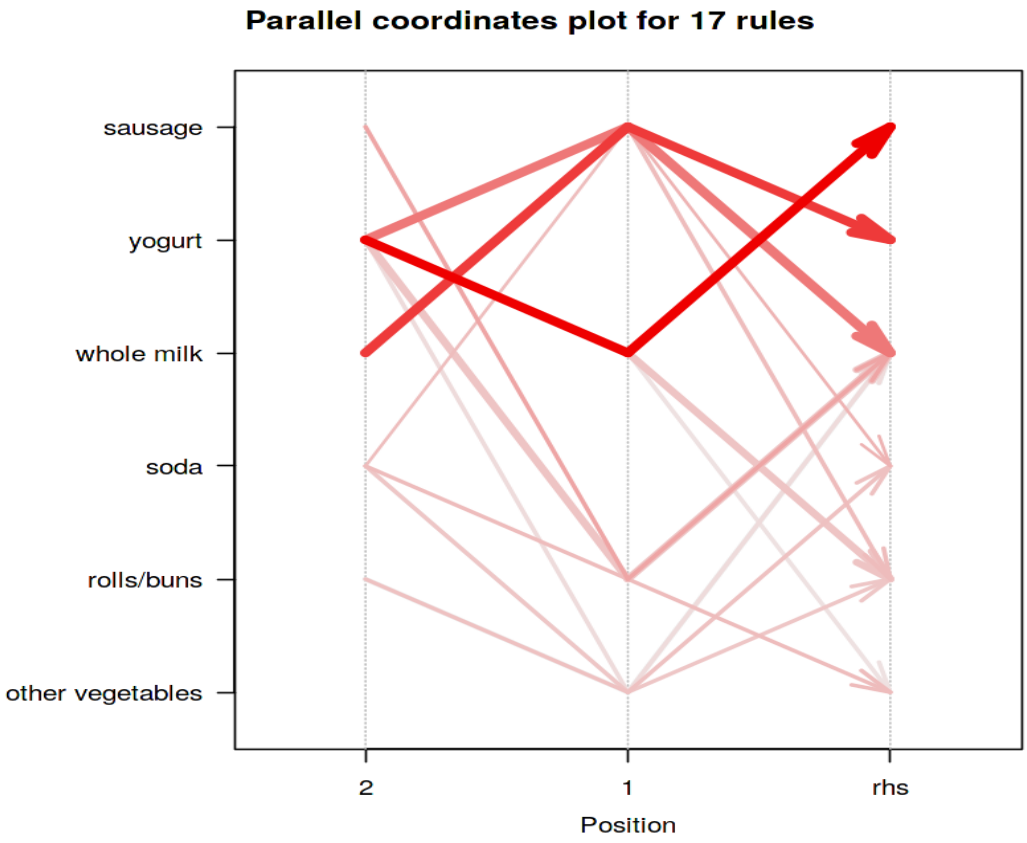
```
plot(basket_rules2, method="graph")
```



شکل فوق ارتباط میان متغیرها در قوانین کشف شده نشان می‌دهد.

```
plot(basket_rules2, method="paracoord")
```

عمق رنگ قرمز در نقاط رسم شده میزان lift را نشان می‌دهد و ابعاد هر دایره نشان‌دهنده میزان confidence می‌باشد.



## نتیجه‌گیری:

در این پروژه به بررسی قوانین انجمنی و دو روش رایج و کارای آن **Apriori** و **FP-Growth** پرداختیم.

نتایج حاصل از این مدل می‌تواند در تنظیم مناسب جایگاه مواد غذایی استفاده نمود.

در نتیجه، تجزیه و تحلیل مجموعه داده Groceries با استفاده از الگوریتم Apriori و قوانین ارتباط، بینش‌های ارزشمندی را در مورد روابط و الگوهای بین اقلام در مجموعه داده ارائه کرده است. با تنظیم آستانه اطمینان و پشتیبانی مناسب، ما توانستیم تعداد زیادی از قوانین مرتبط را ایجاد کنیم که مجموعه آیتم‌ها و جفت‌های مکرر مجموعه اقلامی را که همزمان در تراکنش‌ها اتفاق می‌افتند، برجسته می‌کنند.

همچنین درک کردیم که متداول‌ترین محصولات در سبد فروشگاه‌ها شیر پرچرب و سبزیجات می‌باشد.

تجسم‌ها، از جمله نمودارهای فرکانس آیتم، نمودارهای پراکنده، و نمودارهای شبکه، به ما این امکان را می‌دهند که روابط و وابستگی‌های بین آیتم‌ها، خوشه‌های آیتم‌های مرتبط و قوانین موجود در مجموعه داده را بیشتر کشف و درک کنیم.

پیوست

کدهای R

```
if(sessionInfo()['basePkgs']=="dplyr" | sessionInfo()['otherPkgs']=="dplyr"){
  detach(package:dplyr, unload=TRUE)
}

if(sessionInfo()['basePkgs']=="tm" | sessionInfo()['otherPkgs']=="tm"){
  detach(package:sentiment, unload=TRUE)
  detach(package:tm, unload=TRUE)
}
groceries <- read.csv("C:/Users/acer/Desktop/Groceries_dataset.csv")
class(groceries)
str(groceries)
head(groceries)
sum(is.na(groceries))
sorted <- groceries[order(groceries$Member_number),]
sorted$Member_number <- as.numeric(sorted$Member_number)
str(sorted)

itemList <- ddply(sorted, c("Member_number","Date"),
function(df1)paste(df1$itemDescription,collapse = ","))

head(itemList,15)
itemList$Member_number <- NULL
itemList$Date <- NULL
colnames(itemList) <- c("itemList")
```

```
write.csv(itemList,"ItemList.csv", quote = FALSE, row.names = TRUE)

head(itemList)

txn = read.transactions(file="ItemList.csv", rm.duplicates= TRUE,
format="basket",sep=",",cols=1);

print(txn)

txn@itemInfo$labels <- gsub("\"","",txn@itemInfo$labels)

basket_rules <- apriori(txn, parameter = list(minlen=2, sup = 0.001, conf = 0.05, target="rules"))

print(length(basket_rules))

summary(basket_rules)

inspect(basket_rules[1:20])

plot(basket_rules, jitter = 0)

plot(basket_rules, method = "grouped", control = list(k = 5))

plot(basket_rules[1:20], method="graph")

plot(basket_rules[1:50], method="graph")

plot(basket_rules[1:20], method="paracoord")

itemFrequencyPlot(txn, topN = 10)

basket_rules2 <- apriori(txn, parameter = list(minlen=3, sup = 0.001, conf = 0.1, target="rules"))

print(length(basket_rules2))

summary(basket_rules2)

inspect(basket_rules2)

plot(basket_rules2, method="graph")

plot(basket_rules2, method="paracoord")
```