DATA INSIGHTS

Manipulating Data and Unsupervised Methods with R

John Chandler

Founder, Data Insights

Clinical Professor of Marketing, U of MT

Agenda

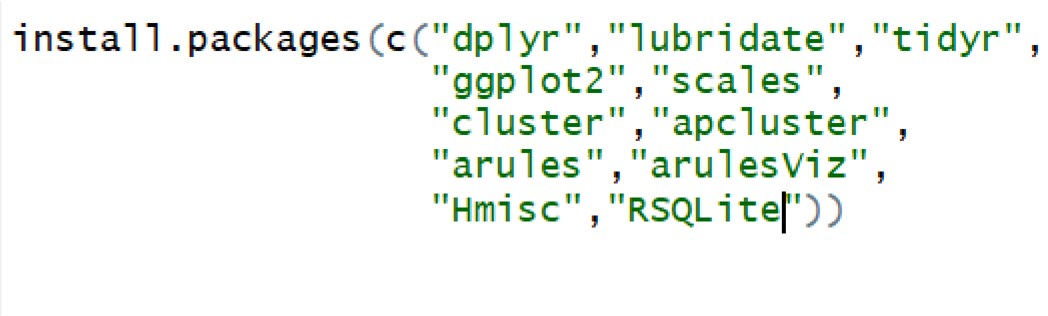
- Introductions
- Data access and manipulation in R
- Segmentation
- Association Analysis
- Wrapping-up

Data and Code Access

1. Make a pull request or download code and data from here:

   https://github.com/37chandler/BigDataTech2017

2. Alternatively, files available on a USB.
3. Open up R studio and install the packages we need:



```
install.packages(c("dplyr","lubridate","tidyr",
                   "ggplot2","scales",
                   "cluster","apcluster",
                   "arules","arulesViz",
                   "Hmisc","RSQLite"))
```

4. Follow along during lecture and be ready for hands-on work.

Introductions

My background

- 1999: Finishing grad school round 1 (MS in Math at UW), started at Avenue A

avenue a™

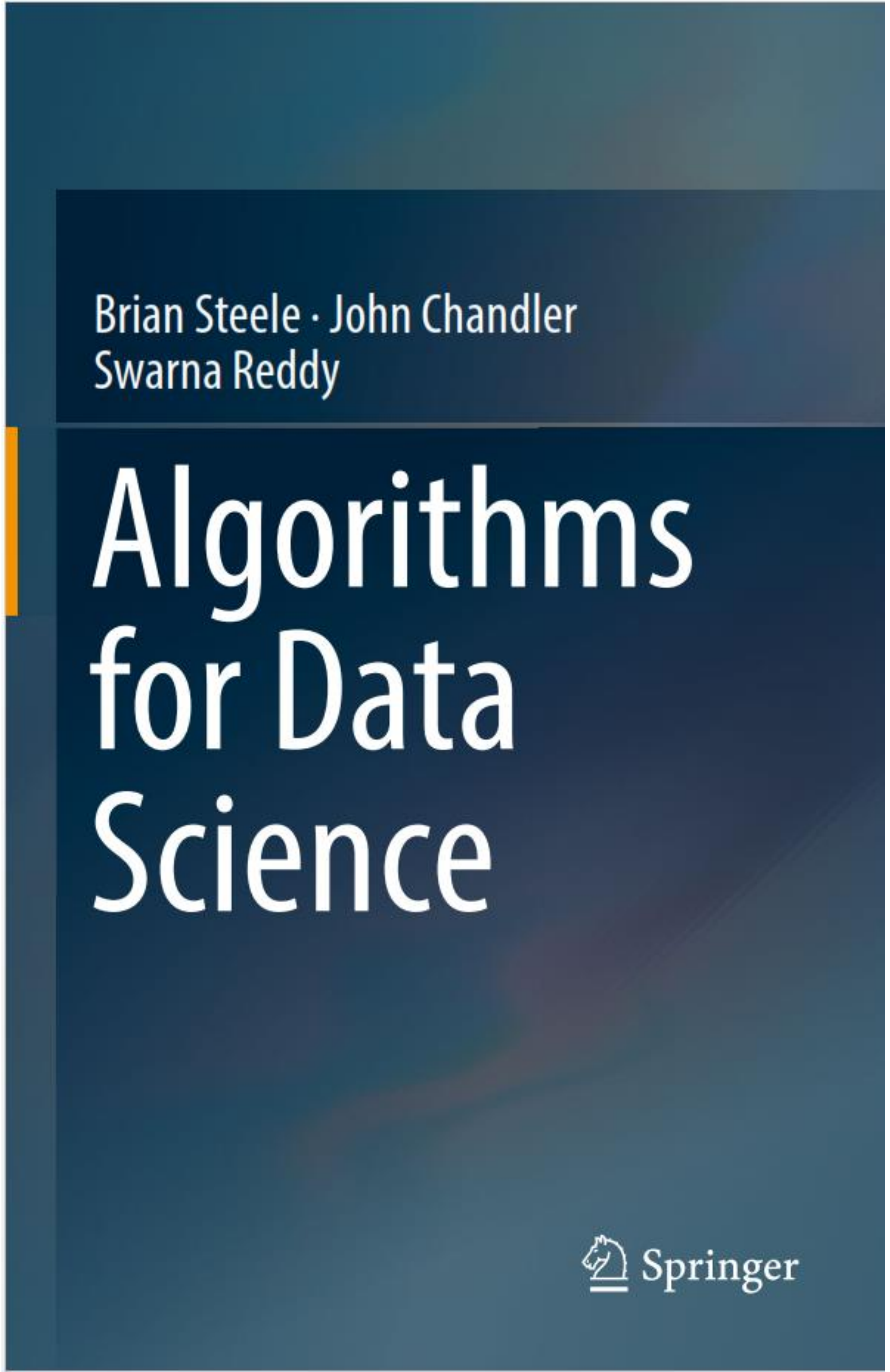
Know what works.

- 2000: Parent company formed, division split, IPO, Dot Com meltdown.

aQuantive = atlas DMT™ + avenue a™ Know what works.

My background

- 2003: Got harangued about tuition reimbursement, started PhD.
- 2004: Spent two years working on Media Network.
- 2006: Worked on video business for 2 years.
- 2007: aQuantive acquired by Microsoft, part of MS Advertising.
- 2010: Finished Ph.D. in Statistics.
- 2011: Left Microsoft to start consulting business.
- 2013: Joined faculty at University of Montana School of Business.



Brian Steele · John Chandler
Swarna Reddy

Algorithms for Data Science

Springer

My Technology Choices

Data scientists need solutions for these tasks:

- Data cleaning and munging
- Statistical modeling
- Fast access to moderate data sets
- Ability to work with distributed data sets

My Technology Choices

Data scientists need solutions for these tasks:

- Data cleaning and munging: Python
- Statistical modeling: R
- Fast access to moderate data sets: SQL
- Ability to work with distributed data sets: Hive/Spark

Why R today?

Advantages of R

- Most powerful statistical programming language
- Excellent open-source community
- Great data visualization
- Low commercial barriers to entry
- World-leading data manipulation abilities

Today’s Data Set

- Transaction-level information from the country’s largest co-op grocery store.
- Members in the co-op are called “Owners”, stored in field called `card_no`.
- Two data sources:
  - 732K records across 82 months for 125 owners in `OwnerTransactions_62.txt`.
  - 3.3M records across 82 months for 1360 owners in a SQLite database called `20170530_transaction_files.db`.
- There’s a bunch of “tribal knowledge” of the data that you’ll see applied in our queries.
  - One key one: to summarize sales use the `total` field where `trans_type==I` and the `0 < total < 100`

Goals:

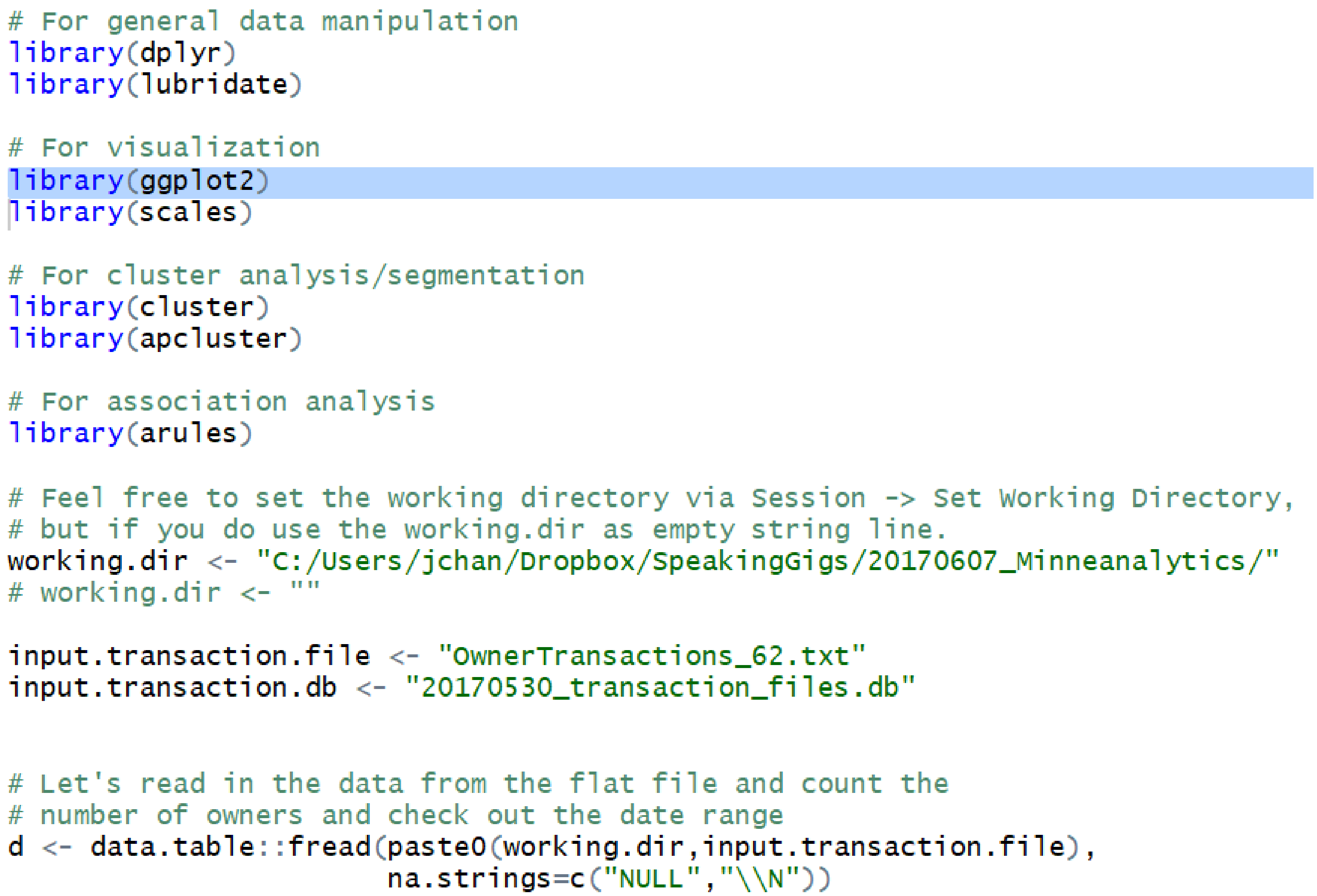
1. Segment these owners.
2. Uncover relationship between a segment and items purchased.

Data Access and Manipulation

Loading and storing data

- R typically only works in memory, so we must be judicious on data set sizes.
- Best way to read in flat files is with the `data.table` function `fread` (for “fast read”).
- Larger datasets that fit on a single machine can be easily queried out of SQL.

Loading libraries, reading data…



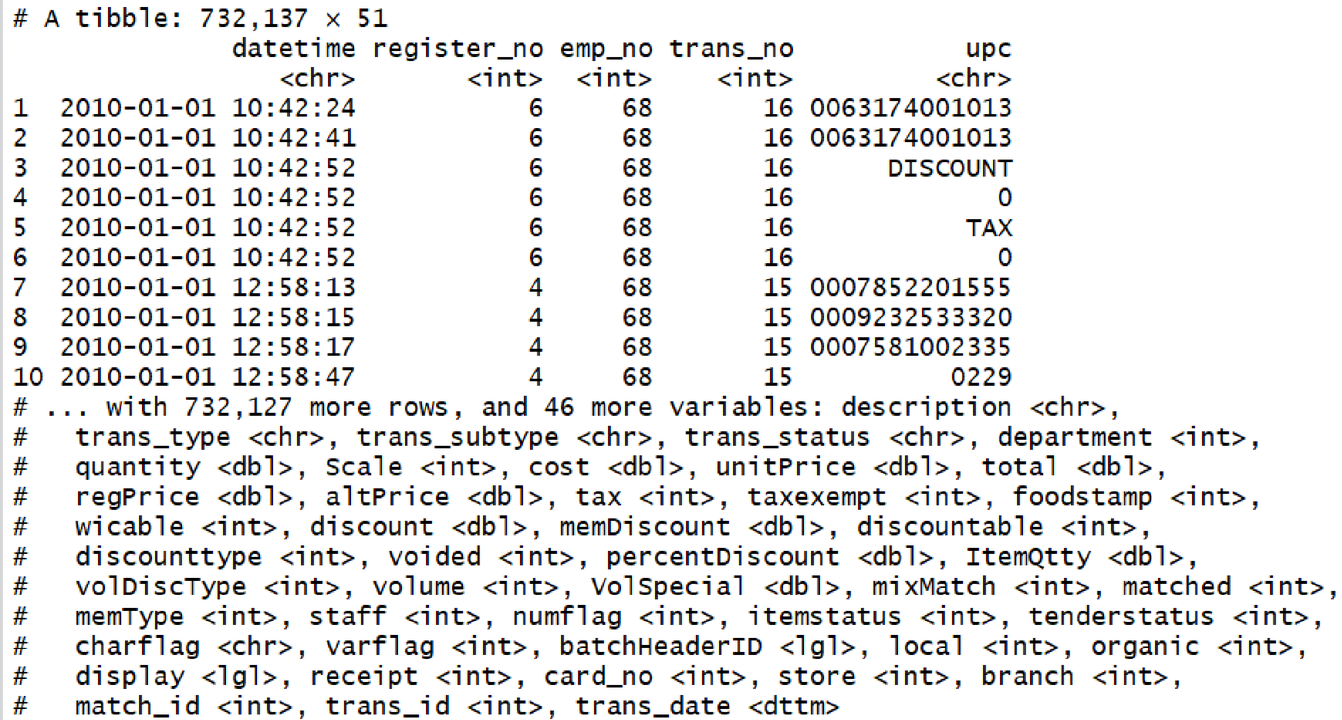
```
# For general data manipulation
library(dplyr)
library(lubridate)

# For visualization
library(ggplot2)
library(scales)

# For cluster analysis/segmentation
library(cluster)
library(apcluster)

# For association analysis
library(arules)

# Feel free to set the working directory via Session -> Set Working Directory,
# but if you do use the working.dir as empty string line.
working.dir <- "C:/Users/jchan/Dropbox/SpeakingGigs/20170607_Minneanalytics/"
# working.dir <- ""

input.transaction.file <- "OwnerTransactions_62.txt"
input.transaction.db <- "20170530_transaction_files.db"


# Let's read in the data from the flat file and count the
# number of owners and check out the date range
d <- data.table::fread(paste0(working.dir,input.transaction.file),
                       na.strings=c("NULL","\\N"))
```

`dplyr` Package in R

- Created by Hadley Wickham and Roman Francois (part of the “Hadleyverse” which includes `ggplot`)
- A grammar for the manipulation of data sets.

Tables in dplyr

- dplyr introduces the concept of a “tibble” (formerly tbl_df, now in its own package) that allows for
  - Better printing to the screen
  - Faster subsetting



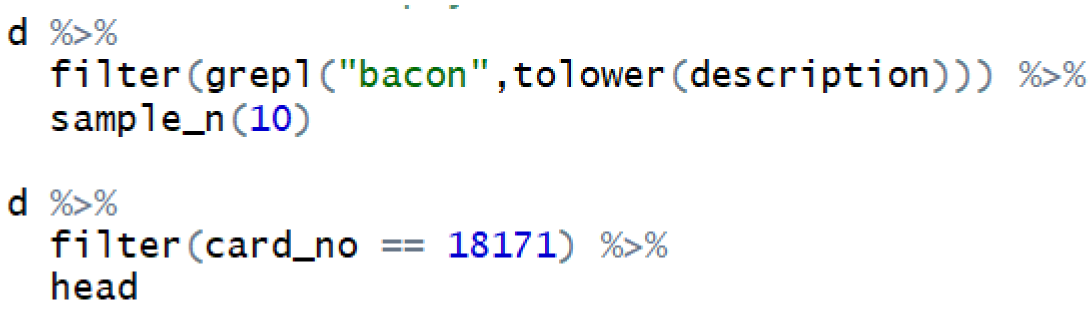
```
# A tibble: 732,137 × 51
              datetime register_no emp_no trans_no           upc
                 <chr>       <int>  <int>    <int>         <chr>
1  2010-01-01 10:42:24           6     68       16 0063174001013
2  2010-01-01 10:42:41           6     68       16 0063174001013
3  2010-01-01 10:42:52           6     68       16      DISCOUNT
4  2010-01-01 10:42:52           6     68       16             0
5  2010-01-01 10:42:52           6     68       16           TAX
6  2010-01-01 10:42:52           6     68       16             0
7  2010-01-01 12:58:13           4     68       15 0007852201555
8  2010-01-01 12:58:15           4     68       15 0009232533320
9  2010-01-01 12:58:17           4     68       15 0007581002335
10 2010-01-01 12:58:47           4     68       15          0229
# ... with 732,127 more rows, and 46 more variables: description <chr>,
#   trans_type <chr>, trans_subtype <chr>, trans_status <chr>, department <int>,
#   quantity <dbl>, Scale <int>, cost <dbl>, unitPrice <dbl>, total <dbl>,
#   regPrice <dbl>, altPrice <dbl>, tax <int>, taxexempt <int>, foodstamp <int>,
#   wicable <int>, discount <dbl>, memDiscount <dbl>, discountable <int>,
#   discounttype <int>, voided <int>, percentDiscount <dbl>, ItemQtty <dbl>,
#   volDiscType <int>, volume <int>, VolSpecial <dbl>, mixMatch <int>, matched <int>,
#   memType <int>, staff <int>, numflag <int>, itemstatus <int>, tenderstatus <int>,
#   charflag <chr>, varflag <int>, batchHeaderID <lgl>, local <int>, organic <int>,
#   display <lgl>, receipt <int>, card_no <int>, store <int>, branch <int>,
#   match_id <int>, trans_id <int>, trans_date <dttm>
```

The Verbs of dplyr

- `filter`()
- `arrange`()
- `select`()
- `mutate`()
- `summarize`(); `group_by`(); `ungroup()`
- `sample_n`(); `sample_frac`()

filter

- Used to filter rows of a data frame.

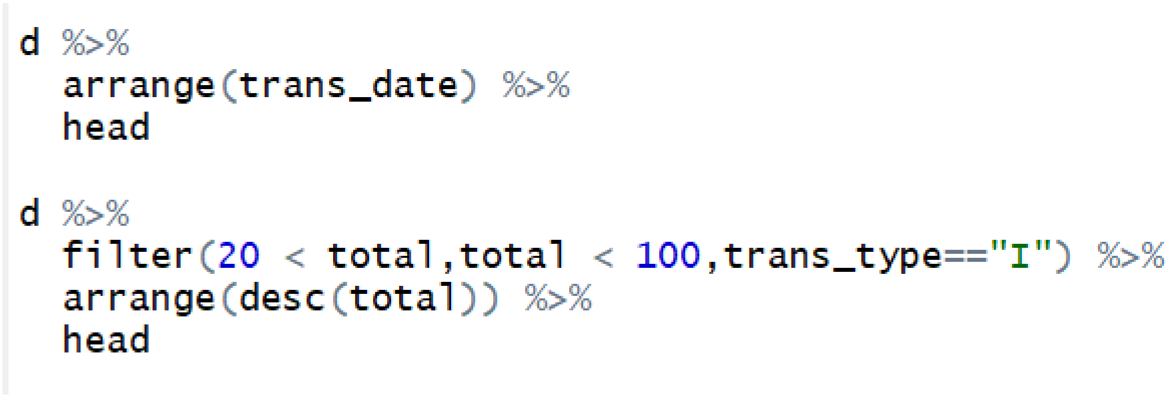


```
d %>%
  filter(grepl("bacon",tolower(description))) %>%
  sample_n(10)

d %>%
  filter(card_no == 18171) %>%
  head
```

- Similar to subset in R, but much, much faster.

arrange

- Used to sort rows of a data frame.

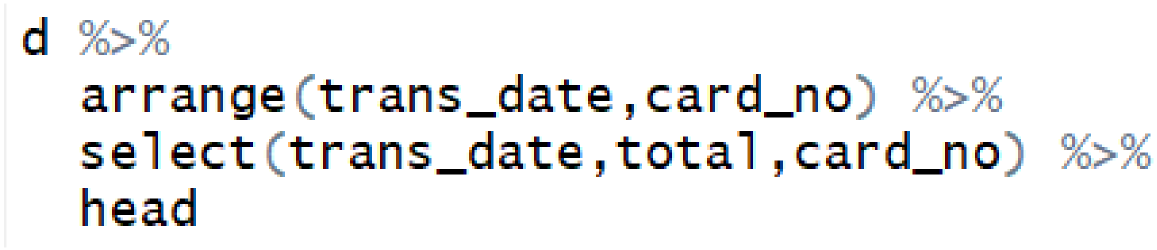


```
d %>%
  arrange(trans_date) %>%
  head

d %>%
  filter(20 < total,total < 100,trans_type=="I") %>%
  arrange(desc(total)) %>%
  head
```

- Use “desc” to get descending ordering.
- Can arrange by multiple columns.

select

- Selects columns from a data frame.

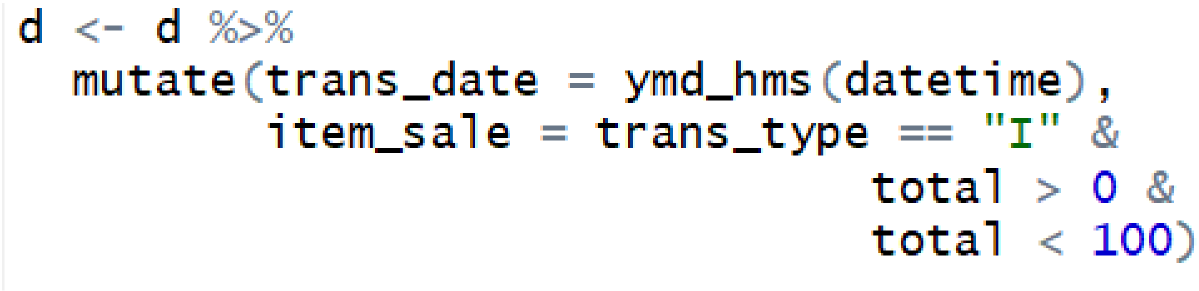


```
d %>%
  arrange(trans_date,card_no) %>%
  select(trans_date,total,card_no) %>%
  head
```

- List as many columns as you want!

mutate

- Adds new columns to a data frame.

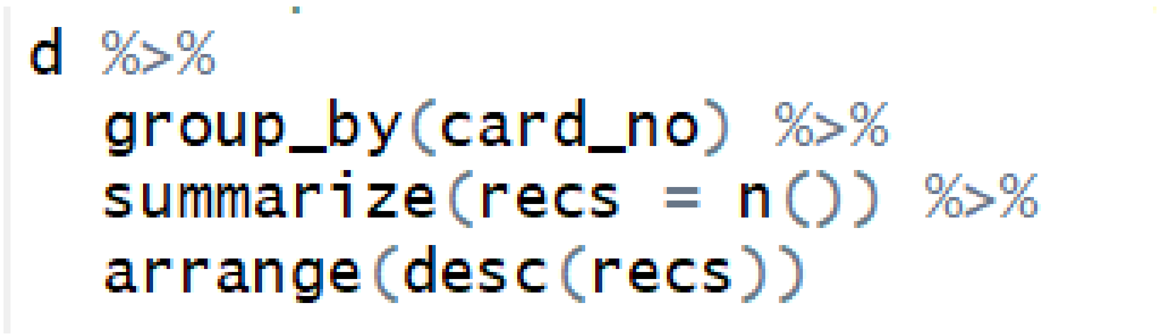


```
d <- d %>%
  mutate(trans_date = ymd_hms(datetime),
         item_sale = trans_type == "I" &
                               total > 0 &
                               total < 100)
```

- Note that data frame isn’t changed in place—you must assign it back to itself for the change to be permanent.

summarize and group_by

- summarize calculates summary statistics.
- group_by allows you to do summary statistics in groups.



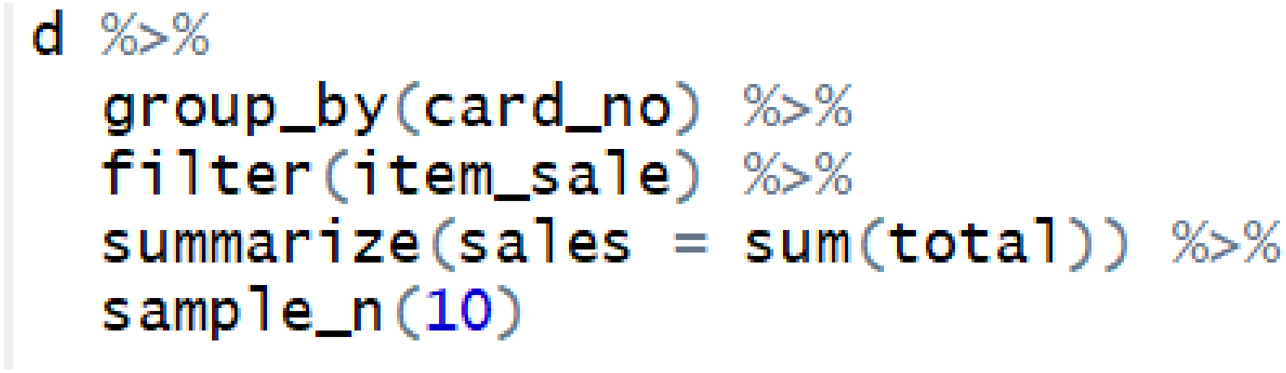
```
d %>%
  group_by(card_no) %>%
  summarize(recs = n()) %>%
  arrange(desc(recs))
```

- I typically call ungroup at the end of these statements if I’m making assignments.
- Never (rarely) use aggregate again!

Ceci n’est pas un pipe

Magrittr pipe operator

- The pipe operator %>% pushes a data frame through these verbs.
- Once you get used to it, code becomes highly readable.



```
d %>%
  group_by(card_no) %>%
  filter(item_sale) %>%
  summarize(sales = sum(total)) %>%
  sample_n(10)
```

`dplyr` Conclusion

- `dplyr` takes a task that’s annoying in R and makes it easier.
- Great way to slice and dice data frames.
- Designed to work well with databases.
- Much faster than subset.
- Pipe operator reduces typing but is a real paradigm shift.
- Lots of information available online. Check out the vignettes.

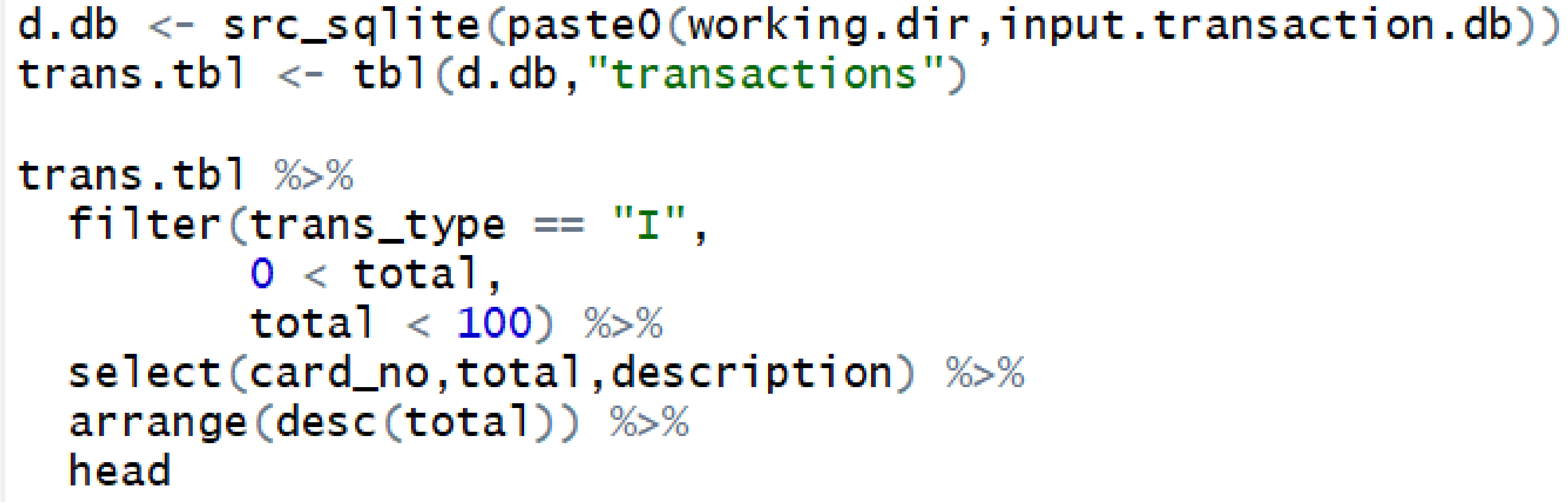
Data manipulation work

Working from the R code on GitHub:

- Summarize sales by owner by month.
  - Do you notice anything unusual in the numbers?
  - If you're familiar with ggplot, make a plot of sales across all time by owner.
- Find the owners with the top five largest sales in the Cheese department (department == 5) in March of 2016.

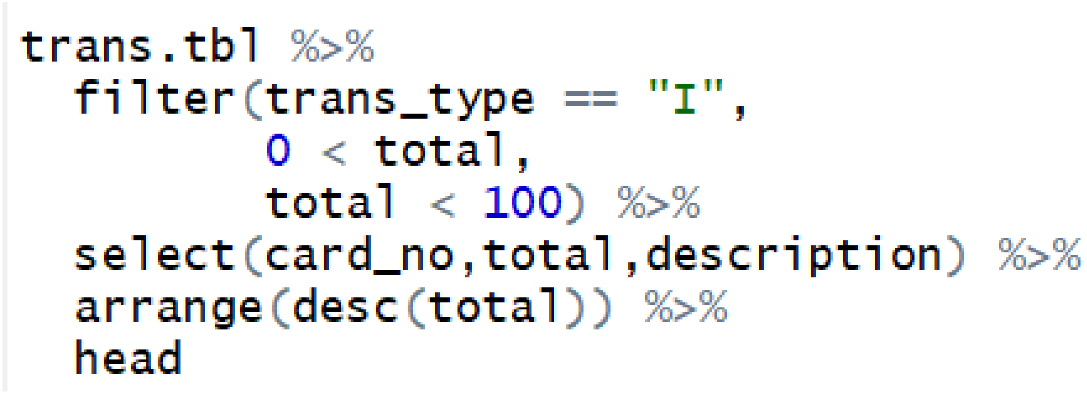
Using databases instead of flat files

- One great feature of `dplyr` is database integration.
- Relatively simple changes get you *way* more power.



```r
d.db <- src_sqlite(paste0(working.dir,input.transaction.db))
trans.tbl <- tbl(d.db,"transactions")

trans.tbl %>%
  filter(trans_type == "I",
         0 < total,
         total < 100) %>%
  select(card_no,total,description) %>%
  arrange(desc(total)) %>%
  head
```

A database example



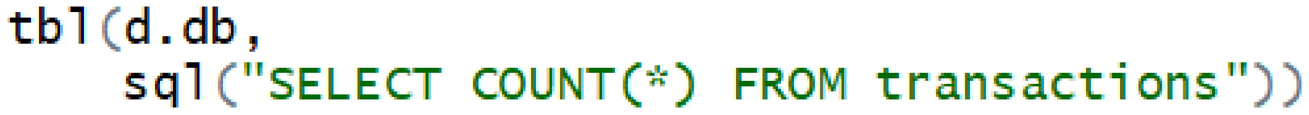
```
trans.tbl %>%
  filter(trans_type == "I",
         0 < total,
         total < 100) %>%
  select(card_no,total,description) %>%
  arrange(desc(total)) %>%
  head
```

Same results as this:

```
SELECT TOP 10 card_no, total, description
FROM transactions
WHERE trans_type = “I” and 0 < total < 100
ORDER BY total DESC
```

Some points about `dplyr` and databases

- Is very performance-focused. You need `collect`, `compute`, or `collapse` to actually get the calculation done.
  - 99% of the time I use `collect`.
- Some functions (`tail`, `nrow`) aren't available for this reason.
  - For instance, this does `nrow`



```
tbl(d.db,
    sql("SELECT COUNT(*) FROM transactions"))
```

- I recommend SQLite for very simple situations, Postgres for everything else, basically.

Trying it out

Use the database to try to answer one or more of the questions from before.

- Summarize sales by owner (bonus: by month).
- Find the owners with the top five largest sales in the Cheese department (department == 5) overall.

Segmentation

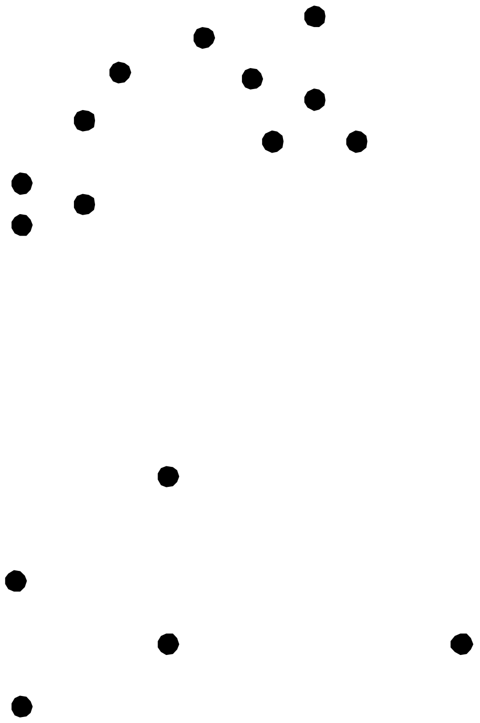
Segmentation

- Grouping customers together is useful for business leaders.
- Typically an unsupervised exercise.
- Cluster analysis is the technique from statistics that groups observations.
- Number of clusters typically chosen in pretty *ad hoc* ways.

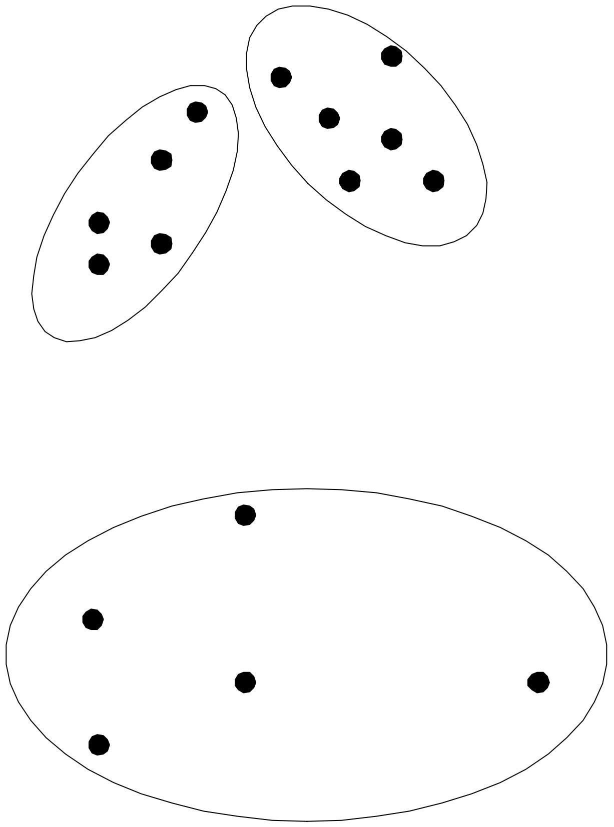
Types of Clusterings

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

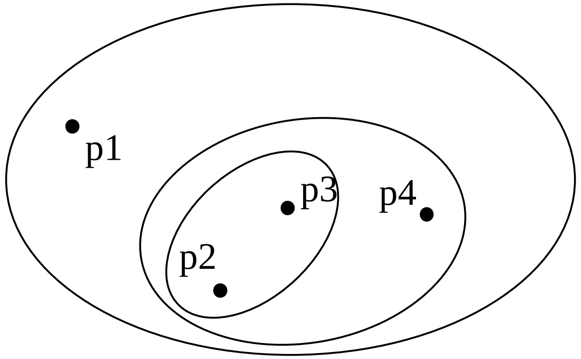


Original Points

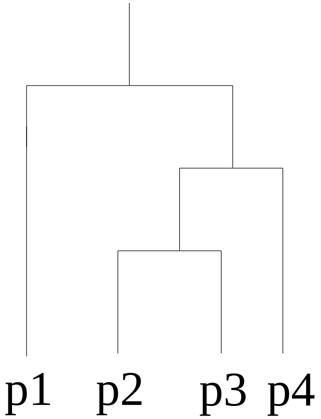


A Partitional Clustering

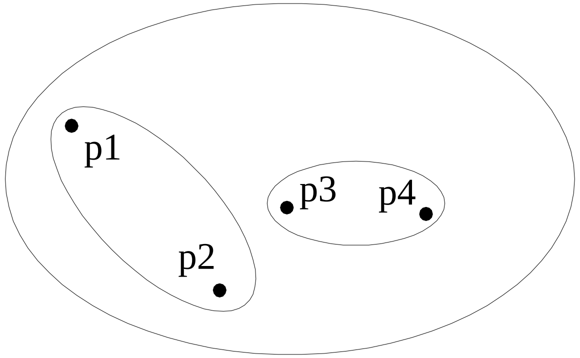
Hierarchical Clustering



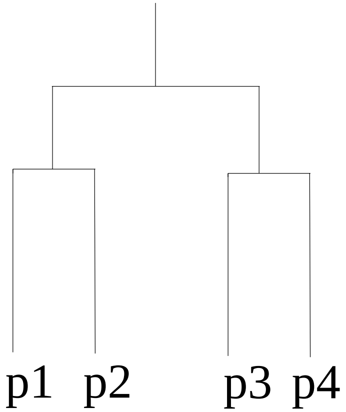
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Types of Clusters: Objective Function

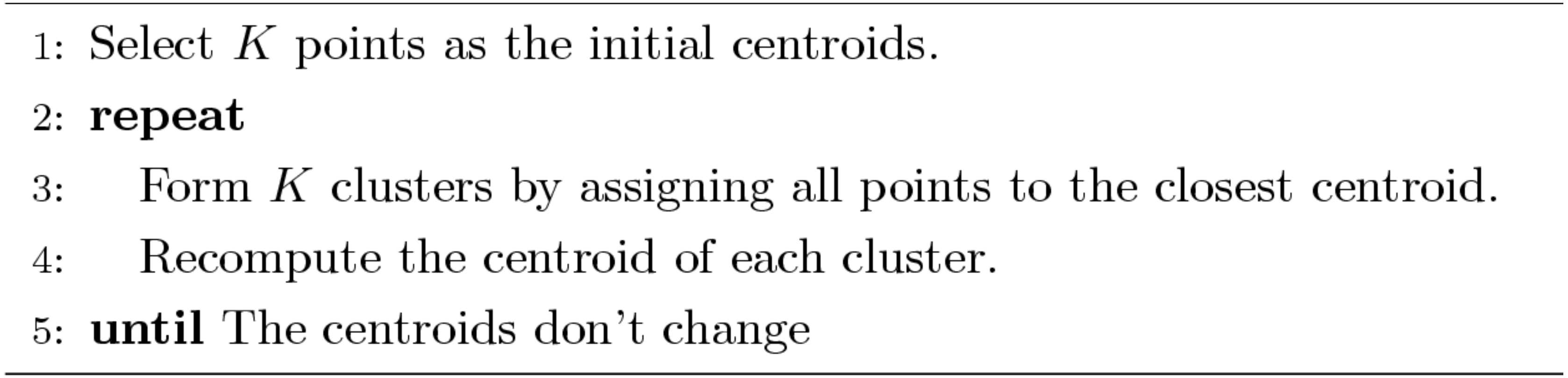
- Finds clusters that minimize or maximize an objective function.
- Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
- Can have global or local objectives.
  - Hierarchical clustering algorithms typically have local objectives
  - Partitioning algorithms typically have global objectives
- A variation of the global objective function approach is to fit the data to a parameterized model.
  - Parameters for the model are determined from the data.
  - Mixture models assume that the data is a ‘mixture' of a number of statistical distributions.

The input data are important

- Distance function: This is a derived measure, but central to clustering
- Sparseness
- Attribute type
- Type of Data
    - Dictates type of similarity
    - Other characteristics, e.g., autocorrelation
- Dimensionality
- Noise and Outliers

K-means Clustering

- Partitioning cluster approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple



```
Select $K$ points as the initial centroids.
repeat
    Form $K$ clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change
```

R exercise

What dimensions might we cluster owners on?

R exercise

What dimensions might we cluster owners on?

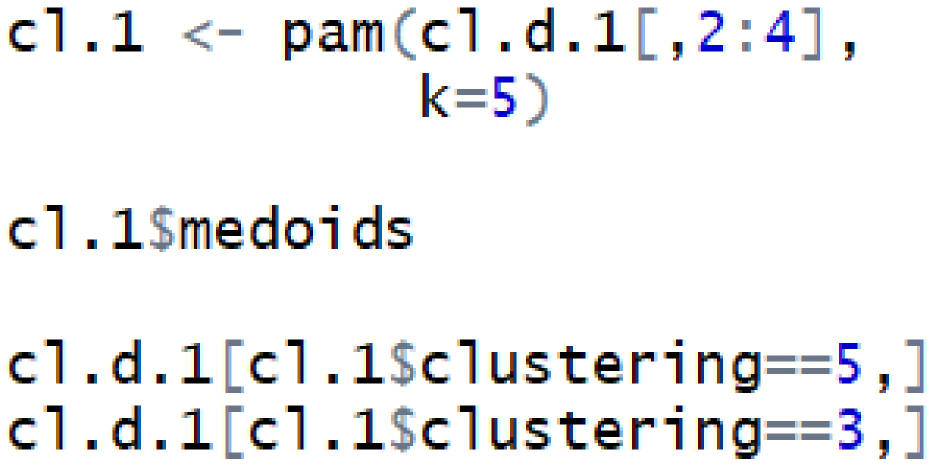
One possible answer:

- Total Spend
- Number of Visits (Unique dates is a good proxy)
- Spend in the Produce department

Take a bit and try to build this table in R.

Clustering in R

- Powerful `cluster` package handles most of what you need.
- The function `pam` does k-means.



```
cl.1 <- pam(cl.d.1[,2:4],
            k=5)

cl.1$medoids

cl.d.1[cl.1$clustering==5,]
cl.d.1[cl.1$clustering==3,]
```

- The code also includes an example with Affinity Propagation clustering.

Evaluation

What issues do we see with this clustering?

Evaluation

What issues do we see with this clustering?

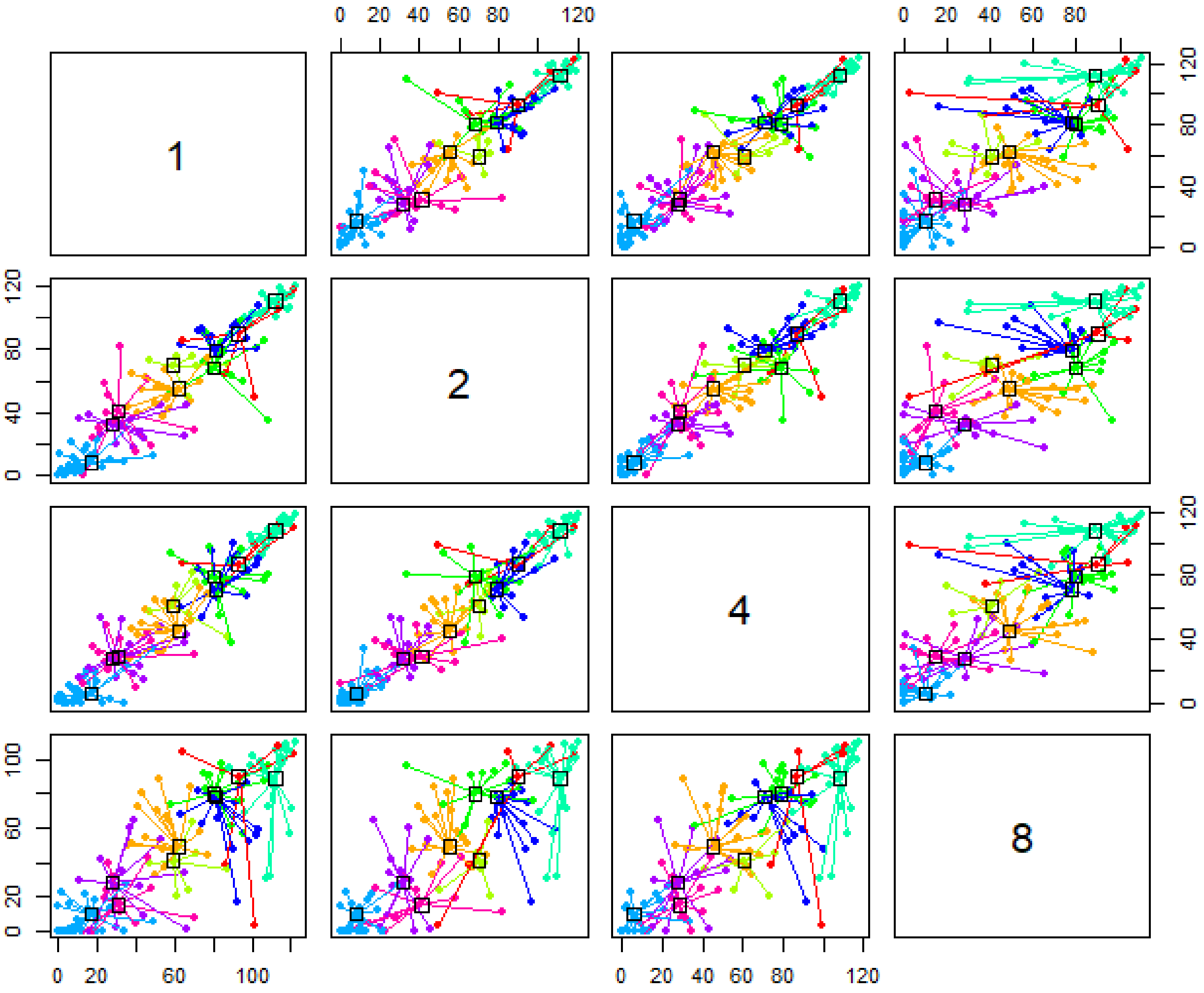
How could we come up with a better measure of distance?

R exercise

Use the R code that transforms department spend to ranks and play around with the clustering.

Re-run the clustering with this new measure. Do the clusters seem more informative?

Affinity Propagation Clustering on Top Depts



Association Analysis

Association Analysis

- A way to look at which items typically occur together.
- Produces rules like

  If a shopper purchases bread, they are also likely to purchase cheese.

  If a shopper purchases chips and salsa, they are likely to purchase beer.
- Goal: Useful, actionable rules. Bonus is surprise factor.
- Inventory curves give total sales, association tells us about co-occurrence.

Association Analysis: Example

10 Wedge *transactions* from the following departments:

1. Pkg Groc, Produce, Ref Groc
2. Fish, Meat, Produce
3. Meat
4. Meat, Produce
5. Pkg Groc
6. Frozen, Juice
7. Pkg Groc
8. Cheese, Pkg Groc, Produce
9. Fish, Meat, Produce
10. Pkg Groc, Produce, Ref Groc

Association Analysis: Example

	CHEESE	FISH	FROZEN	JUICE	MEAT	PKG GROC	PRODUCE	REF GROC
CHEESE	1							
FISH	0	2						
FROZEN	0	0	1					
JUICE BAR	0	0	1	1				
MEAT	0	2	0	0	4			
PKG GROC	1	0	0	0	0	5		
PRODUCE	1	2	0	0	3	3	6	
REF GROC	0	0	0	0	0	2	2	2

Diagonal: Count of Transactions with department.

Off-diagonal: Count of transactions with both departments.

Association Analysis: Example

	CHEESE	FISH	FROZEN	JUICE	MEAT	PKG GROC	PRODUCE	REF GROC
CHEESE	1							
FISH	0	2						
FROZEN	0	0	1					
JUICE BAR	0	0	1	1				
MEAT	0	2	0	0	4			
PKG GROC	1	0	0	0	0	5		
PRODUCE	1	2	0	0	3	3	6	
REF GROC	0	0	0	0	0	2	2	2

Produce and meat are most popular. Frozen and Juice least popular.

Fish is always purchased with meat.

Examples of rules

- If someone buys fish, then they buy meat. (Fish→Meat)
- If someone buys meat, then they buy fish. (Meat→Fish)
- If someone buys produce, then they buy meat. (Produce→Meat)

What makes a rule good?

Support

The number of transactions where the left-hand side (LHS) is true, divided by total transactions.

- Fish→Meat
  - 3 fish transactions out of 10 gives 30%
- Meat→Fish
  - 4/10 = 40%
- Produce→Meat
  - 6/10 = 60%

Support is useful for cutting down our *consideration set*.

Strength

Probability of RHS, given LHS. $A \rightarrow B$, strength is $P(B|A)$.

- Fish→Meat
  - Fish is purchased three times, all of them with meat, so P(Meat|Fish) = 1.
- Meat→Fish
  - Meat is purchased four times, two of them with fish, so P(Fish|Meat) = 0.5
- Produce→Meat
  - Produce is purchased 6 times and meat is purchased in three of those, so P(Meat|Produce) = 0.50

Strength is useful for narrowing our consideration set and measuring rule effectiveness.

Lift

How much more frequent the association than what we'd expect from chance?

$$A \rightarrow B, \text{ lift is } \frac{P(A \cap B)}{P(A) \cdot P(B)}$$

P(Fish) = 0.2, P(Meat) = 0.4, P(Produce) = 0.6

P(Fish∩Meat) = 0.2, P(Produce∩Meat) = 0.3

- Fish→Meat: $\frac{0.2}{0.2 \cdot 0.4} = \frac{0.2}{0.08} = 2.5$
- Meat→Fish: Same
- Produce→Meat: $\frac{0.3}{0.6 \cdot 0.4} = \frac{0.3}{0.24} = 1.25$

R exercise

- We can run association analysis in R using the `arules` package.

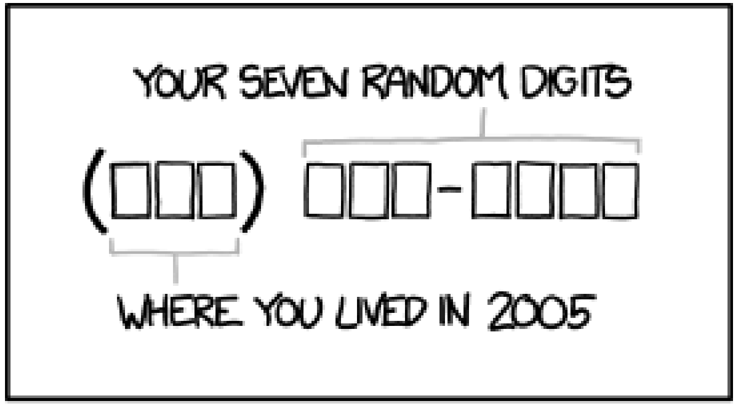
- Step through the R code provided to see the rule-mining in action.

- If you’re way ahead, pick one of segments from the previous section and run an association analysis on the shoppers from that segment.

Wrapping Up

Where we’ve been

- R (or similar tools) is an important part of the data science stack.
- `dplyr` provides powerful data manipulation functionality.
- Segmentation helps us put customers into groups.
  - The distance function is *really* important.
- Association analysis lets us explore the relationship between items.



STRUCTURE OF A US CELL PHONE NUMBER

John Chandler
406-544-8720
john@datainsightsllc.com
www.linkedin.com/in/johncp/

Thanks and Q&A