Customer Clustering for Retailer Marketing

An exploratory data science project with reference to useful R packages

LondonR March 2014
Jonathan Sedar, Applied AI
@jonsedar

Customer Clustering something something something ...

- Thats a convoluted title!
- Client problem is really: “Tell me something about my customer base”
- Time to do some “Exploratory Data Science”
- aka data hacking
- Take a sample of data, fire up an R session and lets see what we can learn…

Overview

Process

Sourcing, Cleaning & Exploration
What does the data let us do?

Feature Creation
Extract additional information to enrich the set

Feature Selection
Reduce to a smaller dataset to speed up computation

Clustering
Finding similar customers without prior information
… and interpreting the results

R Toolkit

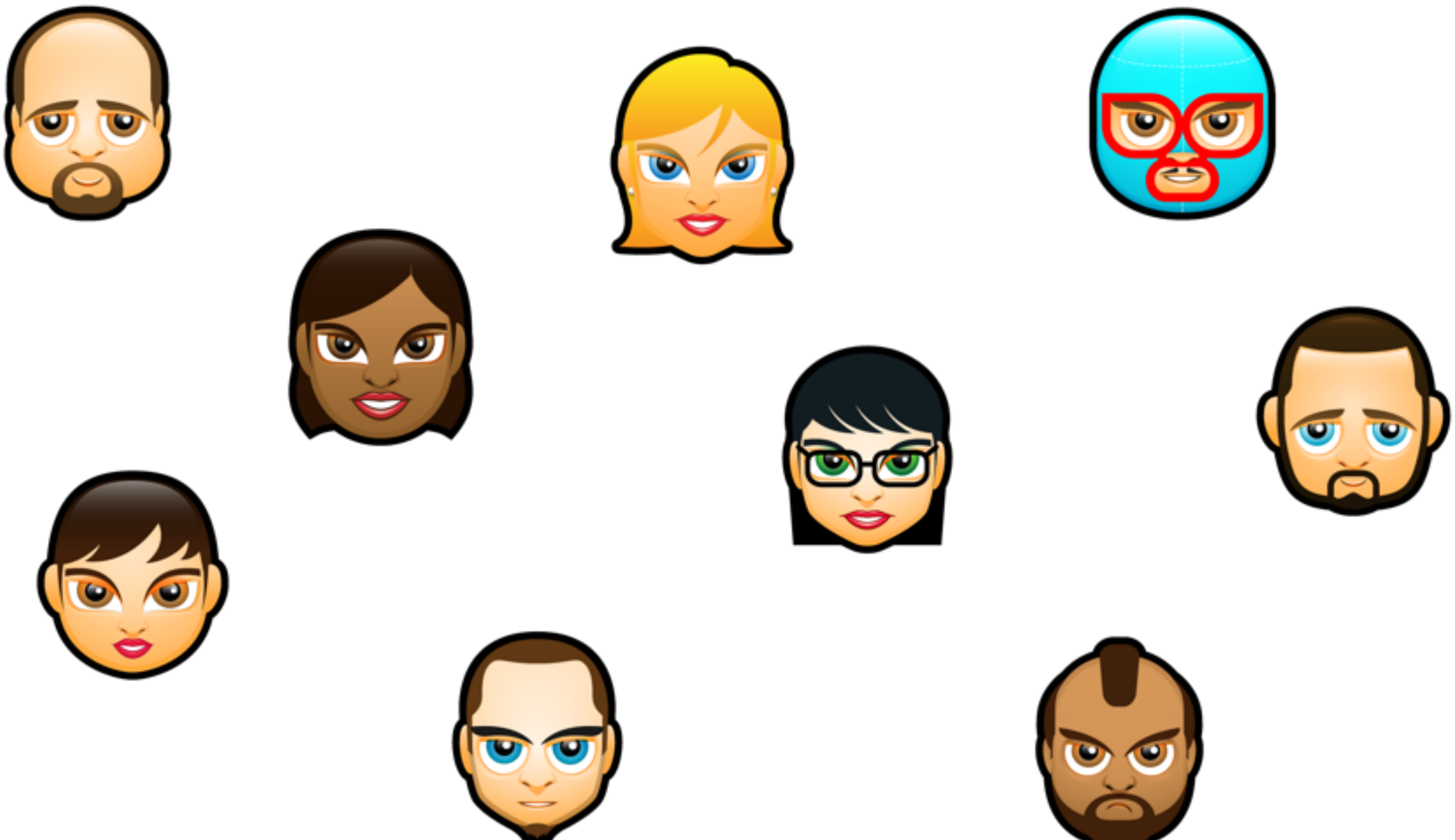
```
read.table {utils}
ggplot {ggplot2}
lubridate {lubridate}
```

```
data.table {data.table}
cut2 {Hmisc}
dcast {reshape2}
```

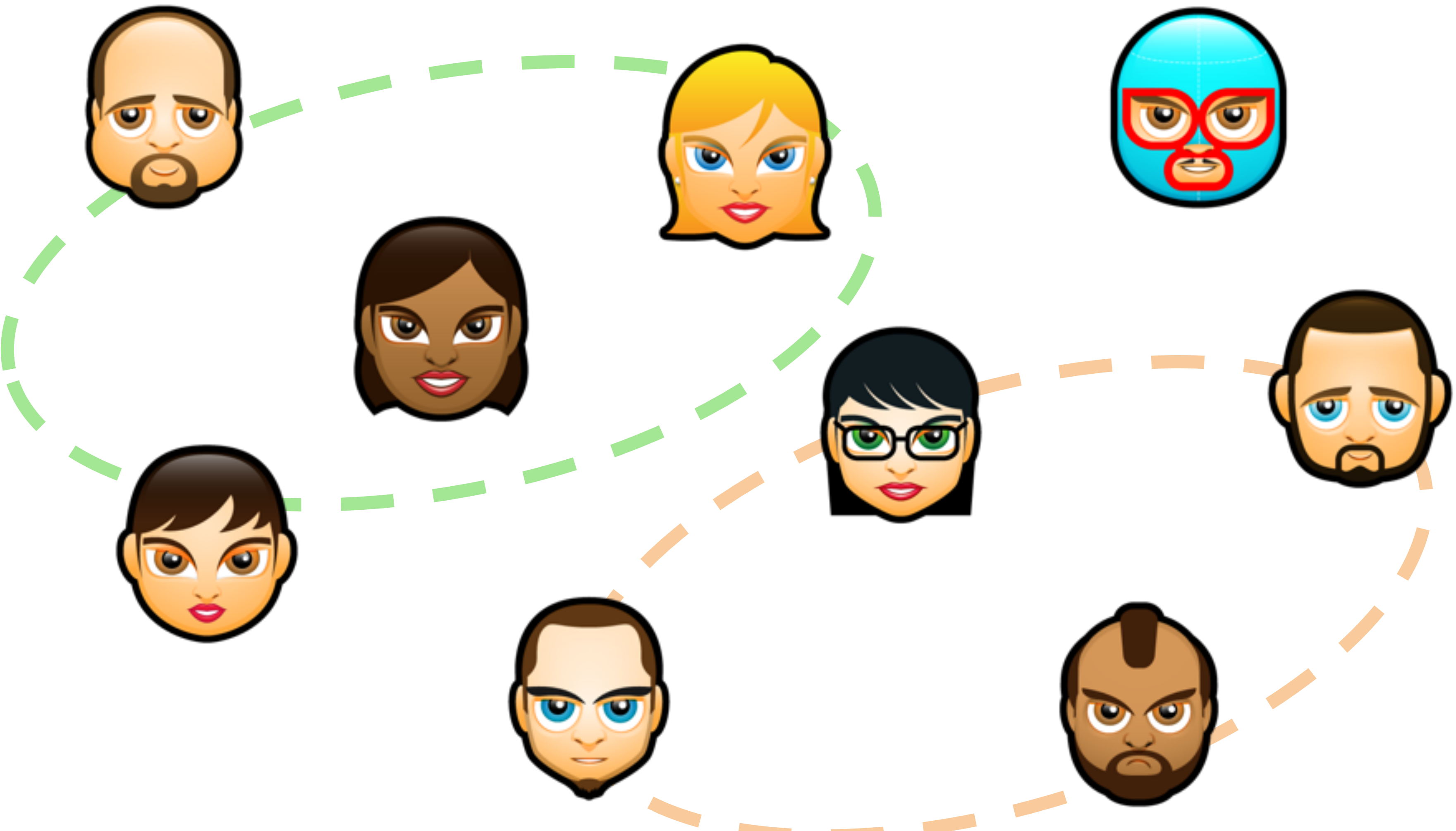
```
covMcd {robustbase}
scale {base}
prcomp {stats}
```

```
mclust {mclust}
```

“Who’s shopping at my stores and how can I market to them?”



Lets try to group them by similar shopping behaviours



Benefits: customer profiling enables targeted marketing and operations

Bob
Lives “SW15”
Age “40”

Bob
Lives “SW15”
Age “40”

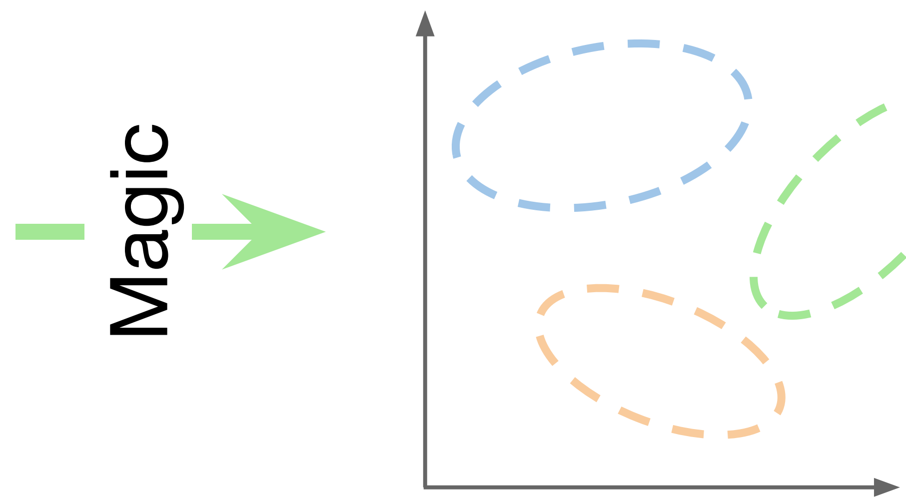
Type: Family First

- Retention offers
- Product promotions
- Loyalty rewards
- Optimise stock levels & store layout

Turning existing data into insights

A real dataset:

32,000 customers
24,000 items
800,000 transactions
… over a 4 month pd



Many ways to approach the problem!
R has tools to help reach a quick solution

Exploratory data science projects tend to have a similar structure

Source your raw data

Exploration & visualisation

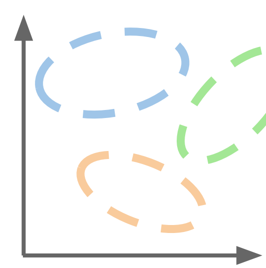
Cleaning & importing

Feature creation

Feature selection

Model creation

Model optimisation & **interpretation**



Data sources

Sample dataset

“Ta-Feng” grocery shopping dataset

```
800,000 transactions
32,000 customer ids
24,000 product ids
4-month period over Winter 2000-2001
http://recsyswiki.com/wiki/Grocery_shopping_datasets
```

```
   trans_date  cust_id   age res_area     product_id quantity price
1: 2000-11-01 00046855     D        E  4710085120468        3    57
2: 2000-11-01 00539166     E        E  4714981010038        2    48
3: 2000-11-01 00663373     F        E  4710265847666        1   135
...
```

A README file, excellent...

4 ASCII text files:

```
# D11: Transaction data collected in November, 2000
# D12: Transaction data collected in December, 2000
# D01: Transaction data collected in January, 2001
# D02: Transaction data collected in February, 2001
```

Curious choice of delimiter and an extended charset

```
# First line: Column definition in Traditional Chinese
#     §È¥¡;∑|≠˚•d∏π;¶~ƒ÷;∞œ∞Ï;∞”´~§¿√˛;∞”´~ΩsΩX;°∆∂q;¶®•ª;æP∞,
# Second line and the rest: data columns separated by ";"
```

Preclean with shell tools

```
awk -F";" 'gsub("\:","",$1)' D02
```

Data definition and audit (2 of 2)

Be prepared for undocumented gotchas:

```
    # 1: Transaction date and time (time invalid and useless)
    # 2: Customer ID
    # 3: Age: 10 possible values,
    #    A <25,B 25-29,C 30-34,D 35-39,E 40-44,F 45-49,G 50-54,H 55-59,I 60-64,J >65
    # actually there's 22362 rows with value K, will assume it's Unknown
    # 4: Residence Area: 8 possible values,
    #    A-F: zipcode area: 105,106,110,114,115,221,G: others, H: Unknown
    #    Distance to store, from the closest: 115,221,114,105,106,110
    #    so we'll factor this with levels "E","F","D","A","B","C","G","H"
    # 5: Product subclass
    # 6: Product ID
    # 7: Amount
    # 8: Asset                 # not explained, low values, not an id, will ignore
    # 9: Sales price
```

Import & preprocess `(read.table)`

Read each file into a data.table, whilst applying basic data types

```
> dtnov <- data.table(read.table(fqn
                                  ,col.names=cl$names
                                  ,colClasses=cl$types
                                  ,encoding="UTF-8"
                                  ,stringsAsFactors=F));
```

Alternatives inc RODBC / RPostgreSQL

```
> con <- dbConnect(dbDriver("PostgreSQL"), host="localhost", port=5432
                     ,dbname=”tafeng”, user=”jon”, password=””)

> dtnov <- dbGetQuery(con,"select * from NovTransactions")
```

Import & preprocess `(lubridate)`

Convert some datatypes to be more useful:

String -> POSIXct datetime (UNIX time UTC) using lubridate

```
> dtraw[,trans_date:=ymd(trans_date)]
```

```
> cat(ymd("2013-11-05"))
1383609600
```

… also, applying factor levels to the residential area

```
> dtraw[,res_area:=factor(res_area
                          ,levels=c("E","F","D","A","B","C","G","H") )]
```

Explore: Group By (data.table)

How many transactions, dates, customers, products, product subclasses?

```
> nrow(dt[,.N,by=cust_id])                          # 32,266
```

Using data.table's dt[i,j,k] structure where:

```
i subselects rows                  SQL WHERE
j selects / creates columns        SQL SELECT
k groups by columns                SQL GROUP BY
```

e.g. the above is:

```
select count (*)
from dt
group by cust_id
```

Clean: logic-level cleaning example

Product hierarchies: assumed many product_ids <-> one product_category, but actually a some product_ids belong to 2 or 3 product_cats:

```
> transcatid <- dt[,list(nbask=length(trans_id)),by=list(prod_cat,
prod_id)]
> transid <- transcatid[,list(ncat=length(prod_cat),nbask=sum(nbask))
                         ,by=prod_id]
> transid[,length(prod_id),by=ncat]
       ncat    V1
    1:    1 23557
    2:    2   253
    3:    3     2
```

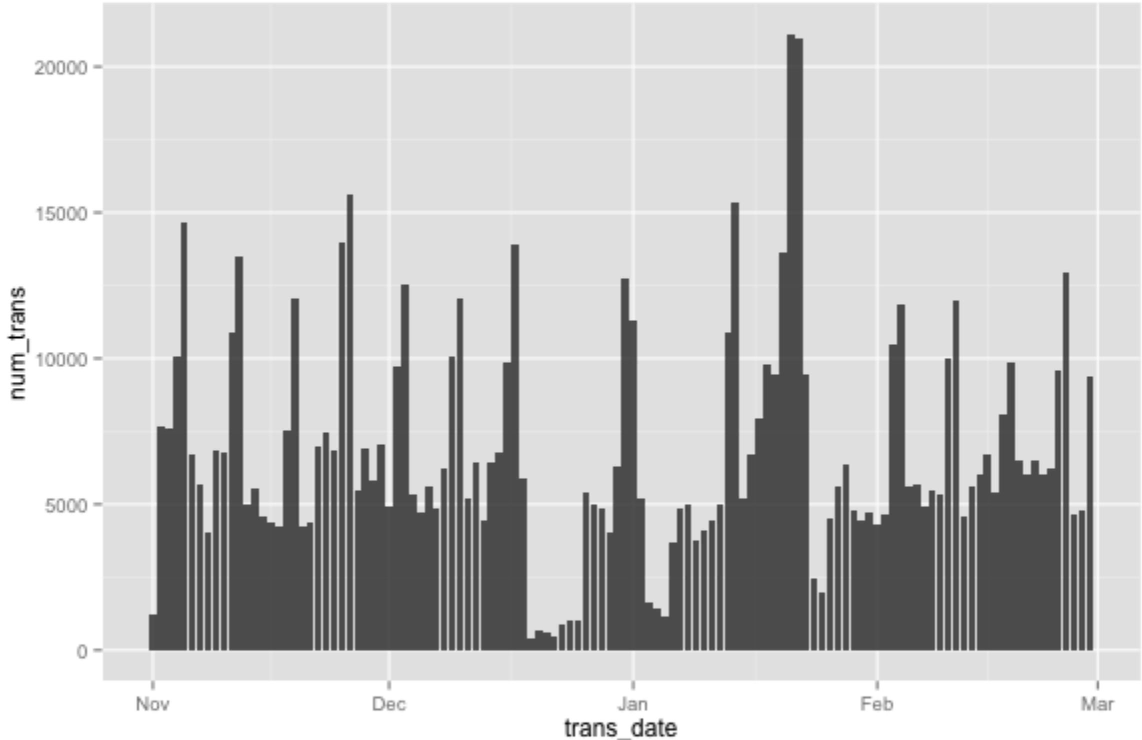
Solution: dedupe. keep prod_id-prod_cat combos with the largest nbask

```
> ids <- transid[ncat>1,prod_id]
> transcatid[prod_id %in% ids,rank :=rank(-nbask),by=prod_id]
> goodprodcat <- transcatid[is.na(rank) | rank ==1,list(prod_cat,
prod_id)]
> dt <- merge(dt,goodprodcat)
```

Explore: Visualise (1 of 4) `(ggplot)`

e.g. transactions by date

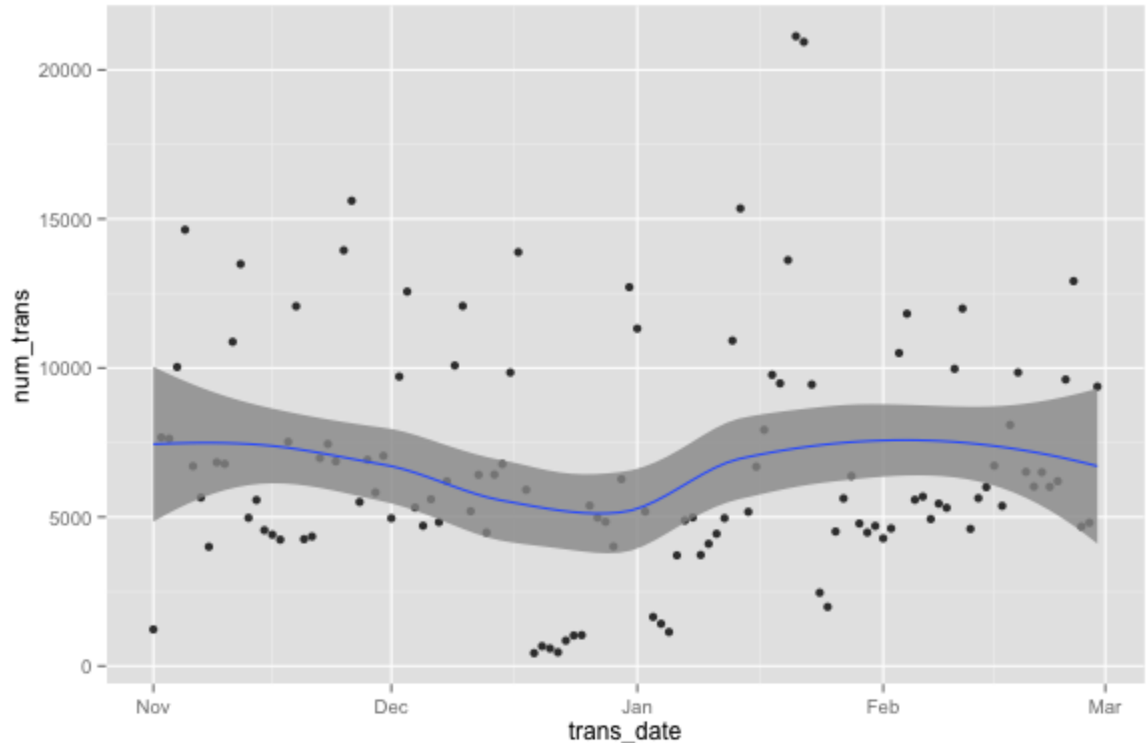
```
p1 <- ggplot(dt[,list(num_trans=length(trans_id)),by=trans_date]) +
    geom_bar(aes(x=trans_date,y=num_trans),stat='identity',alpha=0.8)
plot(p1)
```



Explore: Visualise (1.5 of 4) `(ggplot)`

e.g. transactions by date (alternate plotting)

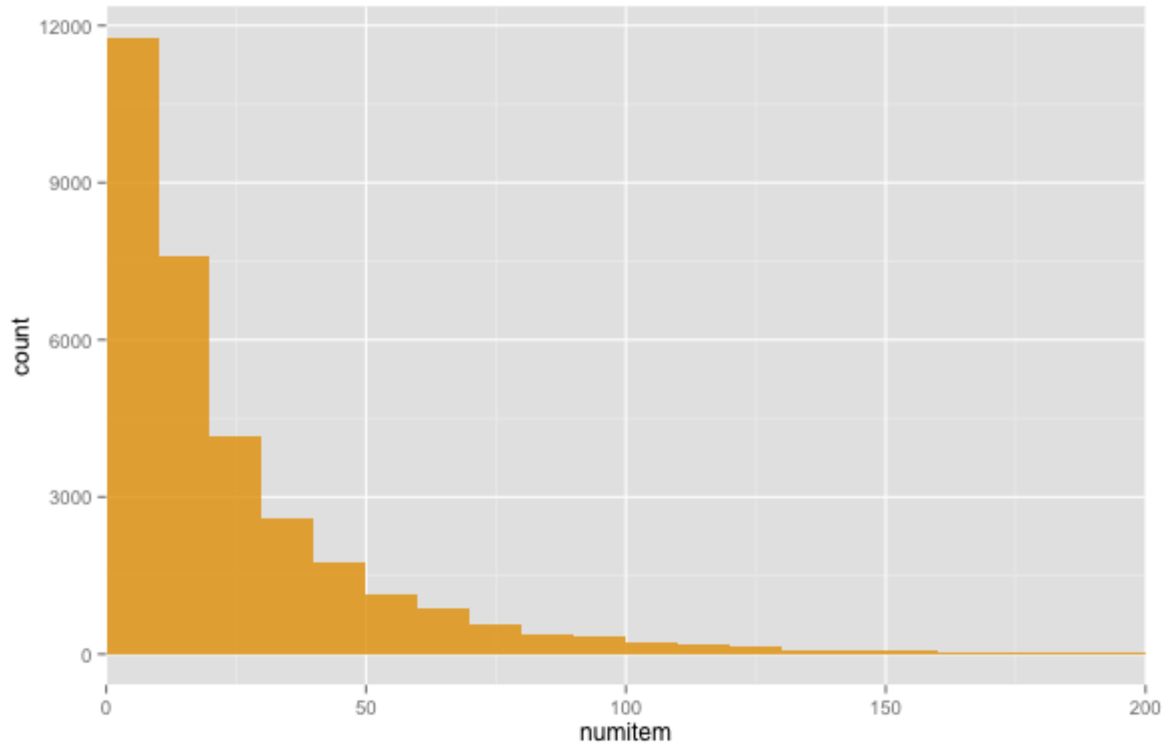
```
p1b <- ggplot(dt[,list(num_trans=length(trans_id)),by=trans_date]) +
    geom_point(aes(x=trans_date,y=num_trans),stat='identity',alpha=0.8) +
    geom_smooth(aes(x=trans_date,y=num_trans), method='loess',alpha=0.8)
plot(p1b)
```



Explore: Visualise (2 of 4) (ggplot)

e.g. histogram count of customers with N items bought

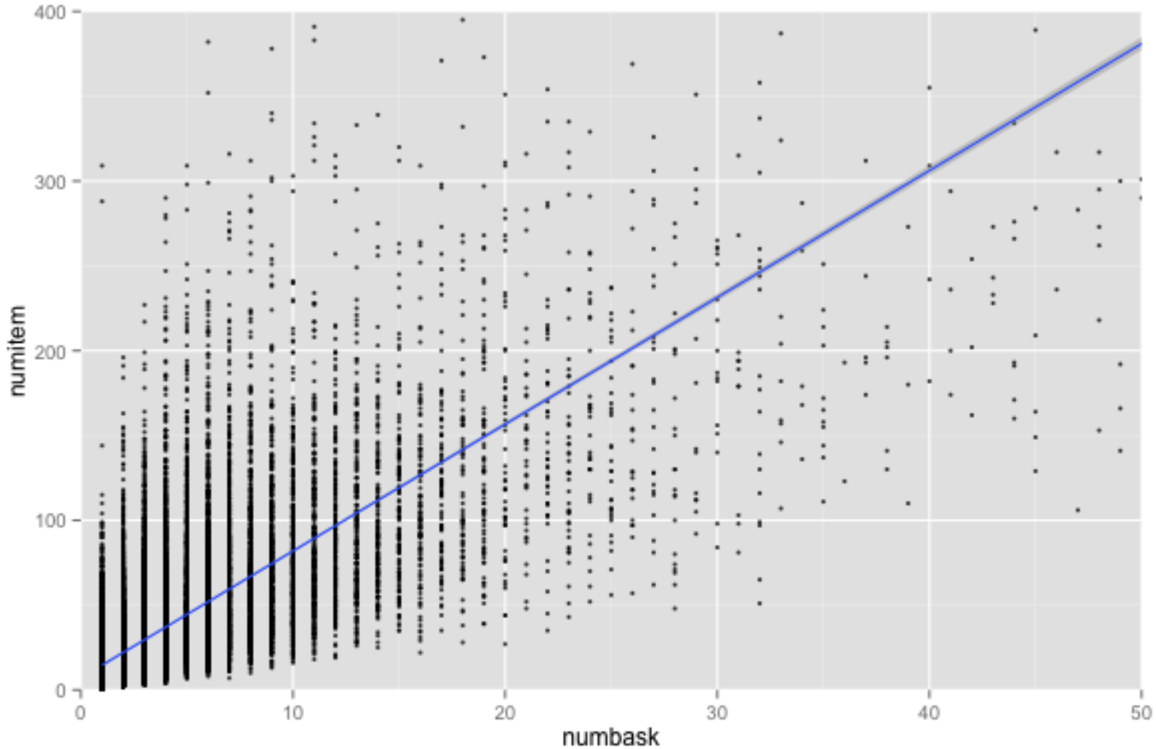
```
p2 <- ggplot(dt[,list(numitem=length(trans_id)),by=cust_id]) +
    geom_bar(aes(x=numitem),stat='bin',binwidth=10,alpha=0.8,fill=orange)
+
    coord_cartesian(xlim=
plot(p2)
```



Explore: Visualise (3 of 4) (ggplot)

e.g. scatterplot of total items vs total baskets per customer

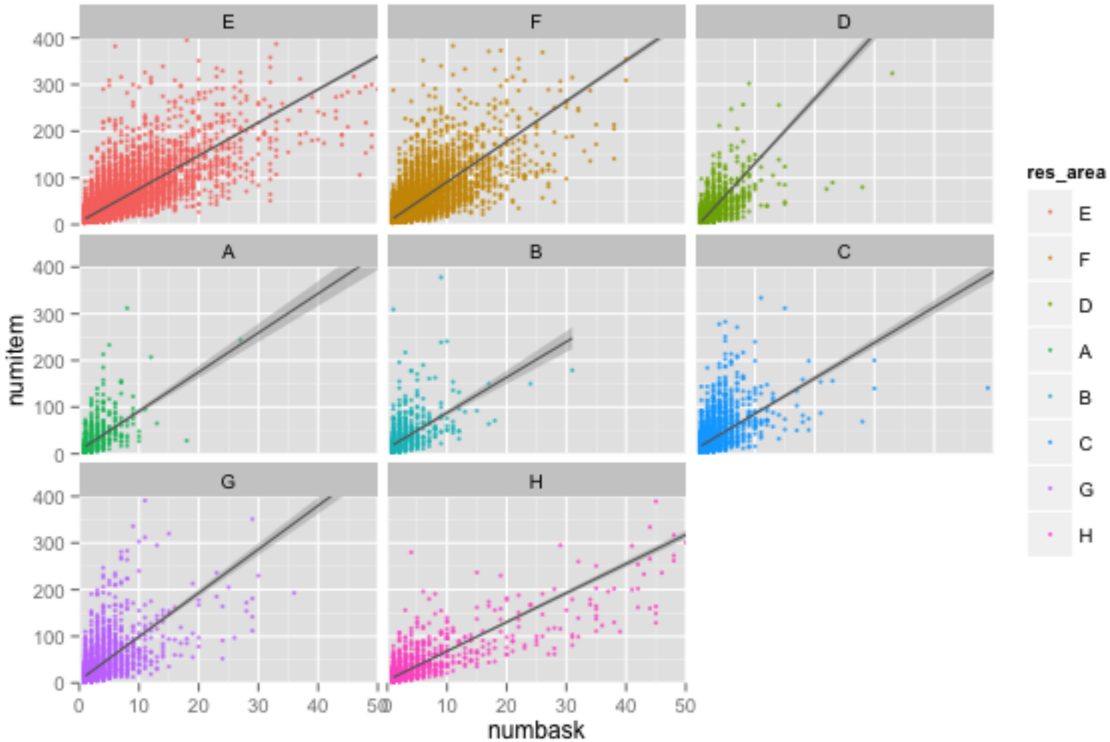
```
p4a <- ggplot(dttt) +
    geom_point(aes(x=numbask,y=numitem),size=1,alpha=0.8) +
    geom_smooth(aes(x=numbask,y=numitem),method="lm")
plot(p4a)
```



Explore: Visualise (4 of 4) (ggplot)

e.g. scatterplot of total items vs total baskets per customer per res_area

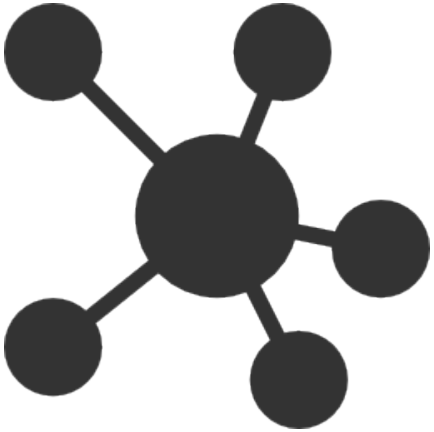
```
p5 <- ggplot(dttt) +
    geom_point(aes(x=numbask,y=numitem,color=res_area),size=1,alpha=0.8)
+
    geom_smooth(aes(x=numba
+
    facet_wrap(~res_area)
plot(p5)
```




```
A-F: zipcode area:
     105,106,110,114,115,221
G: others
H: Unknown

Dist to store, from closest:
     E < F < D < A <B < C
```

Can we find or extract more info?



Create New Features

Per customer (32,000 of them):

Counts:

```
# total baskets (==unique days)
# total items
# total spend
# unique prod_subclass,  unique prod_id
```

Distributions (min - med - max will do):

```
# items per basket
# spend per basket
# product_ids , prod_cats per basket
# duration between visits
```

Product preferences

```
# prop. of  baskets in the N bands of product cats & ids by item pop.
# prop. of baskets in the N bands of product ids by item price
```

Frequency Summaries (data.table)

Pretty straightforward use of group by with data.table

```
> counts <- dt[,list(nbask=length(trans_id)
                     ,nitem=sum(quantity)
                     ,spend=sum(quantity*price))
               ,by=list(cust_id)]

> setkey(counts,cust_id)

> counts
     cust_id nbask nitem spend
  1: 00046855     1     3   171
  2: 00539166     4     8   300
  3: 00663373     1     1   135
```

Frequency Distributions (data.table)

Feature Creation

Again, making use of data.table and list groupby to form new data table

```
> dists_ispb <- dt[,list(nitem=sum(quantity)
                          ,spend=sum(quantity*price))
                  ,by=list(cust_id,trans_date)]

> dists_ispb <- dists_ispb[,list(ipb_max=max(nitem)
                               ,ipb_med=median(nitem)
                               ,ipb_min=min(nitem)
                               ,spb_max=max(spend)
                               ,spb_med=median(spend)
                               ,spb_min=min(spend))
                          ,by=cust_id]
> setkey(dists_ispb,cust_id)
```

Considerations: it is acceptable to lose datapoints?

Feature: duration between visits

If customers visited once only, they have value NA - causes issues later

Solutions:

A: remove them from modelling? wasteful in this case (lose 30%!)

But maybe we don't care about classifying one-time shoppers

B: or give them all the same value

But which value? all == 0 isn't quite true, and many will skew clustering

C: impute values based on the global mean and SD of each col

Usually a reasonable fix, except for ratio columns, where very clumsy and likely misleading, requiring mirroring to get +ve axes

Product Preferences (1 of 2) (Hmisc)

Trickier, since we don't have a product hierarchy

e.g Food > Bakery > Bread > Sliced White > Hovis

But we do price per unit & inherent measure of popularity in the transaction log, e.g.

```
> priceid <- dt[,list(aveprice=median(price)),by=prod_id]

> priceid[,prodid_pricerank:=LETTERS[as.numeric(cut2(aveprice,g=5))]]
# A low, E high

> priceid
            prod_id aveprice prodid_pricerank
    1: 4710085120468       21                A
    2: 4714981010038       26                A
    3: 4710265847666      185                D
```

Product Preferences (2 of 2) (dcast)

Now:

1. Merge the product price class back onto each transaction row
2. Reformat and sum transaction count in each class per customer id, e.g.

```
> dtpop_prodprice <- data.table(dcast(dtpop
                                         ,cust_id~prodid_pricerank
                                         ,value.var="trans_id"))

> dtpop_prodprice
      cust_id  A   B  C  D  E
   1: 00001069 1   3  3  2  2
   2: 00001113 7   1  4  5  1
   3: 00001250 6   4  0  2  2
```

3. And further process to transform counts to proportions per row

Too many features, outliers?

Currently have a data.table 20,000 customers x 40 synthetic features
… which we hope represent their behaviour sufficiently to distinguish them

However, more features -> heavier processing for clustering

Can we / should we lighten the load?

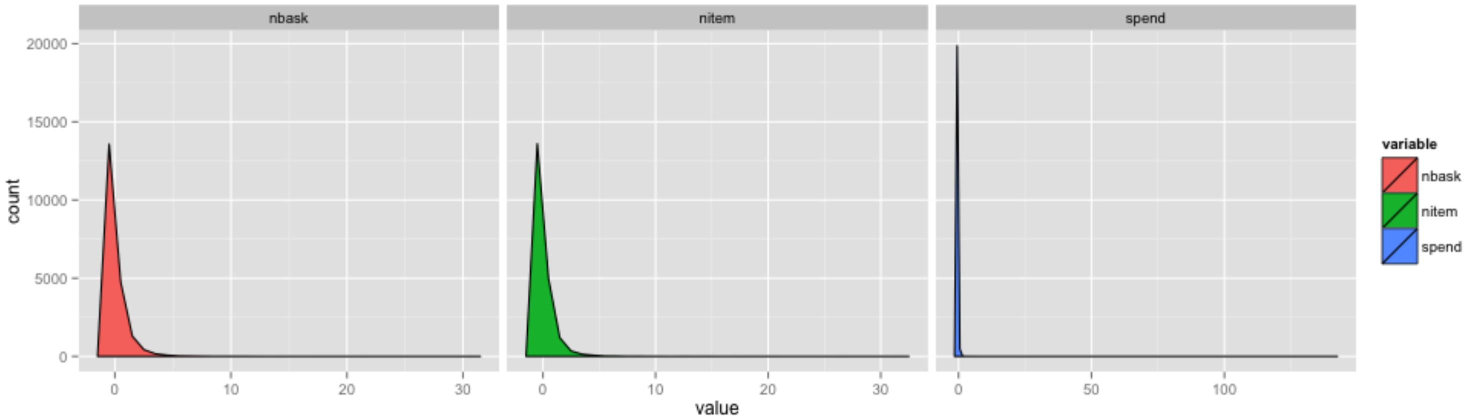
Clean: Removing Outliers

Feature Selection

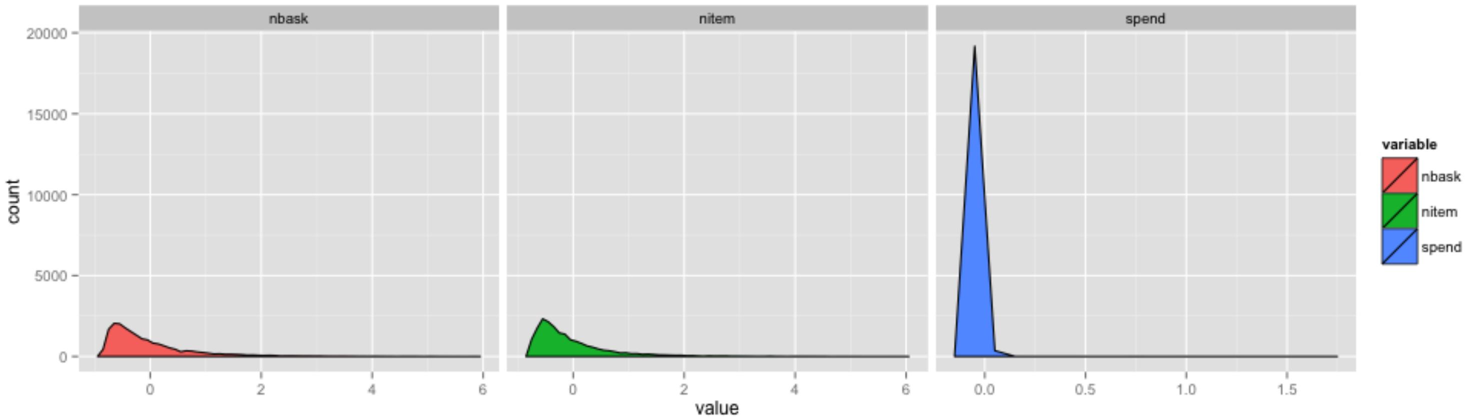
Clustering doesn't tend to like outliers, can we safely remove some?

Simple method: calculate z-score and threshold at some std dev value

Pre:



Post:



Clean: Removing Outliers `(covMcd)`

More fair method: calculate threshold based on Mahalanobis distance from the group's 'central' value (Minimum Covariance Determinant).

We'll use covMcd which implements Rousseeuw's FastMCD.

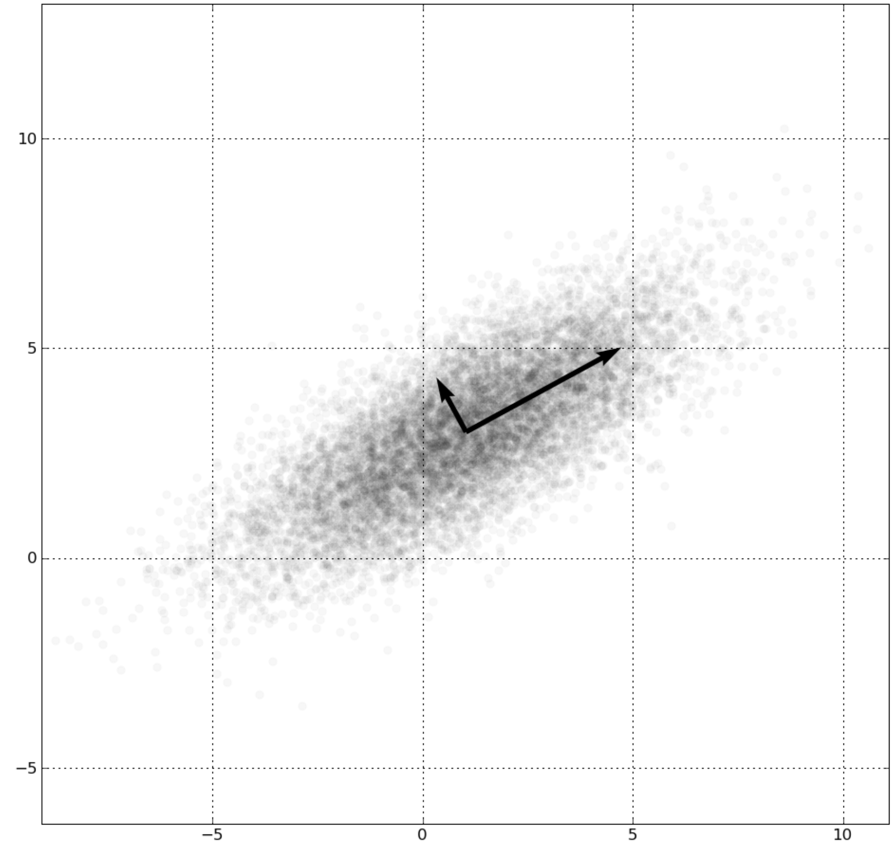
- Find a given proportion (h) of "good" observations which are not outliers and compute their empirical covariance matrix.
- Rescale to compensate the performed selection of observations.
- Assign weights to observations according to their Mahalanobis distance across all features.
- Cutoff at a threshold value, conventionally h/2

Principal Components Analysis (PCA)

Standard method for reducing dimensionality, maps each datapoint to a new coordinate system created from principal components (PCs).

PCs are ordered:

- 1st PC aligned to max variance in all features
- 2nd PC aligned to remaining max variance in orthogonal plane
  ...etc.



Each datapoint had N features; now it has N PC values which are a composite of the original features.

We can now use the first few PCs for clustering and keep the majority of the variance.

PCA (1 of 2) (scale & prcomp)

Scale first, so we can remove extreme outliers in original features

```
> cstZi_sc <- scale(cst[,which(!colnames(cst) %in% c("cust_id")),with=F])
> cstZall <- data.table(cst[,list(cust_id)],cstZi_sc)
```

Now all features are in units of 1 s.d.

For each row, if any one feature has value > 6 s.d., record in a filter vector

```
> sel <- apply(cstZall[,colnames(cstZi),with=F]
                 ,1
                 ,function(x){if(max(abs(x))>6){T}else{F}})
> cstZoutliers <- cstZall[sel]
> nrow(cstZoutliers)      # 830 (/20381 == 4% loss)
```

Health warning: we've moved the centre, but prcomp will re-center for us

PCA (2 of 2) (scale & prcomp)

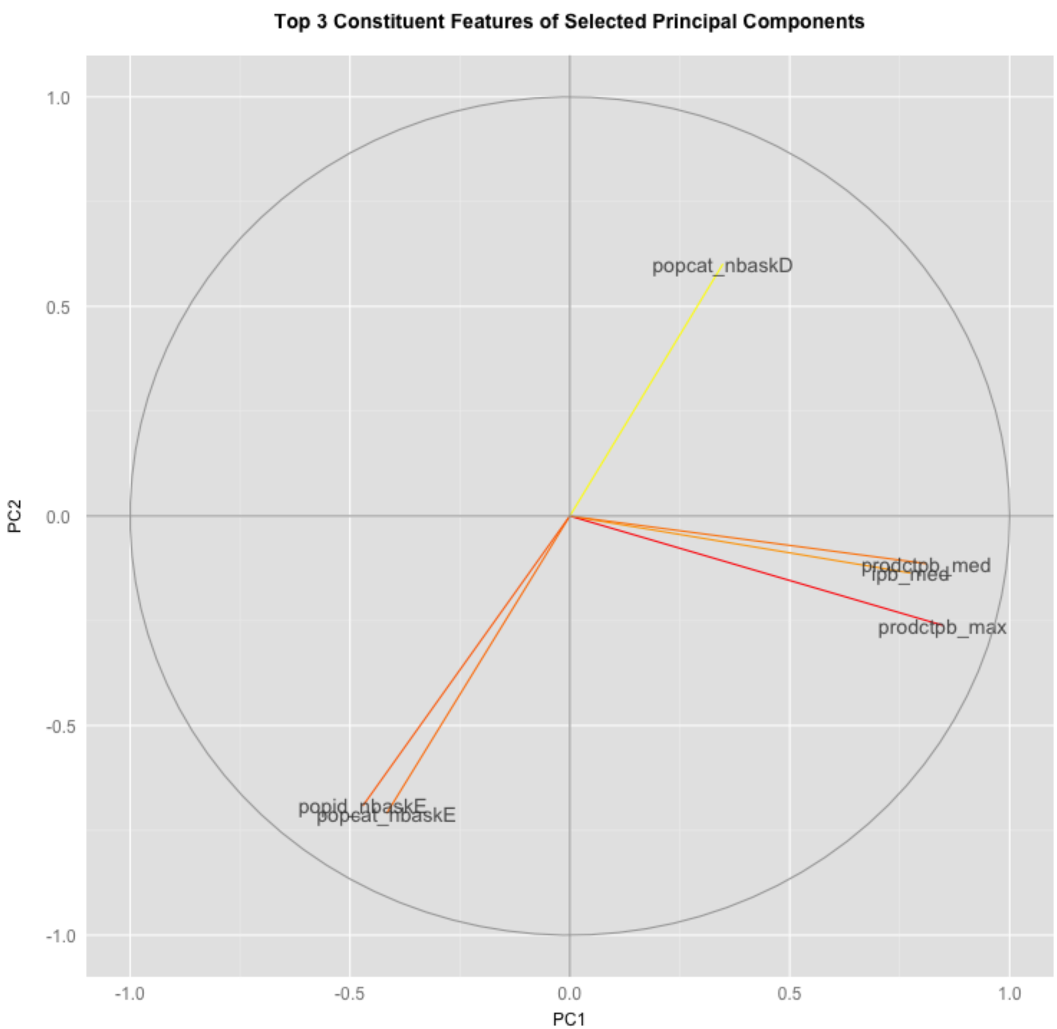
And now run prcomp to generate PCs

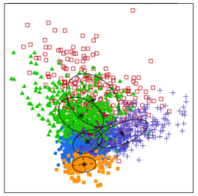
```
> cstPCAi <- prcomp(cstZ[,which(!colnames(cstZ) %in% c("cust_id")),
with=F])
> cstPCAi$sdev                 # sqrt of eigenvalues
> cstPCAi$rotation             # loadings
> cstPCAi$x                    # PCs (aka scores)
> summary(cstPCAi)
    Importance of components:
                              PC1    PC2    PC3     PC4 … etc
    Standard deviation     2.1916 1.8488 1.7923 1.37567
    Proportion of Variance 0.1746 0.1243 0.1168 0.06882
    Cumulative Proportion  0.1746 0.2989 0.4158 0.48457
```

Wait, prcomp Vs princomp?

Apparently princomp is faster but potentially less accurate. Performance of prcomp is very acceptable on this small dataset (20,000 x 40)

PCA quick visualisation (3 of 2)





Finally! Lets do some clustering

Unsupervised learning

Many techniques inc.

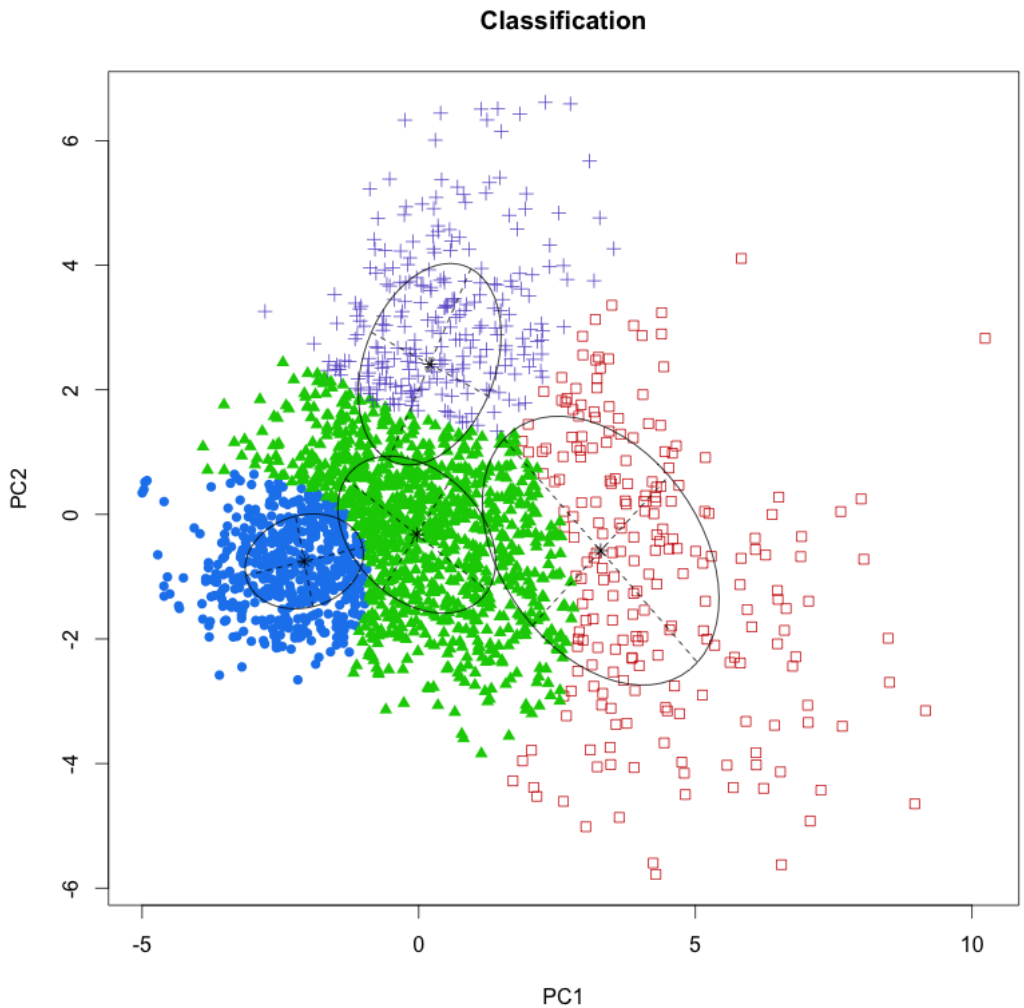
Hierarchical `hclust{stats}`

K-Means `kmeans{stats}`

DBSCAN `dbscan{fpc}`

http://cran.r-project.org/web/views/Cluster.html

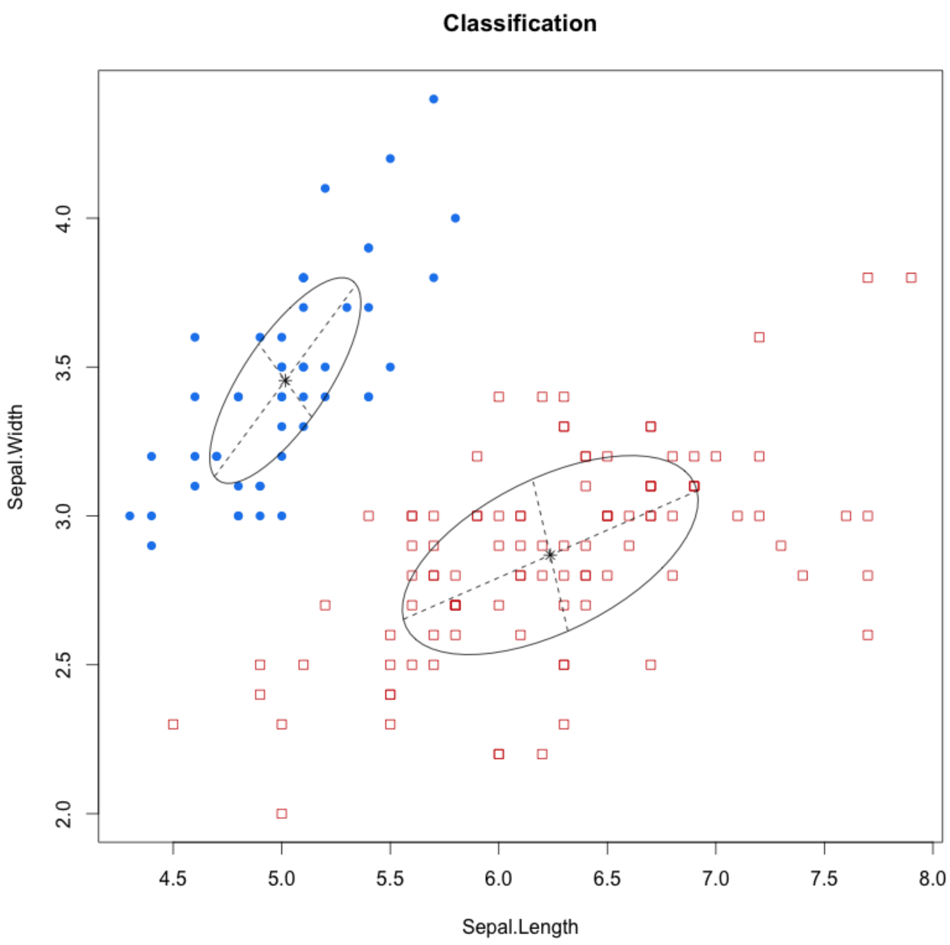
… we're going to play with mixture models



Finite Mixture Modelling

Assume each datapoint has a mixture of classes, each explained by a different model.

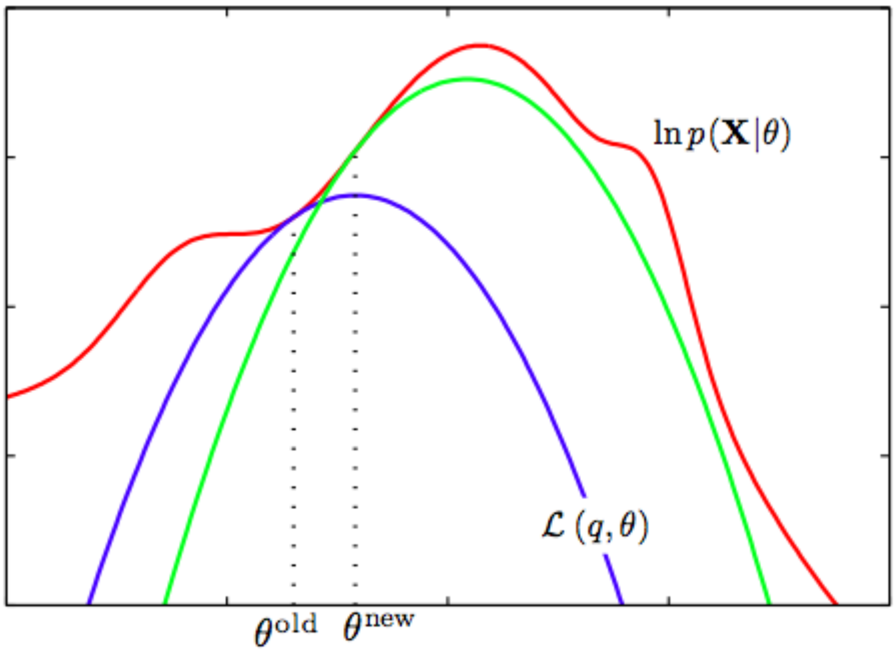
Pick a number of models and fit to the data, best fit wins



Gaussian Mixture Modelling (GMM)

- Models have Gaussian dist., we can vary params
- Place N models at a random point, move and fit to data using Expectation Maximisation (EM) algorithm

- EM is iterative method of finding the local max likelihood estimate.
- Slow but effective
- GMM advantage over e.g. k-means is ability to vary model params to fit better



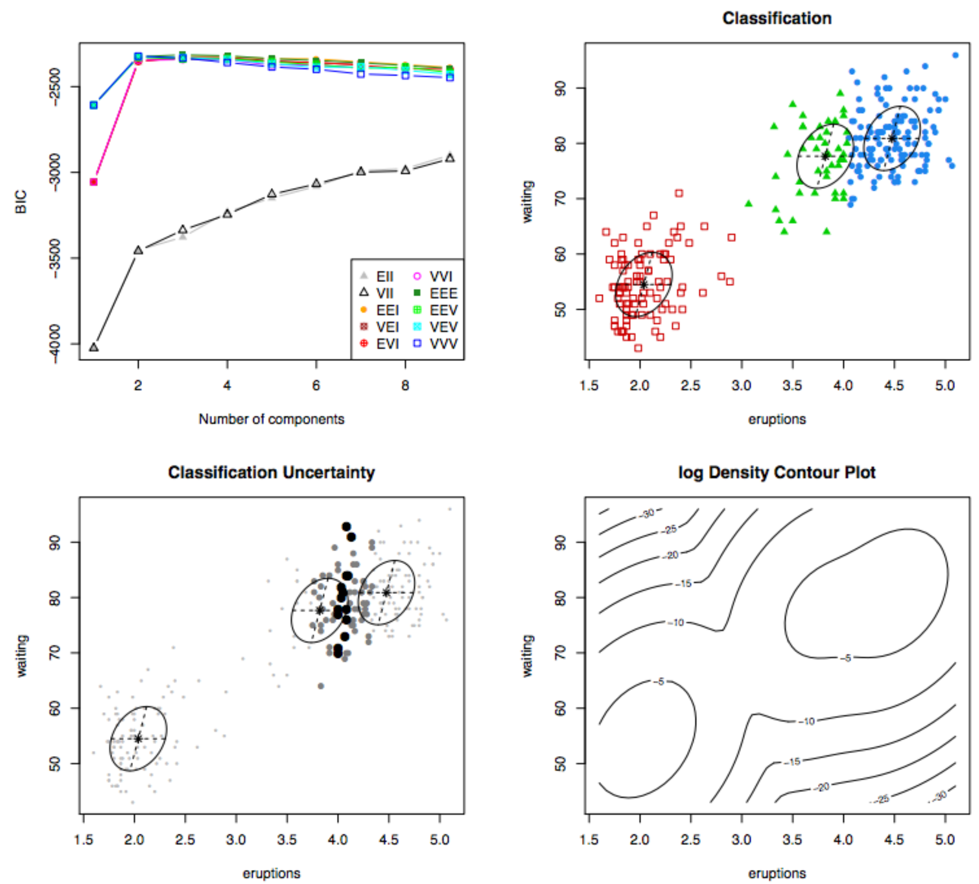
Chris Bishop, PRML 2006

http://ai.stanford.edu/~chuongdo/papers/em_tutorial.pdf

Of course there's an R package (mclust)

Mclust v4, provides:

- Clustering, classification, density estimation
- Auto parameter estimation
- Excellent default plotting to aid live investigation



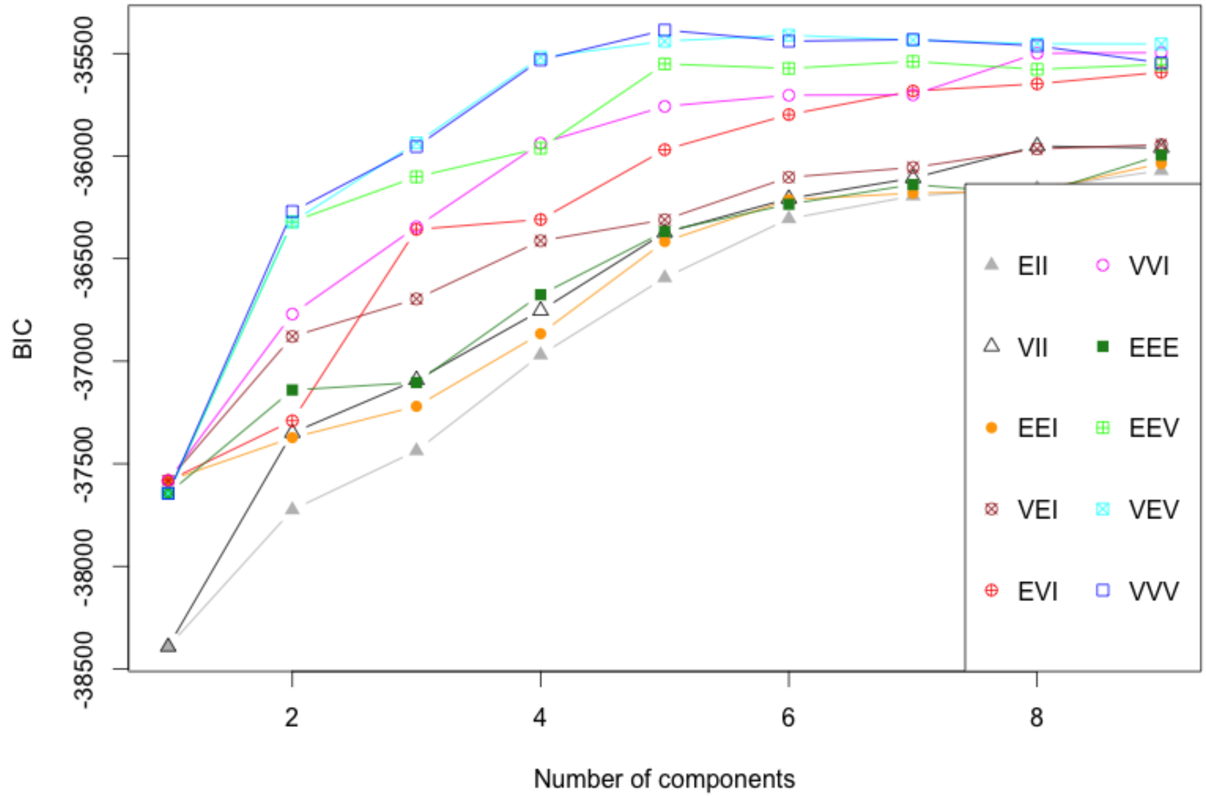
In CRAN and detail at http://www.stat.washington.edu/mclust/

Finding the optimal # models (mclust)

Will automatically iterate over a number of models (components) and covariance params

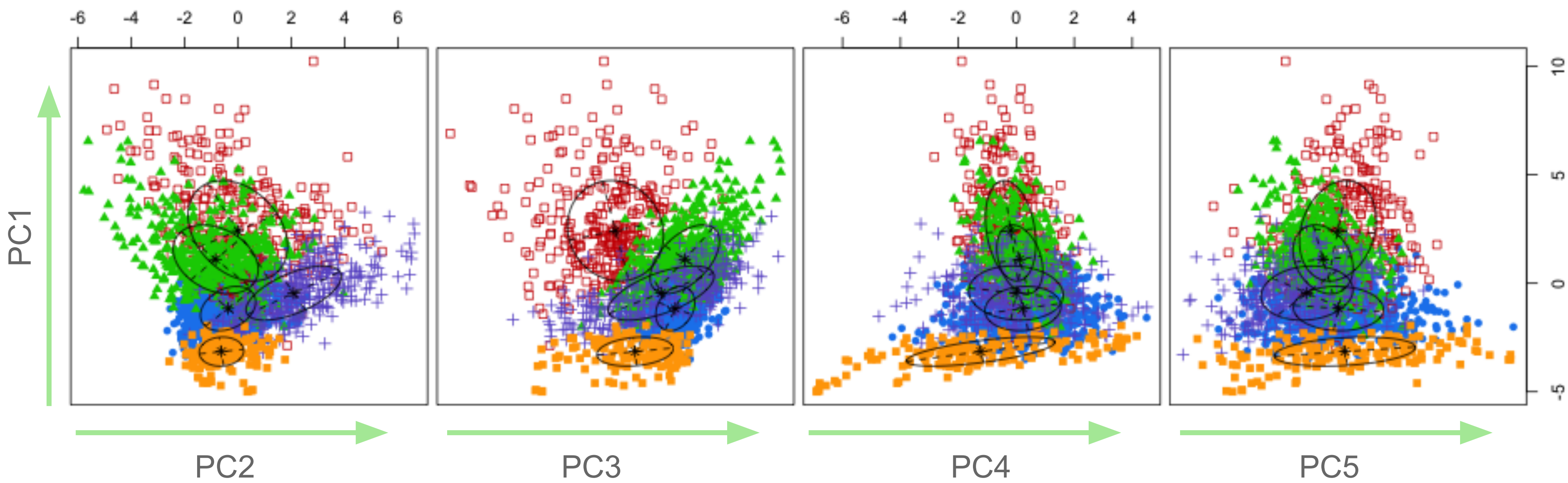
Will use the combination with best fit (highest BIC)

C5, VVV



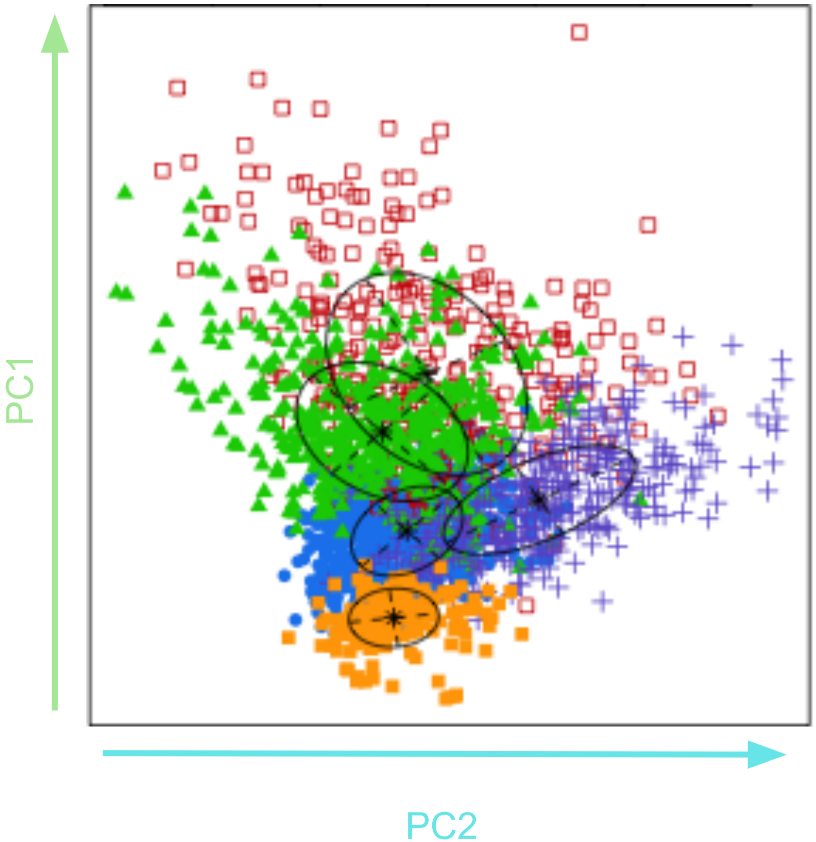
Interpreting model fit (1 of 3) (mclust)

The classification pairs-plot lets us view the clustering by principal component axis-pairs



Interpreting model fit (2 of 3) (mclust)

'Read' the distributions w.r.t PC components



PC1: "Variety axis"

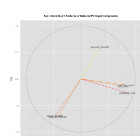
Distinct products per basket and raw count of distinct products overall

prodctpb_max	0.85
prodctpb_med	0.81
ipb_med	0.77
ipb_max	0.77
nprodcat	0.75

PC2: "Spendy axis"

Prop. baskets containing expensive items, and simply raw count of items and visits

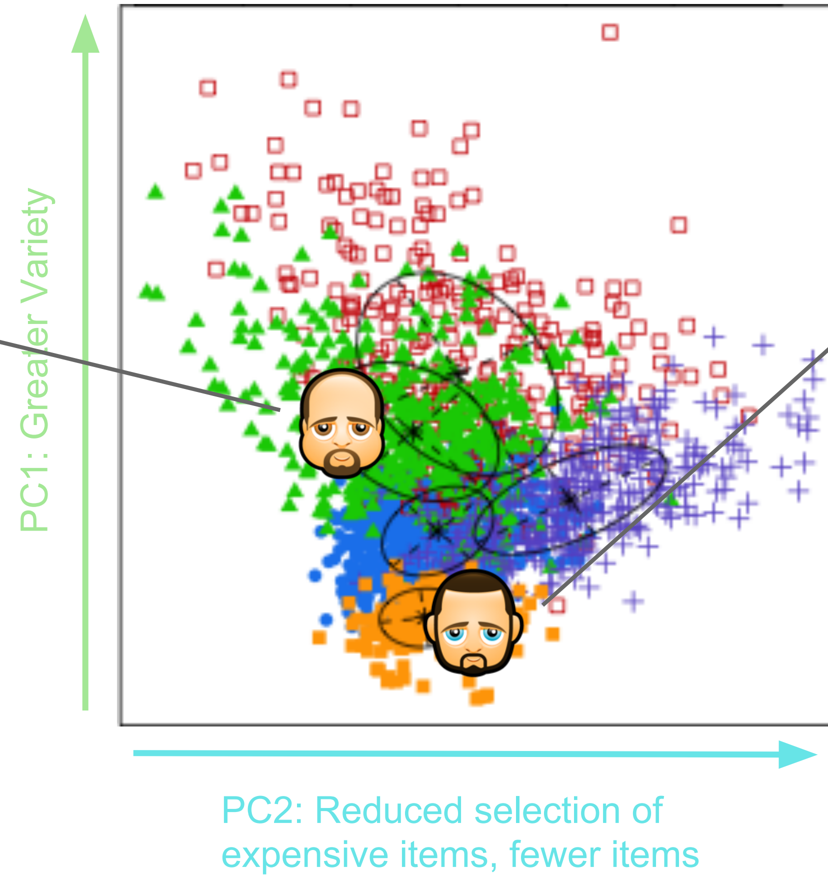
popcat_nbaskE	-0.71
popid_nbaskE	-0.69
popcat_nbaskD	0.60
nbask	-0.51
nitem	-0.51



Interpreting model fit (3 of 3) (mclust)

‘Read’ the distributions w.r.t PC components

Bob
Spendier,
higher variety,
family oriented?



Charles
Thriftier,
reduced selection,
shopping to a
budget?

We covered:

Process

Sourcing, Cleaning & Exploration
What does the data let us do?

Feature Creation
Extract additional information to enrich the set

Feature Selection
Reduce to a smaller dataset to speed up computation

Clustering
Finding similar customers without prior information
… and interpreting the results

R Toolkit

```
read.table {utils}
ggplot {ggplot2}
lubridate {lubridate}
```

```
data.table {data.table}
cut2 {Hmisc}
dcast {reshape2}
```

```
covMcd {robustbase}
scale {base}
prcomp {stats}
```

```
mclust {mclust}
```

Thank you

Any questions?