Chapter 8

January 12, 2021

1 Housekeeping:

[1]:
```
sessionInfo()
options(repr.plot.width=14,repr.plot.antialias='subpixel',repr.plot.res=218)
update.packages()
```

```
R version 4.0.3 Patched (2020-10-12 r79333)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.0.3   ellipsis_0.3.1   IRdisplay_0.7.0 pbdZMQ_0.3-3
 [5] tools_4.0.3      htmltools_0.5.0 pillar_1.4.6    base64enc_0.1-3
 [9] crayon_1.3.4     uuid_0.1-4      IRkernel_1.1.1  jsonlite_1.7.1
[13] digest_0.6.25    lifecycle_0.2.0 repr_1.1.0      rlang_0.4.8
[17] evaluate_0.14
```

2 Congressional records

From yet another Clark medalist (Gentzkow and Shapiro 2010).

[2]:
```
#install.packages("textir")
library(textir)
```

```
Loading required package: distrom

Loading required package: Matrix
```

```
Loading required package: gamlr

Loading required package: parallel
```

[3]:
```
data(congress109)
```

Explore the DTM:

[4]:
```
congress109Counts[c("Barack Obama","John Boehner"),995:998]
```

```
2 x 4 sparse Matrix of class "dgCMatrix"
             stem.cel natural.ga hurricane.katrina trade.agreement
Barack Obama        .          1                20               7
John Boehner        .          .                14               .
```

Prepopulated ideology measures use constituency vote shares and roll-call vote factors:

[5]:
```
congress109Ideology[1:4,]
```

A data.frame: 4 × 7

	name <chr>	party <fct>	state <fct>	chamber <fct>	repshare <dbl>	cs1 <dbl>	cs2 <dl
Chris Cannon	Chris Cannon	R	UT	H	0.7900621	0.534	-0.1
Michael Conaway	Michael Conaway	R	TX	H	0.7836028	0.484	0.05
Spencer Bachus	Spencer Bachus	R	AL	H	0.7812933	0.369	-0.0
Mac Thornberry	Mac Thornberry	R	TX	H	0.7776520	0.493	0.00

Build a slant index from marginal/partial regressions:

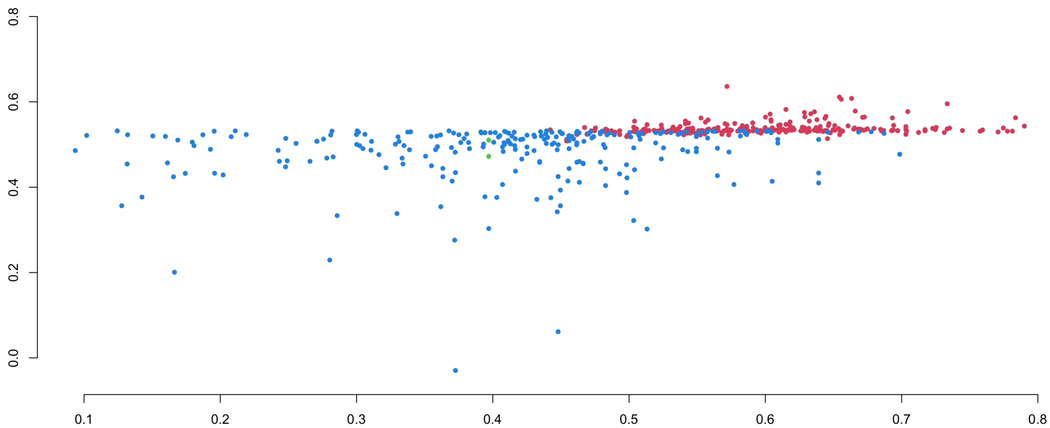
[6]:
```
f <- congress109Counts#t( t(congress109Counts)/rowSums(congress109Counts) )
y <- congress109Ideology$repshare
slant <- pls(f, y, K=3)
```

```
Directions 1, 2, 3, done.
```

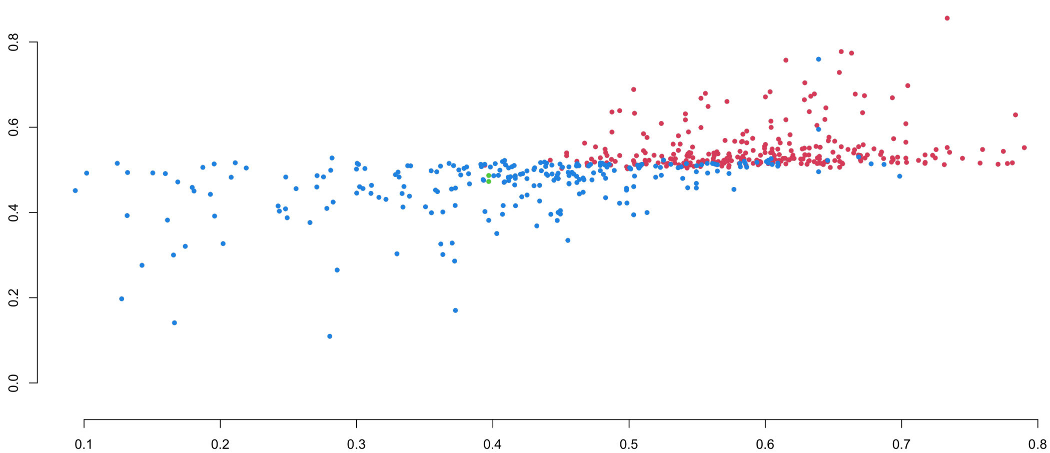
Plot (Figure 8.3):

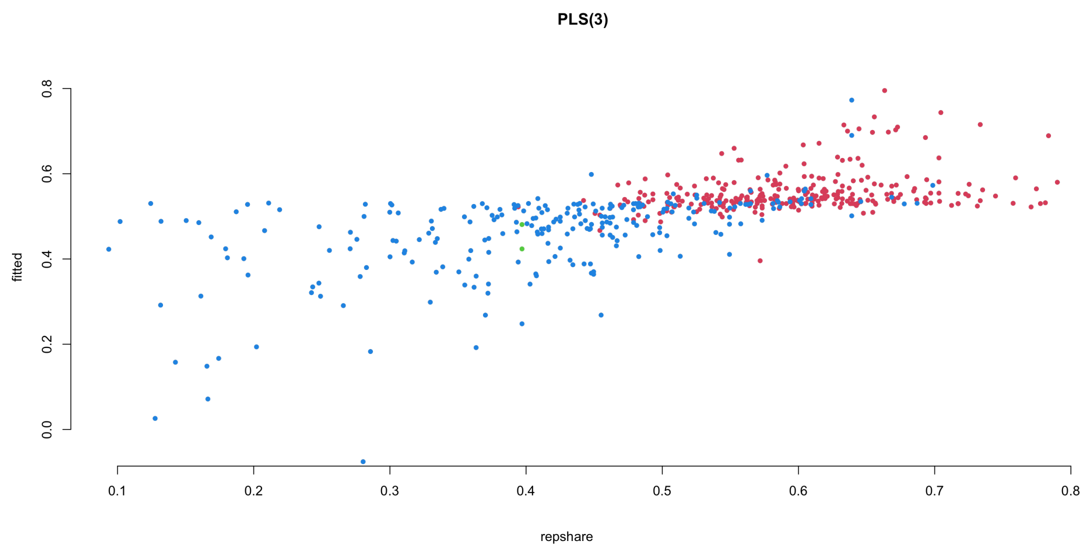
[7]:
```
for(k in 1:3)
        plot(slant$y, slant$fitted[,k], ylim=c(-.05,.85), xlab="", ylab="",
         main=sprintf("PLS(%d)", k),
                   pch=20, col=c(4,3,2)[congress109Ideology$party], bty="n")
mtext(side=1, "repshare", outer=TRUE, line=-1.25)
mtext(side=2, "fitted", outer=TRUE, line=-1.25)
```

PLS(1)


0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

PLS(2)


0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8



Collect out-of-sample deviance (mean squared error) from a five-fold cross-validation experiment (for each of 10 k's):

[8]:
```
foldid <- sample.int(5, nrow(f), replace=TRUE)
OOS <- matrix(nrow=5, ncol=10)
for(b in 1:5){
        print(b)
        fb <- f[,colSums(f[foldid!=b,])!=0]
        for(k in 1:10){
                gpls <- pls(x=fb[foldid!=b,], y=y[foldid!=b], K=k)
                OOS[b,k] <-
                        mean( (y[foldid==b] - predict(gpls, fb[foldid==b,],
 ↪K=k))^2 )
        }
}
cvm <- apply(OOS,2,mean)
cvs <- apply(OOS,2,sd)
OOS <- as.data.frame(OOS)
names(OOS) <- 1:10
```

```
[1] 1
Directions 1, done.
Directions 1, 2, done.
Directions 1, 2, 3, done.
Directions 1, 2, 3, 4, done.
Directions 1, 2, 3, 4, 5, done.
Directions 1, 2, 3, 4, 5, 6, done.
Directions 1, 2, 3, 4, 5, 6, 7, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, done.
```

```
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, done.
[1] 2
Directions 1, done.
Directions 1, 2, done.
Directions 1, 2, 3, done.
Directions 1, 2, 3, 4, done.
Directions 1, 2, 3, 4, 5, done.
Directions 1, 2, 3, 4, 5, 6, done.
Directions 1, 2, 3, 4, 5, 6, 7, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, done.
[1] 3
Directions 1, done.
Directions 1, 2, done.
Directions 1, 2, 3, done.
Directions 1, 2, 3, 4, done.
Directions 1, 2, 3, 4, 5, done.
Directions 1, 2, 3, 4, 5, 6, done.
Directions 1, 2, 3, 4, 5, 6, 7, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, done.
[1] 4
Directions 1, done.
Directions 1, 2, done.
Directions 1, 2, 3, done.
Directions 1, 2, 3, 4, done.
Directions 1, 2, 3, 4, 5, done.
Directions 1, 2, 3, 4, 5, 6, done.
Directions 1, 2, 3, 4, 5, 6, 7, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, done.
[1] 5
Directions 1, done.
Directions 1, 2, done.
Directions 1, 2, 3, done.
Directions 1, 2, 3, 4, done.
Directions 1, 2, 3, 4, 5, done.
Directions 1, 2, 3, 4, 5, 6, done.
Directions 1, 2, 3, 4, 5, 6, 7, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, done.
Directions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, done.
```

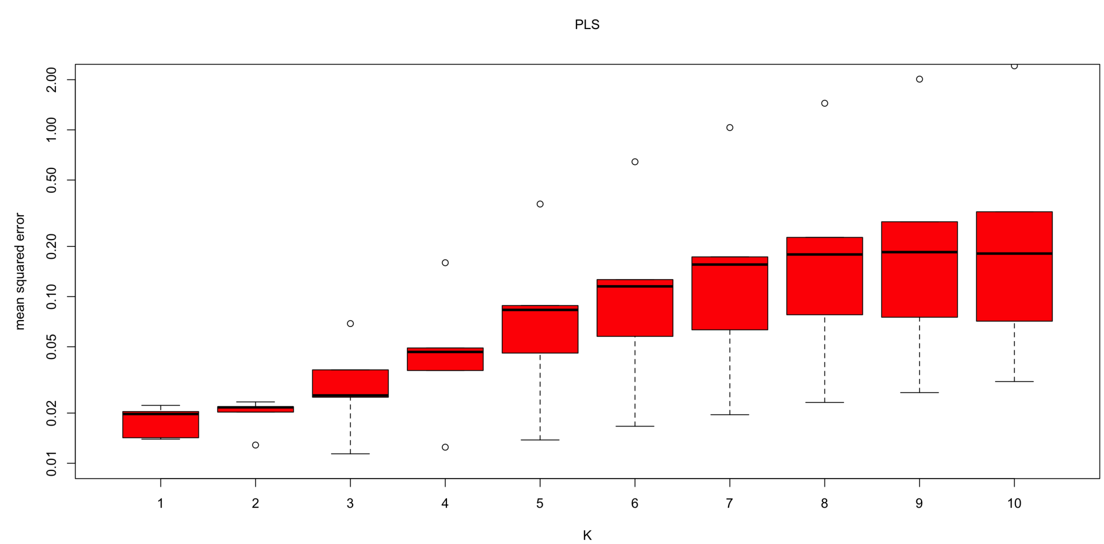
We can also try a lasso instead of PLS:

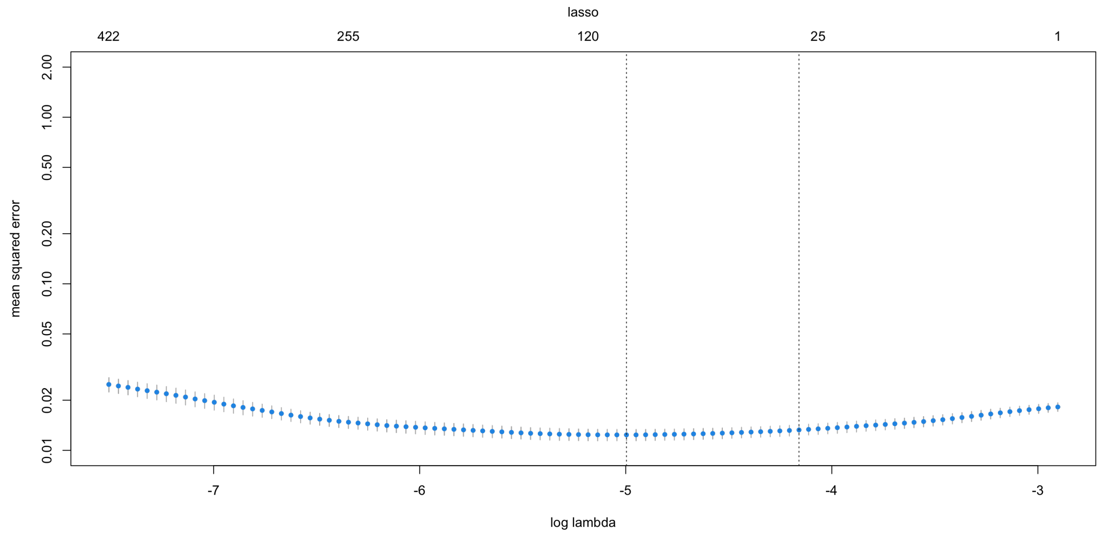
```
[9]: lassoslant <- cv.gamlr(congress109Counts>0, y)
B <- coef(lassoslant$gamlr)[-1,]
sort(round(B[B!=0],4))
```

congressional.black.caucu -0.0839 **family.value** -0.0443 **issue.facing.american** -0.0324 **voter.registration** -0.0298 **minority.owned.business** -0.0284 **strong.opposition** -0.0264 **civil.right** -0.0259 **universal.health.care** -0.0254 **congressional.hispanic.caucu** -0.0187 **ohio.electoral.vote** -0.0183 **american.community** -0.0181 **public.broadcasting** -0.0153 **republican.party** -0.0138 **tax.break** -0.0138 **voting.machine** -0.0134 **pay.tax.cut** -0.0114 **american.heritage.month** -0.0095 **war.iraq** -0.009 **tuskege.airmen** -0.0088 **million.american** -0.0088 **tax.cut.benefit** -0.0084 **bil.cut** -0.0083 **feder.emergency.management** -0.007 **middle.class** -0.0066 **international.labor.standard** -0.0063 **medic.malpractice** -0.0061 **percent.african.american** -0.0056 **malpractice.insurance** -0.0055 **oil.ga.compani** -0.0019 **bil.fal.short** -0.0015 **provisional.ballot** -6e-04 **armenian.genocide** -1e-04 **energy.natural.resource** 6e-04 **repeal.death.tax** 0.0014 **washington.dc** 0.0015 **time.move** 0.0018 **time.vote** 0.0019 **food.program** 0.0027 **committe.foreign.relation** 0.003 **highway.bil** 0.003 **reform.class.action** 0.0037 **pass.bil** 0.0038 **natural.ga** 0.0039 **district.judge** 0.0045 **private.property.owner** 0.0046 **class.action.fairness** 0.0047 **feder.spending** 0.0049 **serving.country** 0.0053 **economy.growing** 0.0062 **death.tax** 0.0066 **protect.private.property** 0.0071 **farm.bureau** 0.0073 **raise.taxe** 0.0078 **class.action** 0.0078 **illegal.alien** 0.0079 **percent.growth** 0.0083 **illegal.immigration** 0.0087 **global.war** 0.0098 **look.forward** 0.0099 **war.terror** 0.0114 **private.property** 0.0133 **action.lawsuit** 0.0142 **human.embryo** 0.0226 **million.illegal.alien** 0.0328

Voila, compare:

```
[10]: boxplot(OOS, ylab="mean squared error", xlab="K", col="red", log="y", main="",
        ylim=c(0.01,2))
mtext(side=3, "PLS", line=2)
plot(lassoslant, log="y", main="", ylim=c(0.01,2))
mtext(side=3, "lasso", line=2)
```





3 Topic models

Data comes from an older review site, `we8there`:

[11]:
```
data(we8there)
```

DTM:

[12]:
```
x <- we8thereCounts
x[1,x[1,]!=0]
```

even though 1 **larg portion** 1 **mouth water** 1 **red sauc** 1 **babi back** 1 **back rib** 1 **chocol mouss** 1 **veri satisfi** 1

Let's run a PCA (wait patiently) and grab four factors:

[13]:
```
pca <- prcomp(x, scale=TRUE)
v <- predict(pca)[,1:4]
```

Experiment with a bottom-up look:

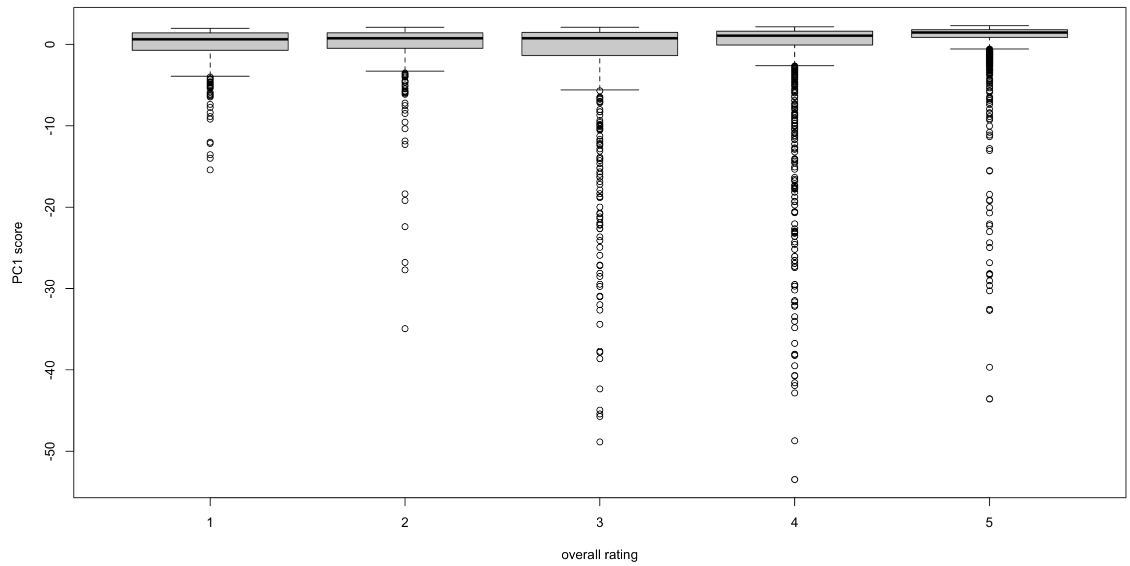
[14]:
```
tail(sort(pca$rotation[,1]))
head(sort(pca$rotation[,4]))
```

food great 0.00738685978714776 **staff veri** 0.00759337395130864 **excel food** 0.00762977062977721 **high recommend** 0.00782117134699393 **great food** 0.00850359379273913 **food excel** 0.00873618082316544

pizza like -0.179416634080406 **thin crust** -0.170530060756619 **thin crispi** -0.155187689201611 **deep dish** -0.153182046483436 **crust pizza** -0.131116067021328 **italian beef** -0.125010372685448

Figure 8.5 plots the estimated PC1 score against the raw ratings in the data for a top-down view:

[15]:
```
par(mai=c(.8,.8,.1,.1))
boxplot(v[,1] ~ we8thereRatings$Overall, xlab="overall rating", ylab="PC1 score")
```



The topics function will come from yet another library, and actually one Taddy removed from the standard repository. To stick to the book's notation, let's not replace it with our own functions, instead load his original package from his GitHub site.

[16]:
```
#install.packages("devtools")
#devtools::install_github("r-lib/devtools")
#library(devtools)
#install_github("TaddyLab/maptpx")
```

[17]:
```
library(maptpx)
```

```
Loading required package: slam
```

This is a bit tedious, but we'll need to convert the data from a `Matrix` to a `slam` `simple_triplet_matrix`.

[18]:
```
x <- as.simple_triplet_matrix(we8thereCounts)
```

To fit, just give `topics` the counts, number of topics K, and any other arguments.

[19]:
```
tpc <- topics(x,K=10)
```

```
Estimating on a 6166 document collection.
```

```
Fitting the 10 topic model.
log posterior increase: 4444.6, 462.7, 101.1, 40.2, 41.4, 56.5, 25.8, 15.4,
17.3, 10, 7.5, 11.1, 8.6, 8.1, 5, 3.2, 9.7, 7.9, 3.3, 7.4, 5.3, 3.5, 3.5, 5.4,
9, 5.2, 8.5, 11.3, 10.2, 11.9, 21.2, 9.7, 8.3, 15.9, 8.2, 5.6, 3.3, 8.3, 6.8,
7.6, 14.5, 7.7, 4.2, 6.2, 2.9, 2.2, 3.3, 2, 3.4, 1.7, 3.8, 5.2, 1.4, 1.9, 1.7,
2, 2.9, 1.9, 2, 1.1, 0.9, 1.2, 0.8, 12.5, 0.8, 2, 2.2, 1.9, 1.5, 0.2, 0.2, 0.4,
0.2, done.
```

3.1 Choosing the number of topics

If you supply a vector of topic sizes, the command uses a Bayes factor to choose the best-fitting size. (BF is like exp(-BIC), so we choose the biggest BF.) The algorithm stops if BF drops twice in a row.

[20]:
```
tpcs <- topics(x,K=5*(1:5), verb=1) # it chooses 10 topics
```

```
Estimating on a 6166 document collection.
Fit and Bayes Factor Estimation for K = 5 ... 25
log posterior increase: 2854.4, 321.3, 85, 37.3, 84, 21.7, 18.5, 7, 17.3, 5.7,
4.4, 6.5, 5.5, 12, 7, 3.6, 2.7, 2.7, 3.7, 6.3, 3.9, 3.1, 3, 2.3, 3.7, 4.1, 2.7,
3.3, 1.4, 1.9, 2.1, 6.8, 16, 5, 15.2, 7.5, 9.7, 12.6, 4.4, 5.8, 5.4, 4.2, 5.3,
4.4, 7.9, 10.8, 24.2, 15.7, 12.4, 12.9, 11.6, 13.4, 6.9, 6.1, 5.1, 4.3, 2.2,
2.7, 1.3, 2, 1.3, 3.5, 2.2, 1.3, 3, 1.9, 0.9, 1.9, 1.4, 1, 0.9, 0.8, 0.8, 0.3,
0.5, 0.4, 0.5, 0.2, 0.2, 0.3, 0.2, 2.4, 0.8, 1.2, 0.2, 0.2, 0.1, done.
log BF( 5 ) = 79606.54
log posterior increase: 4626.2, 199.7, 54.4, 25.5, 15.8, 13.5, 6.1, 5.8, 5.8,
3.6, 4.2, 5.4, 2, 1.7, 3.4, 0.9, 0.7, 0.6, 0.7, 0.3, 1.6, 0.5, 0.5, 1.1, 0.4,
0.1, 0.4, 1.1, 1.1, 0.3, 0.4, 1.2, 0.4, 0.3, 0.1, 0.3, done.
log BF( 10 ) = 87552.6
log posterior increase: 3445.7, 176.6, 55.4, 23.1, 12, 9.8, 6.7, 7, 6.3, 2.8,
1.7, 1.7, 4, 3.1, 1.4, 0.7, 2.4, 0.9, 0.7, 0.8, 0.3, 1.6, 0.6, 0.3, 0.5, 0.3,
0.2, 0.8, 0.2, 0.3, 0.3, 0.2, done.
log BF( 15 ) = 4182.53
log posterior increase: 2319.8, 134.2, 33.6, 16, 21.2, 5.4, 1.9, 4.7, 4.9, 2.6,
2.4, 0.8, 2.3, 0.6, 0.8, 1.2, 1, 2, 0.7, 0.9, 0.3, 0.9, 0.6, 0.4, 0.2, 0.6, 0.3,
0.5, 0.2, 4.2, 0.3, 0.9, 0.5, 0.2, done.
log BF( 20 ) = -66477.85
```

3.2 Interpretation

summary prints the top n words for each topic, under ordering by 'topic over aggregate' lift: the topic word probability over marginal word probability.

[21]:
```
summary(tpcs, n=10)
```

```
Top 10 phrases by topic-over-null term lift (and usage %):
```

```
[1] 'food great', 'great food', 'veri good', 'great servic', 'food veri', 'food
excel', 'veri nice', 'veri friend', 'excel food', 'excel servic' (14.3)
[2] 'best italian', 'high recommend', 'italian food', 'list extens', 'staff
friend', 'mexican food', 'wait staff', 'don miss', 'food wonder', 'authent
mexican' (11.7)
[3] 'open daili', 'until pm', 'pm friday', 'monday through', 'enough share',
'select includ', 'fresh veget', 'dinner buffet', 'dine spot', 'highlight menu'
(10.5)
[4] 'over minut', 'arriv after', 'never go', 'go back', 'flag down', 'anoth
minut', 'ask manag', 'wait minut', 'after minut', 'wait anoth' (10.1)
[5] 'don wast', 'never return', 'wast time', 'one worst', 'here sever', 'sever
time', 'tourist trap', 'small portion', 'time money', 'veri bland' (9.6)
[6] 'take out', 'wait go', 'can get', 'can wait', 'everi time', 'well worth',
'worth wait', 'sushi chef', 'sushi bar', 'home cook' (9.4)
[7] 'enjoy dine', 'veri pleasant', 'thai food', 'thai restaur', 'year ago',
'indian food', 'first time', 'best meal', 'breakfast lunch', 'pleasant experi'
(9)
[8] 'good work', 'great experi', 'out world', 'just right', 'best kept', 'prime
rib', 'live music', 'kept secret', 'real treat', 'again again' (8.9)
[9] 'food good', 'food place', 'good select', 'chees steak', 'food pretti',
'place hang', 'don expect', 'price littl', 'portion size', 'nice littl' (8.6)
[10] 'wasn whole', 'came chip', 'got littl', 'took seat', 'over drink', 'came
pile', 'toast bun', 'fri out', 'east lans', 'bill just' (7.8)

Log Bayes factor and estimated dispersion, by number of topics:

             5        10       15        20
logBF 79606.54 87552.60 4182.53 -66477.85
Disp      7.10     4.97     3.94      3.34

Selected the K = 10 topic model
```

This will promote rare words that with high in-topic probability.

Alternatively, we can look at words ordered by simple in-topic probability. The topic-term probability matrix is called 'theta', and each column is a topic. We can use these to rank terms by probability within topics.

[22]:
```
rownames(tpcs$theta)[order(tpcs$theta[,1], decreasing=TRUE)[1:10]]
rownames(tpcs$theta)[order(tpcs$theta[,2], decreasing=TRUE)[1:10]]
rownames(tpcs$theta)[order(tpcs$theta[,3], decreasing=TRUE)[1:10]]
```

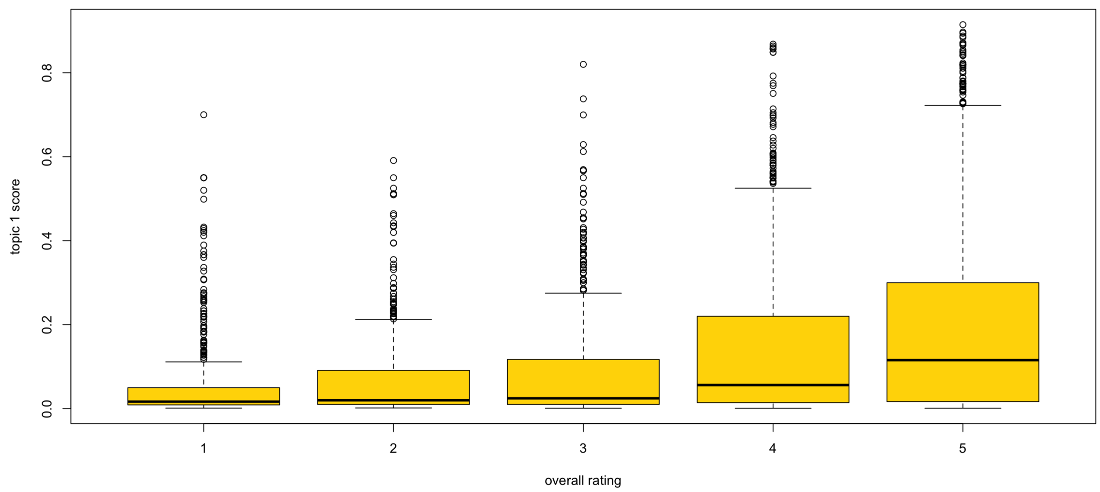
1. 'veri good' 2. 'great food' 3. 'food great' 4. 'great place' 5. 'veri friend' 6. 'veri nice' 7. 'good food' 8. 'great servic' 9. 'food excel' 10. 'servic great'

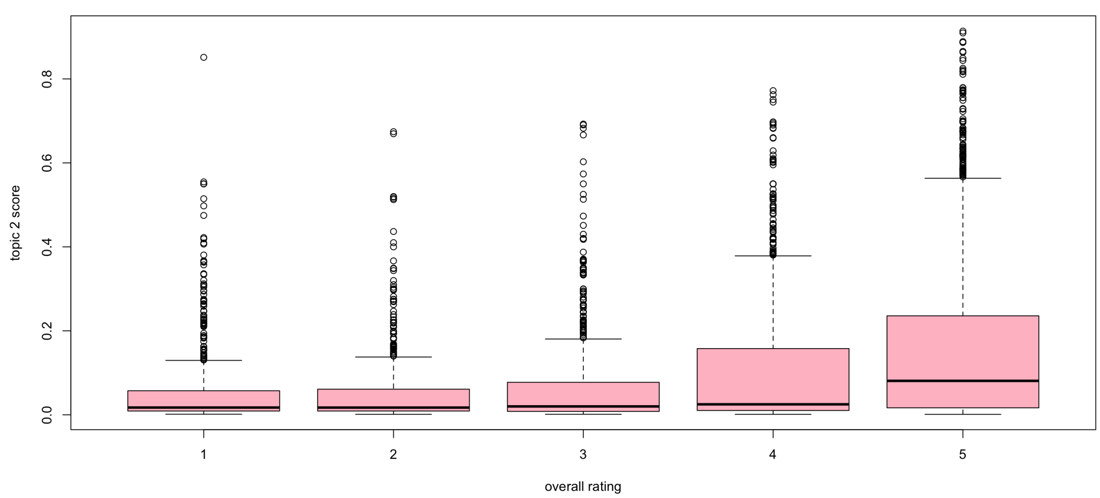
1. 'high recommend' 2. 'dine experi' 3. 'wait staff' 4. 'wine list' 5. 'one best' 6. 'mexican food' 7. 'italian food' 8. 'italian restaur' 9. 'staff friend' 10. 'make feel'

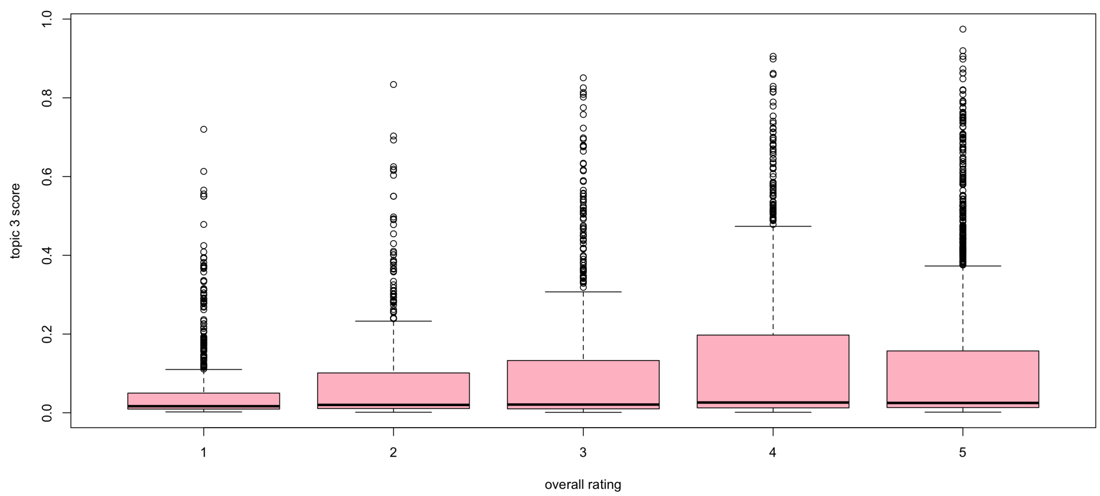
1. 'san francisco' 2. 'salad bar' 3. 'mash potato' 4. 'crab cake' 5. 'restaur locat' 6. 'main cours'

7. 'lunch dinner' 8. 'wide varieti' 9. 'soup salad' 10. 'reason price'

[23]:
```
boxplot(tpcs$omega[,1] ~ we8thereRatings$Overall, col="gold", xlab="overall␣
↪rating", ylab="topic 1 score")
boxplot(tpcs$omega[,2] ~ we8thereRatings$Overall, col="pink", xlab="overall␣
↪rating", ylab="topic 2 score")
boxplot(tpcs$omega[,3] ~ we8thereRatings$Overall, col="pink", xlab="overall␣
↪rating", ylab="topic 3 score")
```







Interpret the relationship between topics and overall rating:

omega is the n x K matrix of document topic weights, i.e., how much of each document is from each topic. We'll regress overall ratings onto it.

[24]:
```
library(gamlr)
stars <- we8thereRatings[,"Overall"]
tpcreg <- gamlr(tpcs$omega, stars, lmr=1e-3)
```

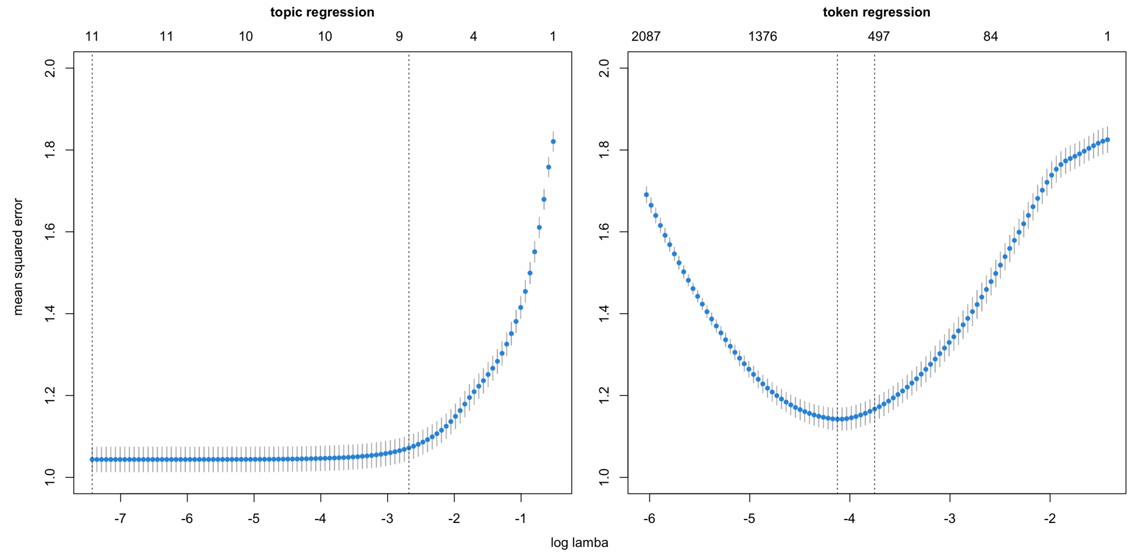
Calculate the number of stars more or less for moving up 10% weight in that topic:

[25]:
```
round(coef(tpcreg)[-1,]*0.1,1)
```

1 0.1 **2** 0.1 **3** 0 **4** -0.3 **5** -0.4 **6** 0 **7** 0.1 **8** 0.1 **9** -0.1 **10** -0.1

[26]:
```
regtopics.cv <- cv.gamlr(tpcs$omega, stars, lmr=1e-3)
regwords.cv <- cv.gamlr(we8thereCounts, stars)
```

[27]:
```
par(mfrow=c(1,2), mai=c(.3,.6,.7,.1), omi=c(.5,.2,0,0))
plot(regtopics.cv, ylim=c(1,2), xlab="", ylab="")
mtext("topic regression", font=2, line=2)
plot(regwords.cv, ylim=c(1,2), xlab="", ylab="")
mtext("token regression", font=2, line=2)
mtext(side=2, "mean squared error", outer=TRUE, line=0)
mtext(side=1, "log lamba", outer=TRUE, line=1)
```



Maximal out-of-sample R2s for each path:

[28]:
```
max(1-regtopics.cv$cvm/regtopics.cv$cvm[1])
max(1-regwords.cv$cvm/regwords.cv$cvm[1])
```

0.426778369854959

0.374185426821656

4 Multinomial text Regression

This could get (computationally) intense, so we can start with setting up parallelization as much as possible. `cl=NULL` would instead imply a serial run.

[29]:
```
cl <- makeCluster(detectCores())
```

Here we pick a small `nlambda` for a fast example. You could pick a larger one if you had time.

[30]:
```
fits <- dmr(cl, we8thereRatings,
                        we8thereCounts, bins=5,nlambda=10, lmr=1e-3)
```

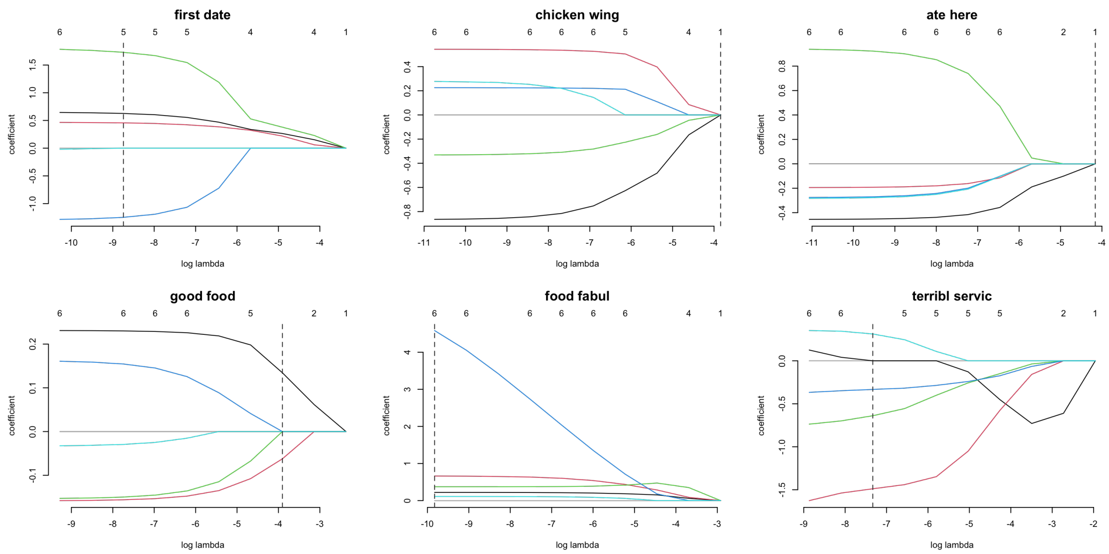
Small thing, but note that it is often prudent to shut down clusters you set up.

[31]:
```
stopCluster(cl)
```

We can plot the fit for a few individual terms:

[32]:
```
terms <- c("first date","chicken wing",
                        "ate here", "good food",
                        "food fabul","terribl servic")
```

```
par(mfrow=c(2,3))
for(j in terms)
{        plot(fits[[j]]); mtext(j,font=2,line=2) }
```



And extract coefficients:

[33]:
```
B <- coef(fits)
B[,1:5]
```

```
6 x 5 sparse Matrix of class "dgCMatrix"
             veri good    go back  dine room dine experi  great food
intercept  -5.14953418 -4.7247330 -6.0361782   -5.555694 -9.69426558
Food        0.18245562  .          0.1221376    .          0.38892467
Service     0.01318399  .         -0.1920983    .          0.04234122
Value      -0.15418467 -0.1013329  .            .          0.17131793
Atmosphere -0.24317814  .          0.2205456    .          .
Overall     0.24107687  .          .            .          0.34407390
```

See how sparse the loadings get:

[34]:
```
mean(B[-1,]==0)
```

0.69530303030303

Check out some of the biggest loadings on `overall`:

[35]:
```
B[2,order(B[2,])[1:10]]
B[2,order(-B[2,])[1:10]]
```

food poison -1.65771686627984 **hot food** -1.29037357246786 **old town** -1.28290999276968 **food bland** -1.27165565984116 **first date** -1.24761369670255 **chicken pork** -1.13238784515442 **more**

flavor -1.11148292631045 **qualiti ingredi** -1.10916461858124 **fri calamari** -1.10430424821932 **one worst** -1.09685994144327

cannot wait 5.3939478620321 **food fabul** 4.58459058559232 **food superb** 4.55972419141395 **best sushi** 2.38094116932909 **food awesom** 2.16189769932406 **outstand servic** 1.98341823219872 **around world** 1.95725136067638 **francisco bay** 1.91176161140464 **mouth water** 1.88886028095408 **best kept** 1.78268649713592

We can do a MNIR projection onto factors:

[36]:
```
z <- srproj(B,we8thereCounts)
z[1:5,]
```

A matrix: 5 × 6 of type dbl

	Food	Service	Value	Atmosphere	Overall	m
1	0.46365320	0.27274793	0.39579271	-0.05736961	-0.23398473	8
2	0.00000000	-0.11664138	0.00000000	0.00000000	-0.16916446	2
5	0.00000000	-0.15458926	0.15263267	0.11693316	0.70421596	2
11	-0.03397961	0.09429418	0.08080977	0.03001613	0.21227263	11
12	0.73388490	0.09232618	0.10932085	0.03084464	0.03271238	8

Then fit a forward model to the factors:

[37]:
```
summary(fwd <- lm(we8thereRatings$Overall ~ z))
```

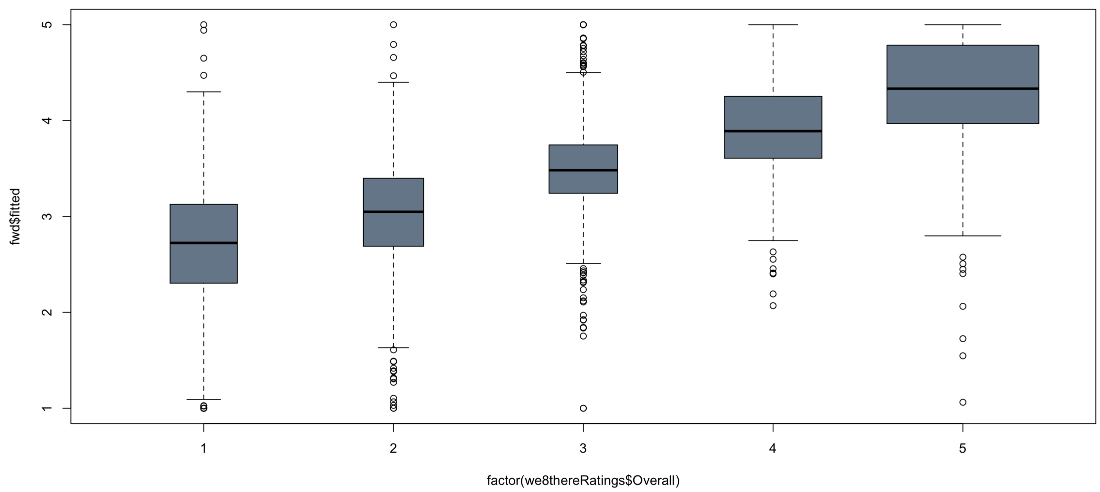
```
Call:
lm(formula = we8thereRatings$Overall ~ z)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7239 -0.5929  0.1663  0.7257  3.9374

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.421197   0.020610 165.996  < 2e-16 ***
zFood       1.822132   0.078650  23.168  < 2e-16 ***
zService    3.026174   0.097451  31.053  < 2e-16 ***
zValue      2.537348   0.106949  23.725  < 2e-16 ***
zAtmosphere 1.026889   0.105121   9.769  < 2e-16 ***
zOverall    3.391911   0.083258  40.740  < 2e-16 ***
zm          0.006308   0.001149   5.491 4.16e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.003 on 6159 degrees of freedom
Multiple R-squared:  0.4495,	Adjusted R-squared:  0.449
F-statistic: 838.2 on 6 and 6159 DF,  p-value: < 2.2e-16
```

It makes sense to truncate the forward predictions to our known range:

[38]:
```
fwd$fitted[fwd$fitted<1] <- 1
fwd$fitted[fwd$fitted>5] <- 5
```

And now we can plot the fitted rating by the true rating:

[39]:
```
plot(fwd$fitted ~ factor(we8thereRatings$Overall),
        varwidth=TRUE, col="lightslategrey")
```



5 Collaborative Filtering

For this we use LastFM play counts

[40]:
```
lastfm <- read.csv("lastfm.csv")
lastfm$user <- factor(lastfm$user)
lastfm$artist <- factor(lastfm$artist)
lastfm$sex <- factor(lastfm$sex)
lastfm$country <- factor(lastfm$country)
head(lastfm)
```

A data.frame: 6 × 4

	user	artist	sex	country
	<fct>	<fct>	<fct>	<fct>
1	1	red hot chili peppers	f	Germany
2	1	the black dahlia murder	f	Germany
3	1	goldfrapp	f	Germany
4	1	dropkick murphys	f	Germany
5	1	le tigre	f	Germany
6	1	schandmaul	f	Germany

Doing `topics` first (unlike the book):

[41]:
```
x <- simple_triplet_matrix(i=as.numeric(lastfm$user),
                j=as.numeric(lastfm$artist), v=rep(1,nrow(lastfm)),
                nrow = nlevels(lastfm$user), ncol = nlevels(lastfm$artist),
        dimnames = list(levels(lastfm$user), levels(lastfm$artist)))

tpcs <- topics(x, K=5*(1:5), verb=1)
```

```
Estimating on a 15000 document collection.
Fit and Bayes Factor Estimation for K = 5 ... 25
log posterior increase: 10390.3, 2106.5, 905.9, 409.9, 167.7, 83.7, 30.3, 24.1,
12, 12.3, 13.4, 6.2, 6.3, 8.7, 9, 13, 1.6, 7.4, 0.7, 0.7, 0.4, 0.3, 0.1, done.
log BF( 5 ) = 343193.27
log posterior increase: 12081.7, 1655.4, 350.4, 153.4, 110, 57.5, 57.4, 13.3,
13.6, 20.8, 23.1, 33.1, 3.1, 1.5, 3.2, 6.6, 2.1, 0.7, 0.8, 0.7, 0.9, 0.8, 0.9,
0.9, 0.9, 1.3, 1.2, 7.9, 2.9, 2.2, 6.3, 3.8, 17.7, 48.7, 52.1, 33.6, 27.3, 22.8,
9.8, 5, 7, 3.3, 3.6, 1.2, 4.6, 0.9, 0.5, 1.1, 0.1, 0.3, 0.2, 0.1, 0.2, 0.3,
done.
log BF( 10 ) = 523693.26
log posterior increase: 6018.3, 687.5, 161.3, 136.1, 56, 46.2, 28.9, 13.6, 34.2,
30.8, 30.5, 56.7, 50.7, 34.4, 16.3, 79.1, 36.6, 22.2, 27.7, 22.3, 11.7, 7.2,
7.8, 16.5, 18.7, 5.7, 2.3, 3.4, 2.5, 1.9, 6.1, 43.3, 3.4, 1.9, 2.4, 1, 1.2, 1.5,
2.7, 3, 0.4, 0.2, 0.3, 0.5, 0.6, 0.2, 0.2, 0.3, 0.6, 0.4, 0.5, 0.2, done.
log BF( 15 ) = 497743.73
log posterior increase: 2966.5, 689.1, 191, 63.5, 28.3, 21.4, 18.4, 11.6, 2.5,
10.8, 6.3, 9.7, 5.5, 4.4, 12.4, 6.2, 3.8, 3.5, 10.9, 11.2, 13, 9.1, 31.4, 16.7,
12.5, 7.3, 5.7, 5.3, 14, 5.6, 6.2, 4.7, 3.1, 5, 6, 2.3, 1.8, 3.2, 5.3, 11.3, 5,
12.7, 7.3, 6.2, 1.5, 0.8, 1, 2.3, 4.7, 10.7, 9.8, 1.6, 0.6, 1.9, 1.6, 4.4, 0.9,
1.9, 3.9, 8.6, 2.7, 0.8, 4.8, 0.8, 2.2, 1.5, 0.4, 0.7, 3.6, 5.9, 1.4, 0.4, 0.2,
0.1, 1.3, 0.8, 0.4, 0.1, done.
log BF( 20 ) = 449439.24
```

[42]:
```
summary(tpcs)
```

```
Top 5 phrases by topic-over-null term lift (and usage %):

[1] 'guided by voices', 'animal collective', 'pavement', 'the magnetic fields',
'andrew bird' (13.1)
[2] 'the rolling stones', 'the who', 'eric clapton', 'cream', 'john lennon'
(12.4)
[3] 'hypocrisy', 'turisas', 'equilibrium', 'sodom', 'norther' (12.3)
[4] 'leona lewis', 'jennifer lopez', 'the pussycat dolls', 'céline dion',
'christina aguilera' (11.3)
[5] 'tosca', 'the future sound of london', 'plaid', 'amon tobin', 'massive
attack' (10.9)
[6] 'nirvana', 'a perfect circle', 'soundgarden', 'audioslave', 'stone temple
pilots' (8.9)
```

```
[7] 'the pigeon detectives', 'franz ferdinand', 'arctic monkeys', 'the kooks',
'kaiser chiefs' (8.9)
[8] 'red.jumpsuit.apparatus', 'simple plan', 'good charlotte', 'skillet', '30
seconds to mars' (8.4)
[9] 'comeback kid', 'the bouncing souls', 'rancid', 'parkway drive',
'descendents' (6.9)
[10] 'the game', 'talib kweli', 'mobb deep', 'nas', 'notorious b.i.g.' (6.8)

Log Bayes factor and estimated dispersion, by number of topics:

              5        10        15        20
logBF 343193.27 523693.26 497743.73 449439.24
Disp       1.43      1.29      1.18      1.12

Selected the K = 10 topic model
```

The old-school marketing way would be to use the `a-rules` package for association rules

[43]:
```
#install.packages("arules")
library(arules)
```

```
Attaching package: ‘arules’


The following objects are masked from ‘package:base’:

    abbreviate, write
```

In any case, there is an entire ecosystem of packages around association rules. But you need to first create a list of baskets: vectors of items by consumer.

Here's how we do the formatting here: split data into a list of artists for each user

[44]:
```
playlists <- split(x=lastfm$artist, f=lastfm$user)
```

Remove artist repetition in these lists

[45]:
```
playlists <- lapply(playlists, unique)
```

Now tell R to treat this as a special arules `transactions` class.

[46]:
```
playtrans <- as(playlists, "transactions")
```

Now we can apply the actual `apriori` algorithm. We can add a list of arguments called `parameter`. Here, we look at only rules with support > .01 & confidence >.5 & length (# artists) <= 3

[47]:
```
musicrules <- apriori(playtrans,
        parameter=list(support=.01, confidence=.5, maxlen=3))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.5    0.1    1 none FALSE            TRUE       5    0.01      1
 maxlen target  ext
      3  rules TRUE

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

Absolute minimum support count: 150

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[1004 item(s), 15000 transaction(s)] done [0.04s].
sorting and recoding items ... [655 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3
```

```
Warning message in apriori(playtrans, parameter = list(support = 0.01,
confidence = 0.5, :
“Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!”
```

```
 done [0.02s].
writing ... [50 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Now we can check out the rules:

[48]:
```
inspect(musicrules)
```

```
     lhs                                     rhs             support
[1]  {t.i.}                               => {kanye west}    0.01040000
[2]  {the pussycat dolls}                 => {rihanna}       0.01040000
[3]  {the fray}                           => {coldplay}      0.01126667
[4]  {sonata arctica}                     => {nightwish}     0.01346667
[5]  {judas priest}                       => {iron maiden}   0.01353333
[6]  {the kinks}                          => {the beatles}   0.01360000
[7]  {travis}                             => {coldplay}      0.01373333
[8]  {the flaming lips}                   => {radiohead}     0.01306667
[9]  {megadeth}                           => {metallica}     0.01626667
[10] {simon & garfunkel}                  => {the beatles}   0.01540000
[11] {broken social scene}                => {radiohead}     0.01506667
[12] {blur}                               => {radiohead}     0.01753333
[13] {keane}                              => {coldplay}      0.02226667
```

```
[14] {snow patrol}                         => {coldplay}     0.02646667
[15] {beck}                                => {radiohead}    0.02926667
[16] {snow patrol,the killers}             => {coldplay}     0.01040000
[17] {radiohead,snow patrol}               => {coldplay}     0.01006667
[18] {death cab for cutie,the shins}       => {radiohead}    0.01006667
[19] {the beatles,the shins}               => {radiohead}    0.01066667
[20] {led zeppelin,the doors}              => {pink floyd}   0.01066667
[21] {pink floyd,the doors}                => {led zeppelin} 0.01066667
[22] {pink floyd,the doors}                => {the beatles}  0.01000000
[23] {the beatles,the strokes}             => {radiohead}    0.01046667
[24] {oasis,the killers}                   => {coldplay}     0.01113333
[25] {oasis,the beatles}                   => {coldplay}     0.01060000
[26] {oasis,radiohead}                     => {coldplay}     0.01273333
[27] {beck,the beatles}                    => {radiohead}    0.01300000
[28] {bob dylan,the rolling stones}        => {the beatles}  0.01146667
[29] {david bowie,the rolling stones}      => {the beatles}  0.01000000
[30] {led zeppelin,the rolling stones}     => {the beatles}  0.01066667
[31] {radiohead,the rolling stones}        => {the beatles}  0.01060000
[32] {coldplay,the smashing pumpkins}      => {radiohead}    0.01093333
[33] {the beatles,the smashing pumpkins}   => {radiohead}    0.01146667
[34] {radiohead,u2}                        => {coldplay}     0.01140000
[35] {coldplay,sigur rós}                  => {radiohead}    0.01206667
[36] {sigur rós,the beatles}               => {radiohead}    0.01046667
[37] {bob dylan,pink floyd}                => {the beatles}  0.01033333
[38] {bob dylan,radiohead}                 => {the beatles}  0.01386667
[39] {bloc party,the killers}              => {coldplay}     0.01106667
[40] {david bowie,pink floyd}              => {the beatles}  0.01006667
[41] {david bowie,radiohead}               => {the beatles}  0.01393333
[42] {placebo,radiohead}                   => {muse}         0.01366667
[43] {led zeppelin,radiohead}              => {the beatles}  0.01306667
[44] {death cab for cutie,the killers}     => {coldplay}     0.01086667
[45] {death cab for cutie,the beatles}     => {radiohead}    0.01246667
[46] {muse,the killers}                    => {coldplay}     0.01513333
[47] {red hot chili peppers,the killers}   => {coldplay}     0.01086667
[48] {the beatles,the killers}             => {coldplay}     0.01253333
[49] {radiohead,the killers}               => {coldplay}     0.01506667
[50] {muse,the beatles}                    => {radiohead}    0.01380000
     confidence coverage   lift      count
[1]  0.5672727  0.01833333  8.854413 156
[2]  0.5777778  0.01800000 13.415893 156
[3]  0.5168196  0.02180000  3.260006 169
[4]  0.5101010  0.02640000  8.236292 202
[5]  0.5075000  0.02666667  8.562992 203
[6]  0.5298701  0.02566667  2.979030 204
[7]  0.5628415  0.02440000  3.550304 206
[8]  0.5297297  0.02466667  2.938589 196
[9]  0.5281385  0.03080000  4.743759 244
[10] 0.5238095  0.02940000  2.944956 231
```

```
[11] 0.5472155  0.02753333  3.035589 226
[12] 0.5228628  0.03353333  2.900496 263
[13] 0.6374046  0.03493333  4.020634 334
[14] 0.5251323  0.05040000  3.312441 397
[15] 0.5092807  0.05746667  2.825152 439
[16] 0.5954198  0.01746667  3.755802 156
[17] 0.6344538  0.01586667  4.002021 151
[18] 0.5033333  0.02000000  2.792160 151
[19] 0.5673759  0.01880000  3.147425 160
[20] 0.5970149  0.01786667  5.689469 160
[21] 0.5387205  0.01980000  6.802027 160
[22] 0.5050505  0.01980000  2.839489 150
[23] 0.5607143  0.01866667  3.110471 157
[24] 0.6626984  0.01680000  4.180183 167
[25] 0.5196078  0.02040000  3.277594 159
[26] 0.5876923  0.02166667  3.707058 191
[27] 0.5909091  0.02200000  3.277972 195
[28] 0.5910653  0.01940000  3.323081 172
[29] 0.5703422  0.01753333  3.206572 150
[30] 0.5776173  0.01846667  3.247474 160
[31] 0.5638298  0.01880000  3.169958 159
[32] 0.6283525  0.01740000  3.485683 164
[33] 0.6209386  0.01846667  3.444556 172
[34] 0.5213415  0.02186667  3.288529 171
[35] 0.5801282  0.02080000  3.218167 181
[36] 0.6434426  0.01626667  3.569393 157
[37] 0.6150794  0.01680000  3.458092 155
[38] 0.5730028  0.02420000  3.221530 208
[39] 0.5236593  0.02113333  3.303150 166
[40] 0.5741445  0.01753333  3.227949 151
[41] 0.5225000  0.02666667  2.937594 209
[42] 0.5137845  0.02660000  4.504247 205
[43] 0.5283019  0.02473333  2.970213 196
[44] 0.5884477  0.01846667  3.711823 163
[45] 0.5013405  0.02486667  2.781105 187
[46] 0.5089686  0.02973333  3.210483 227
[47] 0.5093750  0.02133333  3.213047 163
[48] 0.5340909  0.02346667  3.368950 188
[49] 0.5243619  0.02873333  3.307582 226
[50] 0.5073529  0.02720000  2.814458 207
```

And also investigate any subset we want.

[49]:
```
inspect(subset(musicrules, subset=lift > 5))
inspect(subset(musicrules, subset=confidence > 0.6))
inspect(subset(musicrules, subset=support > .02 & confidence > 0.6))
inspect(subset(musicrules, subset=lhs%in%"t.i."))
```

```
    lhs                            rhs            support    confidence coverage
[1] {t.i.}                      => {kanye west}   0.01040000 0.5672727  0.01833333
[2] {the pussycat dolls}        => {rihanna}      0.01040000 0.5777778  0.01800000
[3] {sonata arctica}            => {nightwish}    0.01346667 0.5101010  0.02640000
[4] {judas priest}              => {iron maiden}  0.01353333 0.5075000  0.02666667
[5] {led zeppelin,the doors} => {pink floyd}   0.01066667 0.5970149  0.01786667
[6] {pink floyd,the doors}   => {led zeppelin} 0.01066667 0.5387205  0.01980000
    lift      count
[1]  8.854413 156
[2] 13.415893 156
[3]  8.236292 202
[4]  8.562992 203
[5]  5.689469 160
[6]  6.802027 160
    lhs                                    rhs           support    confidence
[1] {keane}                             => {coldplay}    0.02226667 0.6374046
[2] {radiohead,snow patrol}             => {coldplay}    0.01006667 0.6344538
[3] {oasis,the killers}                 => {coldplay}    0.01113333 0.6626984
[4] {coldplay,the smashing pumpkins}    => {radiohead}   0.01093333 0.6283525
[5] {the beatles,the smashing pumpkins} => {radiohead}   0.01146667 0.6209386
[6] {sigur rós,the beatles}             => {radiohead}   0.01046667 0.6434426
[7] {bob dylan,pink floyd}              => {the beatles} 0.01033333 0.6150794
    coverage   lift     count
[1] 0.03493333 4.020634 334
[2] 0.01586667 4.002021 151
[3] 0.01680000 4.180183 167
[4] 0.01740000 3.485683 164
[5] 0.01846667 3.444556 172
[6] 0.01626667 3.569393 157
[7] 0.01680000 3.458092 155
    lhs        rhs        support    confidence coverage   lift     count
[1] {keane} => {coldplay} 0.02226667 0.6374046  0.03493333 4.020634 334
    lhs       rhs          support confidence coverage   lift     count
[1] {t.i.} => {kanye west} 0.0104  0.5672727  0.01833333 8.854413 156
```