STA380Exercises

2022-08-06

STA Exercises - Victor Lai

Probability Practice

Part A. Visitors to your website are asked to answer a single survey question before they get access to the content on the page. Among all of the users, there are two categories: Random Clicker (RC), and Truthful Clicker (TC). There are two possible answers to the survey: yes and no. Random clickers would click either one with equal probability. You are also giving the information that the expected fraction of random clickers is 0.3. After a trial period, you get the following survey results: 65% said Yes and 35% said No. What fraction of people who are truthful clickers answered yes? Hint: use the rule of total probability.

	RC	TC	Total
Yes	15	50	65
No	15	20	35
Total	30	70	100

The total amount of truthful clickers is 70% and 50% answered yes, so 5/7 of truthful clickers answered yes.

Part B. Imagine a medical test for a disease with the following two attributes:

- The sensitivity is about 0.993. That is, if someone has the disease, there is a probability of 0.993 that they will test positive.
- The specificity is about 0.9999. This means that if someone doesn't have the disease, there is probability of 0.9999 that they will test negative.
- In the general population, incidence of the disease is reasonably rare: about 0.0025% of all people have it (or 0.000025 as a decimal probability).

Suppose someone tests positive. What is the probability that they have the disease?

	Test Positive	Test Negative	
Have Disease	0.00002482	0.00000018	0.000025
No Disease	0.0000999975	0.999875	0.999975
	0.0001248175	0.99987518	1

Given that someone tests positive, the probability that they have the disease is 0.00002482/0.0001248175 = 0.19885 or about one fifth.

Wrangling the Billboard Top 100

Consider the data in billboard.csv containing every song to appear on the weekly Billboard Top 100 chart since 1958, up through the middle of 2021. Each row of this data corresponds to a single song in a single week. For our purposes, the relevant columns here are:

- performer: who performed the song
- song: the title of the song
- year: year (1958 to 2021)
- week: chart week of that year (1, 2, etc)
- week_position: what position that song occupied that week on the Billboard top 100 chart.

Use your skills in data wrangling and plotting to answer the following three questions.

Part A: Make a table of the top 10 most popular songs since 1958, as measured by the *total number of weeks that a song spent on the Billboard Top 100.* Note that these data end in week 22 of 2021, so the most popular songs of 2021 will not have up-to-the-minute data; please send our apologies to The Weeknd.

Your table should have **10 rows** and **3 columns**: `performer`, `song`, and `count`, where `count` represents the number of weeks that song appeared in the Billboard Top 100. Make sure the entries are sorted in descending order of the `count` variable, so that the more popular songs appear at the top of the table. Give your table a short caption describing what is shown in the table.

(*Note*: you'll want to use both `performer` and `song` in any `group_by` operations, to account for the fact that multiple unique songs can share the same title.)

```
billboard = read.csv('Data/billboard.csv', header=TRUE)
billboard = subset(billboard, select = c('performer','song','year','week','week_position')) #extract re
head(billboard)
```

```
##    performer                   song year week week_position
## 1 Patty Duke Don't Just Stand There 1965   29            34
## 2 Patty Duke Don't Just Stand There 1965   30            22
## 3 Patty Duke Don't Just Stand There 1965   31            14
## 4 Patty Duke Don't Just Stand There 1965   32            10
## 5 Patty Duke Don't Just Stand There 1965   33             8
## 6 Patty Duke Don't Just Stand There 1965   34             8
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
billboard %>%
  group_by(performer,song) %>%
  summarize(count = length(week)) %>%
  arrange(desc(count)) %>%
  head(10) %>%
  knitr::kable(caption="Top 10 most popular songs since 1958")
```

```
## `summarise()` has grouped output by 'performer'. You can override using the
## `.groups` argument.
```

Table 3: Top 10 most popular songs since 1958

performer	song	count
Imagine Dragons	Radioactive	87
AWOLNATION	Sail	79
Jason Mraz	I'm Yours	76
The Weeknd	Blinding Lights	76
LeAnn Rimes	How Do I Live	69
LMFAO Featuring Lauren Bennett & GoonRock	Party Rock Anthem	68
OneRepublic	Counting Stars	68
Adele	Rolling In The Deep	65
Jewel	Foolish Games/You Were Meant For Me	65
Carrie Underwood	Before He Cheats	64

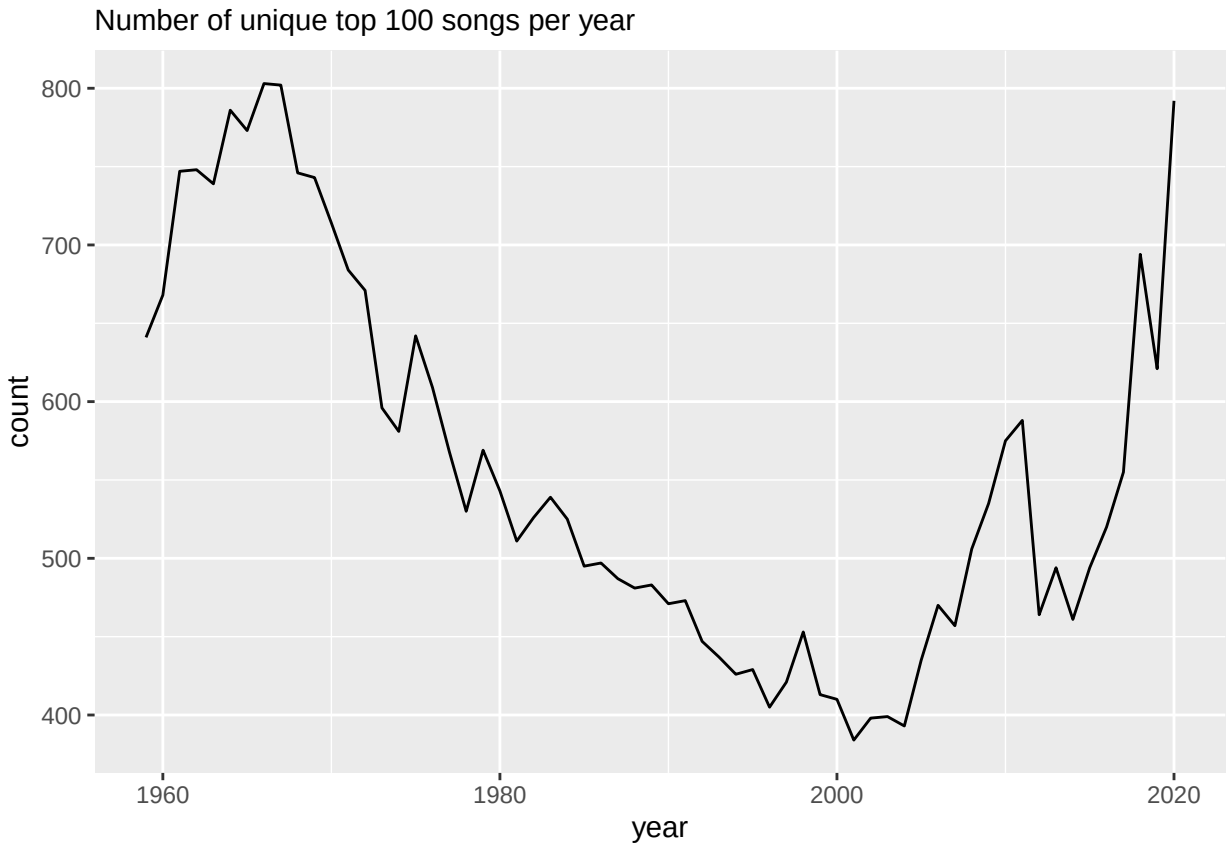
Part B: Is the "musical diversity" of the Billboard Top 100 changing over time? Let's find out. We'll measure the musical diversity of given year as *the number of unique songs that appeared in the Billboard Top 100 that year.* Make a line graph that plots this measure of musical diversity over the years. The x axis should show the year, while the y axis should show the number of unique songs appearing at any position on the Billboard Top 100 chart in any week that year. For this part, please filter the data set so that it excludes the years 1958 and 2021, since we do not have complete data on either of those years. Give the figure an informative caption in which you explain what is shown in the figure and comment on any interesting trends you see.

There are number of ways to accomplish the data wrangling here. We offer you two hints on two possibilities:

1. You could use two distinct sets of data-wrangling steps. The first set of steps would get you a table that counts the number of times that a given song appears on the Top 100 in a given year. The second set of steps operate on the result of the first set of steps; it would count the number of unique songs that appeared on the Top 100 in each year, *irrespective of how many times* it had appeared.

2. You could use a single set of data-wrangling steps that combines the `length` and `unique` commands.

```
#length(billboard)
billboard[!(billboard$year %in% c(1958,2021)),] %>% #exclude 1958,2021
  group_by(year) %>%
  summarize(count = length(unique(song))) %>% #count number of unique per year
  ggplot() +
  geom_line(aes(x=year, y=count)) +#line plot
  labs(subtitle="Number of unique top 100 songs per year")
```



It looks like several songs became really popular around 2000 and occupied spots on the top100 for weeks at a time while back in the 1960s and today there are more diverse songs.

Part C: Let's define a "ten-week hit" as a single song that appeared on the Billboard Top 100 for at least ten weeks. There are 19 artists in U.S. musical history since 1958 who have had *at least 30 songs* that were "ten-week hits." Make a bar plot for these 19 artists, showing how many ten-week hits each one had in their musical career. Give the plot an informative caption in which you explain what is shown.

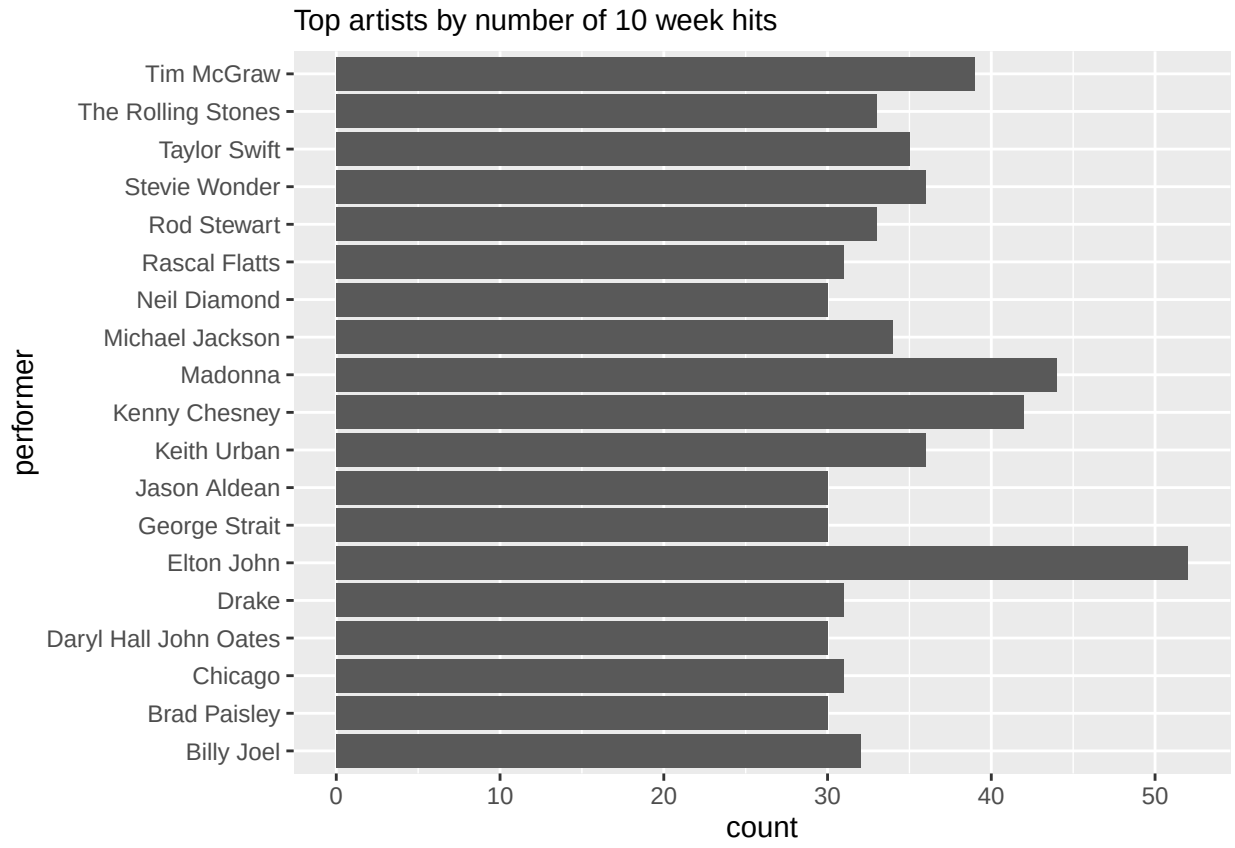
Notes:

1. You might find this easier to accomplish in two distinct sets of data wrangling steps.
2. Make sure that the individuals names of the artists are readable in your plot, and that they're not all jumbled together. If you find that your plot isn't readable with vertical bars, you can add a `coord_flip()` layer to your plot to make the bars (and labels) run horizontally instead.
3. By default a bar plot will order the artists in alphabetical order. This is acceptable to turn in. But if you'd like to order them according to some other variable, you can use the `fct_reorder` function, described in this blog post. This is optional.

```
billboard %>%
  group_by(song,performer) %>% #group by songs
  summarize(count = length(week)) %>% #count weeks
  filter(count>=10) %>% #filter out <10 weeks
  group_by(performer) %>% #group by performer
  summarize(count = length(song)) %>% #count songs
  filter(count>=30) %>% #filter out <30 songs
```

```
  ggplot() +
  geom_col(aes(y=performer, x=count))  +
  labs(subtitle="Top artists by number of 10 week hits")
```

```
## `summarise()` has grouped output by 'song'. You can override using the
## `.groups` argument.
```



Visual story telling part 1: green buildings

The case

Over the past decade, both investors and the general public have paid increasingly close attention to the benefits of environmentally conscious buildings. There are both ethical and economic forces at work here. In commercial real estate, issues of eco-friendliness are intimately tied up with ordinary decisions about how to allocate capital. In this context, the decision to invest in eco-friendly buildings could pay off in at least four ways.

1. Every building has the obvious list of recurring costs: water, climate control, lighting, waste disposal, and so forth. Almost by definition, these costs are lower in green buildings.
2. Green buildings are often associated with better indoor environments—the kind that are full of sunlight, natural materials, and various other humane touches. Such environments, in turn, might result in higher employee productivity and lower absenteeism, and might therefore be more coveted by potential tenants. The financial impact of this factor, however, is rather hard to quantify ex ante; you cannot

simply ask an engineer in the same way that you could ask a question such as, "How much are these solar panels likely to save on the power bill?"

3. Green buildings make for good PR. They send a signal about social responsibility and ecological awareness, and might therefore command a premium from potential tenants who want their customers to associate them with these values. It is widely believed that a good corporate image may enable a firm to charge premium prices, to hire better talent, and to attract socially conscious investors.

4. Finally, sustainable buildings might have longer economically valuable lives. For one thing, they are expected to last longer, in a direct physical sense. (One of the core concepts of the green-building movement is "life-cycle analysis," which accounts for the high front-end environmental impact of acquiring materials and constructing a new building in the first place.) Moreover, green buildings may also be less susceptible to market risk—in particular, the risk that energy prices will spike, driving away tenants into the arms of bolder, greener investors.

Of course, much of this is mere conjecture. At the end of the day, tenants may or may not be willing to pay a premium for rental space in green buildings. We can only find out by carefully examining data on the commercial real-estate market.

The file greenbuildings.csv contains data on 7,894 commercial rental properties from across the United States. Of these, 685 properties have been awarded either LEED or EnergyStar certification as a green building. You can easily find out more about these rating systems on the web, e.g. at www.usgbc.org. The basic idea is that a commercial property can receive a green certification if its energy efficiency, carbon footprint, site selection, and building materials meet certain environmental benchmarks, as certified by outside engineers.

A group of real estate economists constructed the data in the following way. Of the 1,360 green-certified buildings listed as of December 2007 on the LEED or EnergyStar websites, current information about building characteristics and monthly rents were available for 685 of them. In order to provide a control population, each of these 685 buildings was matched to a cluster of nearby commercial buildings in the CoStar database. Each small cluster contains one green-certified building, and all non-rated buildings within a quarter-mile radius of the certified building. On average, each of the 685 clusters contains roughly 12 buildings, for a total of 7,894 data points.

The columns of the data set are coded as follows:

- CS.PropertyID: the building's unique identifier in the CoStar database.
- cluster: an identifier for the building cluster, with each cluster containing one green-certified building and at least one other non-green-certified building within a quarter-mile radius of the cluster center.
- size: the total square footage of available rental space in the building.
- empl.gr: the year-on-year growth rate in employment in the building's geographic region.
- Rent: the rent charged to tenants in the building, in dollars per square foot per calendar year.
- leasing.rate: a measure of occupancy; the fraction of the building's available space currently under lease.
- stories: the height of the building in stories.
- age: the age of the building in years.
- renovated: whether the building has undergone substantial renovations during its lifetime.
- class.a, class.b: indicators for two classes of building quality (the third is Class C). These are relative classifications within a specific market. Class A buildings are generally the highest-quality properties in a given market. Class B buildings are a notch down, but still of reasonable quality. Class C buildings are the least desirable properties in a given market.

- green.rating: an indicator for whether the building is either LEED- or EnergyStar-certified.
- LEED, Energystar: indicators for the two specific kinds of green certifications.
- net: an indicator as to whether the rent is quoted on a "net contract" basis. Tenants with net-rental contracts pay their own utility costs, which are otherwise included in the quoted rental price.
- amenities: an indicator of whether at least one of the following amenities is available on-site: bank, convenience store, dry cleaner, restaurant, retail shops, fitness center.
- cd.total.07: number of cooling degree days in the building's region in 2007. A degree day is a measure of demand for energy; higher values mean greater demand. Cooling degree days are measured relative to a baseline outdoor temperature, below which a building needs no cooling.
- hd.total07: number of heating degree days in the building's region in 2007. Heating degree days are also measured relative to a baseline outdoor temperature, above which a building needs no heating.
- total.dd.07: the total number of degree days (either heating or cooling) in the building's region in 2007.
- Precipitation: annual precipitation in inches in the building's geographic region.
- Gas.Costs: a measure of how much natural gas costs in the building's geographic region.
- Electricity.Costs: a measure of how much electricity costs in the building's geographic region.
- cluster.rent: a measure of average rent per square-foot per calendar year in the building's local market.

The goal

An Austin real-estate developer is interested in the possible economic impact of "going green" in her latest project: a new 15-story mixed-use building on East Cesar Chavez, just across I-35 from downtown. Will investing in a green building be worth it, from an economic perspective? The baseline construction costs are $100 million, with a 5% expected premium for green certification.

The developer has had someone on her staff, who's been described to her as a "total Excel guru from his undergrad statistics course," run some numbers on this data set and make a preliminary recommendation. Here's how this person described his process.

> I began by cleaning the data a little bit. In particular, I noticed that a handful of the buildings in the data set had very low occupancy rates (less than 10% of available space occupied). I decided to remove these buildings from consideration, on the theory that these buildings might have something weird going on with them, and could potentially distort the analysis. Once I scrubbed these low-occupancy buildings from the data set, I looked at the green buildings and non-green buildings separately. The median market rent in the non-green buildings was $25 per square foot per year, while the median market rent in the green buildings was $27.60 per square foot per year: about $2.60 more per square foot. (I used the median rather than the mean, because there were still some outliers in the data, and the median is a lot more robust to outliers.) Because our building would be 250,000 square feet, this would translate into an additional $250000 x 2.6 = $650000 of extra revenue per year if we build the green building.
>
> Our expected baseline construction costs are $100 million, with a 5% expected premium for green certification. Thus we should expect to spend an extra $5 million on the green building. Based on the extra revenue we would make, we would recuperate these costs in $5000000/650000 = 7.7 years. Even if our occupancy rate were only 90%, we would still recuperate the costs in a little over 8 years. Thus from year 9 onwards, we would be making an extra $650,000 per year in profit. Since the building will be earning rents for 30 years or more, it seems like a good financial move to build the green building.

The developer listened to this recommendation, understood the analysis, and still felt unconvinced. She has therefore asked you to revisit the report, so that she can get a second opinion.

Do you agree with the conclusions of her on-staff stats guru? If so, point to evidence supporting his case. If not, explain specifically where and why the analysis goes wrong, and how it can be improved. Do you see the possibility of confounding variables for the relationship between rent and green status? If so, provide evidence for confounding, and see if you can also make a picture that visually shows how we might "adjust" for such a confounder. *Tell your story in pictures, with appropriate introductory and supporting text.*

Note: this is intended as an exercise in visual and numerical story-telling. Your approach should rely on pictures and/or tables, not a regression model. Tell a story understandable to a non-technical audience. Keep it concise.

```
green = read.csv('Data/greenbuildings.csv', header=TRUE)
green %>%
  filter(leasing_rate<10) %>%
  group_by(green_rating) %>%
  summarize(count = length(CS_PropertyID))
```

```
## # A tibble: 2 x 2
##   green_rating count
##          <int> <int>
## 1            0   214
## 2            1     1
```

I think it could be worth mentioning that the low occupancy buildings were overwhelmingly not green, pushing the case further in green buildings' favor.

```
green %>%
  ggplot() +
  geom_point(aes(x=age, y=Rent))  + facet_wrap(~green_rating) +
  labs(subtitle="Rent by building age")
```

Another observation is that older buildings tend not to be green as well as have less rent. This also somewhat supports going green.

```
green %>%
  ggplot() +
  geom_point(aes(x=cluster_rent, y=Rent))  + facet_wrap(~green_rating) +
  labs(subtitle="Rent by cluster")
```

Here we can observe that rent correlates positively with the cluster rent. This could be a potentially confounding variable. If we wanted to adjust for it, we could try fitting a regression model and subtract the line of best fit to account for the impact of cluster rent on overall rent.

Overall, the staff's analysis is reasonably solid. However, he did not account for the time value of money and inflation, that is, how today's money is worth more than the same amount of money 8 years in the future. So it is improbable to fully recuperate our costs within 8 or so years even at max occupancy, and will likely take a bit longer. However, unless inflation is greater than 650k/5M = 13%, we should be more or less seeing returns on investment every year and it should still be worth investing in going green.

Visual story telling part 2: Capital Metro data

The file capmetro_UT.csv contains data from Austin's own Capital Metro bus network, including shuttles to, from, and around the UT campus. These data track ridership on buses in the UT area. Ridership is measured by an optical scanner that counts how many people embark and alight the bus at each stop. Each row in the data set corresponds to a 15-minute period between the hours of 6 AM and 10 PM, each and every day, from September through November 2018. The variables are:

- *timestamp*: the beginning of the 15-minute window for that row of data
- *boarding*: how many people got on board any Capital Metro bus on the UT campus in the specific 15 minute window
- *alighting*: how many people got off ("alit") any Capital Metro bus on the UT campus in the specific 15 minute window
- *day_of_week* and *weekend*: Monday, Tuesday, etc, as well as an indicator for whether it's a weekend.

- *temperature*: temperature at that time in degrees F
- *hour_of_day*: on 24-hour time, so 6 for 6 AM, 13 for 1 PM, 14 for 2 PM, etc.
- *month*: July through December

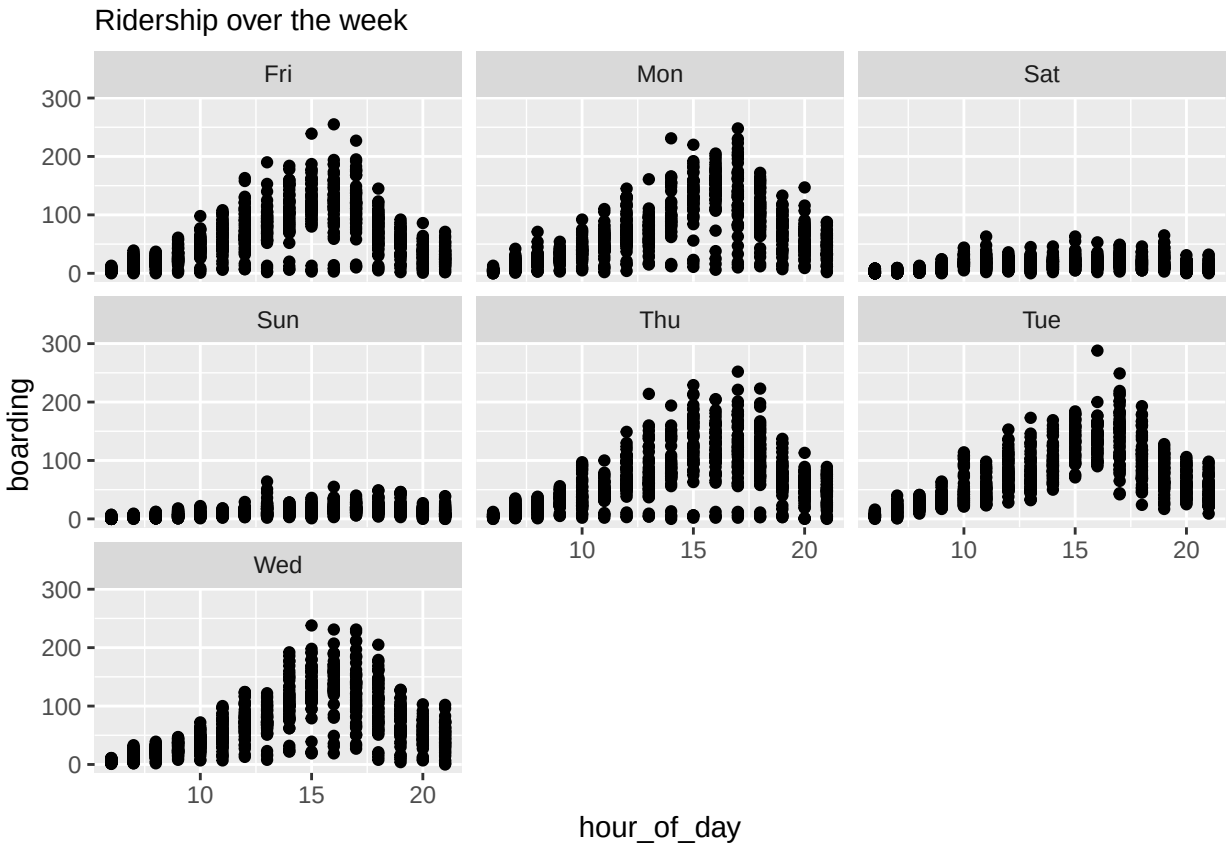
Your task is to create a figure, or set of related figures, that tell an interesting story about Capital Metro ridership patterns around the UT-Austin campus during the semester in question. Provide a clear annotation/caption for each figure, but the figure(s) should be more or less stand-alone, in that you shouldn't need many, many paragraphs to convey its meaning. Rather, the figure together with a concise caption should speak for itself as far as possible.

You have broad freedom to look at any variables you'd like here -- try to find that sweet spot where you're showing genuinely interesting relationships among more than just two variables, but where the resulting figure or set of figures doesn't become overwhelming/confusing. (Faceting/panel plots might be especially useful here.)

```
metro = read.csv('Data/capmetro_UT.csv', header=TRUE)
head(metro)
```

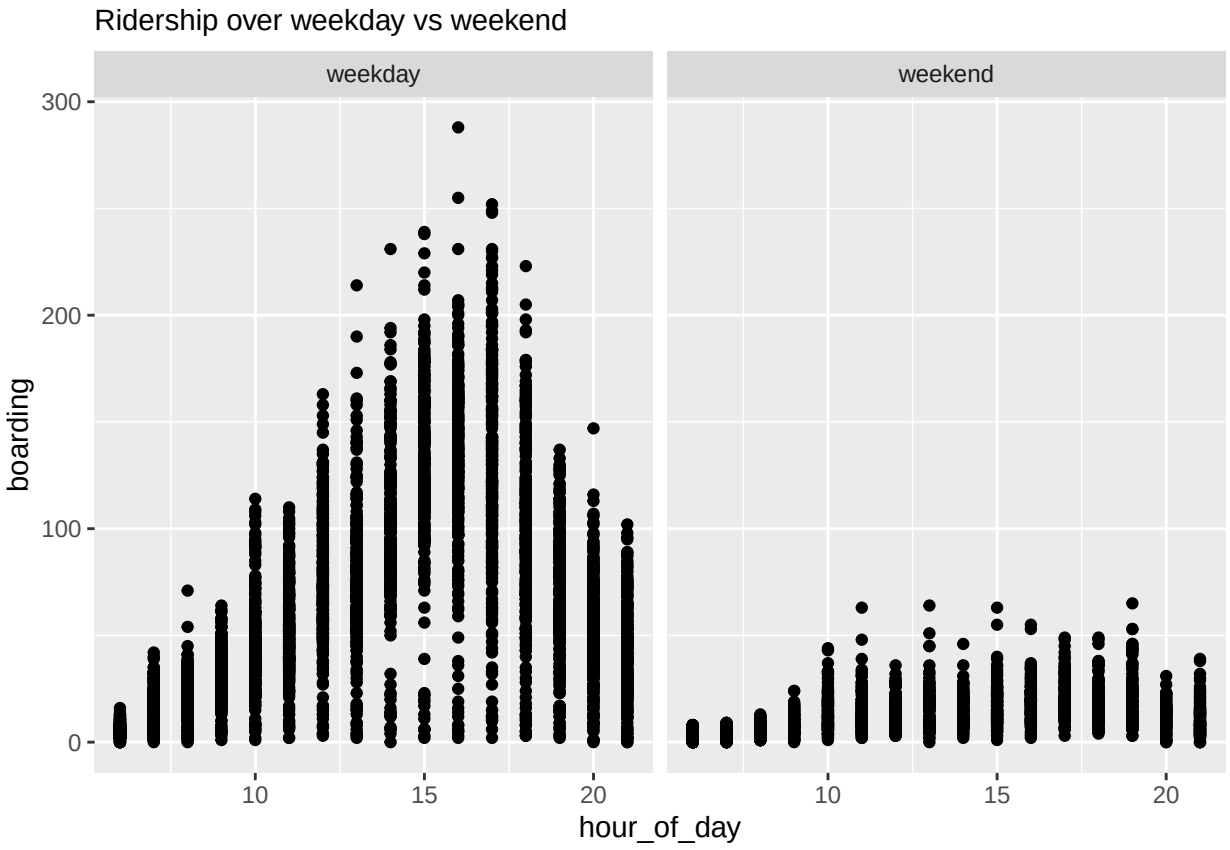
```
##             timestamp boarding alighting day_of_week temperature hour_of_day
## 1 2018-09-01 06:00:00        0         1         Sat       74.82           6
## 2 2018-09-01 06:15:00        2         1         Sat       74.82           6
## 3 2018-09-01 06:30:00        3         4         Sat       74.82           6
## 4 2018-09-01 06:45:00        3         4         Sat       74.82           6
## 5 2018-09-01 07:00:00        2         4         Sat       74.39           7
## 6 2018-09-01 07:15:00        4         4         Sat       74.39           7
##   month weekend
## 1   Sep weekend
## 2   Sep weekend
## 3   Sep weekend
## 4   Sep weekend
## 5   Sep weekend
## 6   Sep weekend
```

```
metro %>%
  ggplot() +
  geom_point(aes(x=hour_of_day, y=boarding))  + facet_wrap(~day_of_week) +
  labs(subtitle="Ridership over the week")
```

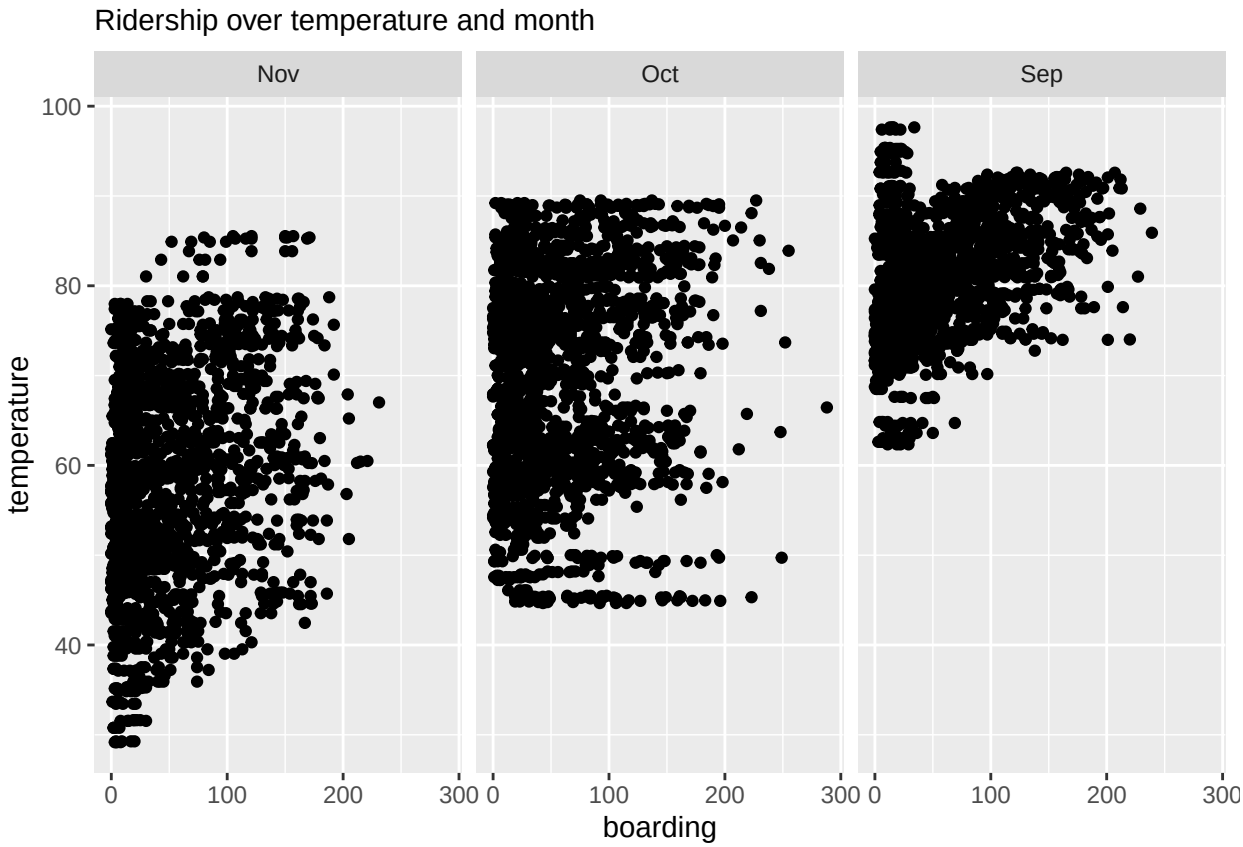


We can see that ridership has a strong pattern over the weekdays and is much lower on weekends.

```
metro %>%
  ggplot() +
  geom_point(aes(x=hour_of_day, y=boarding))  + facet_wrap(~weekend) +
  labs(subtitle="Ridership over weekday vs weekend")
```

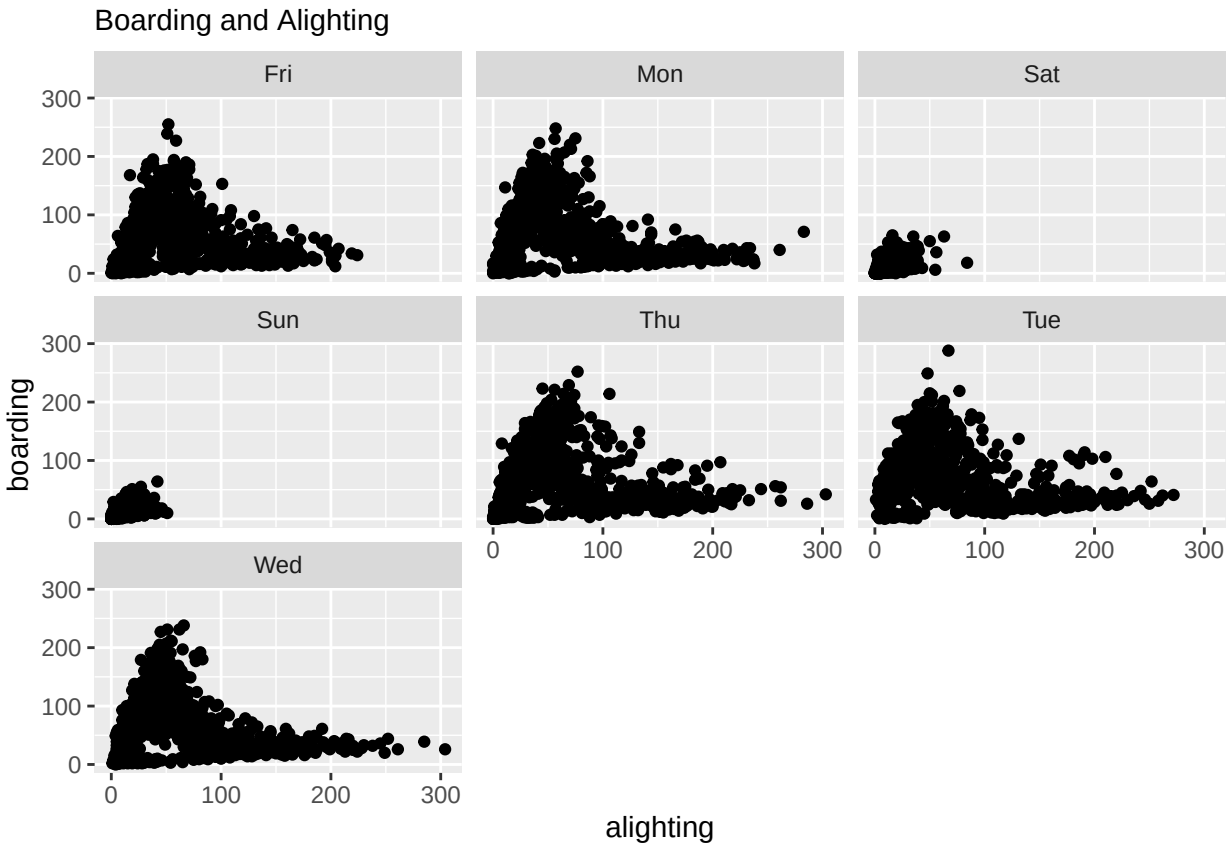



```
metro %>%
  ggplot() +
  geom_point(aes(x=boarding, y=temperature))  + facet_wrap(~month) +
  labs(subtitle="Ridership over temperature and month")
```



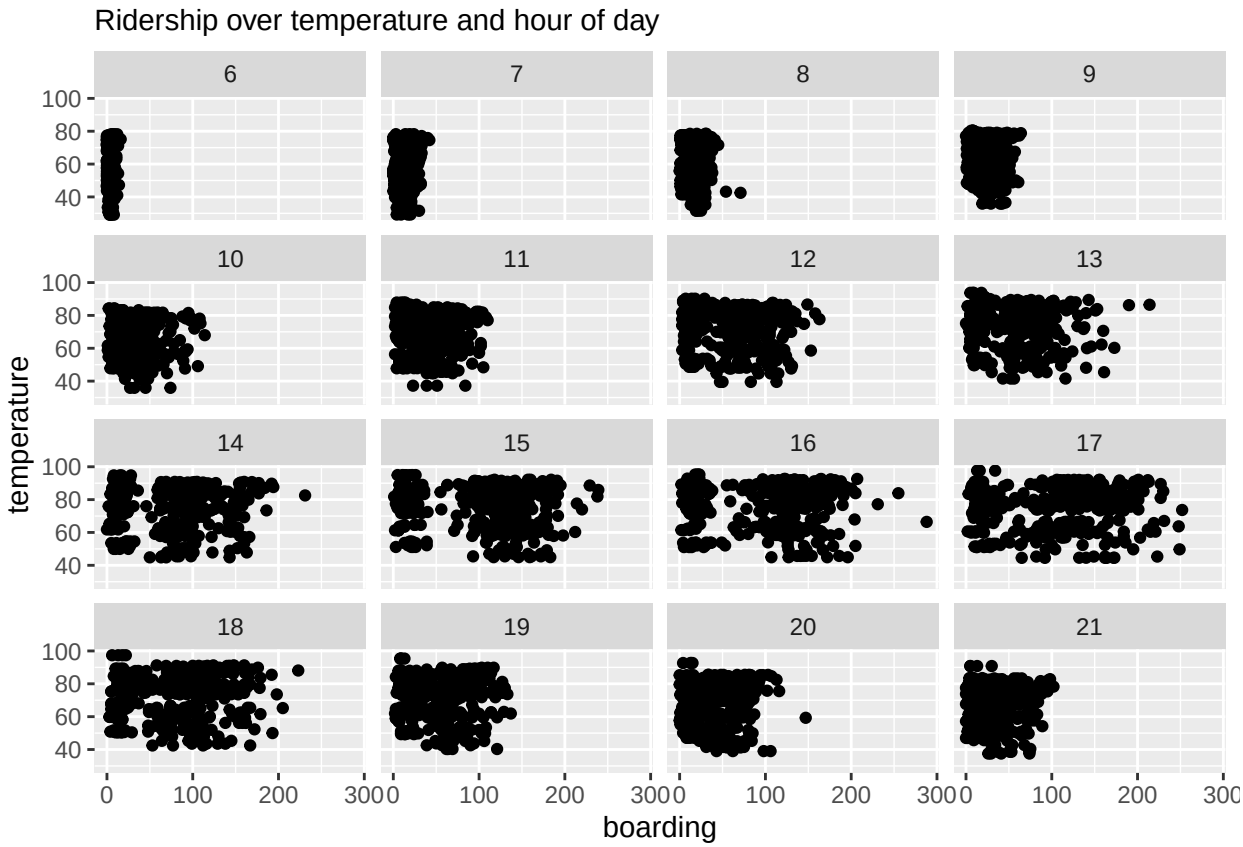
We can see that few people ride at high temperatures above 93 or so, as well as below 35. We also have a sense of the temperature ranges for each month.

```
metro %>%
  ggplot() +
  geom_point(aes(x=alighting, y=boarding))  + facet_wrap(~day_of_week) +
  labs(subtitle="Boarding and Alighting")
```



We can see that it is less busy on the weekends again with few people boarding and alighting. On week days though, there seems to be a bit of a negative correlation between boarding and alighting so fewer people are boarding where many are alighting and vice versa.

```
metro %>%
  ggplot() +
  geom_point(aes(x=boarding, y=temperature))  + facet_wrap(~hour_of_day) +
  labs(subtitle="Ridership over temperature and hour of day")
```



Here we can see both temperature and ridership rising during the day time and shrinking towards the evening.

Portfolio modeling

Background

In this problem, you will construct three different portfolios of exchange-traded funds, or ETFs, and use bootstrap resampling to analyze the short-term tail risk of your portfolios. If you're unfamiliar with exchange-traded funds, you can read a bit about them here.

The goal

Suppose you have $100,000 in capital. Your task is to:

- Construct two different possibilities for an ETF-based portfolio, each involving an allocation of your $100,000 in capital to somewhere between 3 and 10 different ETFs. You can find a big database of ETFs here.
- Download the last five years of daily data on your chosen ETFs, using the functions in the `quantmod` package, as we used in class. Note: make sure to choose ETFs for which at least five years of data are available. There are tons of ETFs and some are quite new!
- Use bootstrap resampling to estimate the 4-week (20 trading day) value at risk of each of your three portfolios at the 5% level.
- Write a report summarizing your portfolios and your VaR findings.

You should assume that your portfolios are rebalanced each day at zero transaction cost. For example, if you're allocating your wealth evenly among 5 ETFs, you always redistribute your wealth at the end of each day so that the equal five-way split is retained, regardless of that day's appreciation/depreciation.

Notes:

- Make sure the portfolios are different from each other! (Maybe one seems safe, another aggressive, or something like that.) You're not being graded on what specific portfolios you choose... just provide some context for your choices.
- If you're unfamiliar with value at risk (VaR), you can refer to any basic explanation of the idea, e.g. here, here, or here.

There are a lot of ETFS to choose from. To filter out some of them, I decided only look at the YTD performance, and use the following strategies. ETFs are listed by symbol - YTD returns %.

Strategies:

Maximize - maximize YTD returns regardless of everything else

- BOIL
- UNG
- GAZ
- UNL

Diversify - pick best ETFs from different sectors (only from sectors with positive overall YTD)

- UNG
- TMV
- PXE
- GSP
- SOYB

Best sector - pick best specific sector and best ETFs in sector(Oil and gas)

- UNG
- GAZ
- UNL
- RJN
- UGA

```
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'
```

```
## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(foreach)
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
#maximize
mystocks = c("BOIL", "UNG", "GAZ","UNL")
myprices = getSymbols(mystocks, from = "2017-08-10")
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/BOIL?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/BOIL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/BOIL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

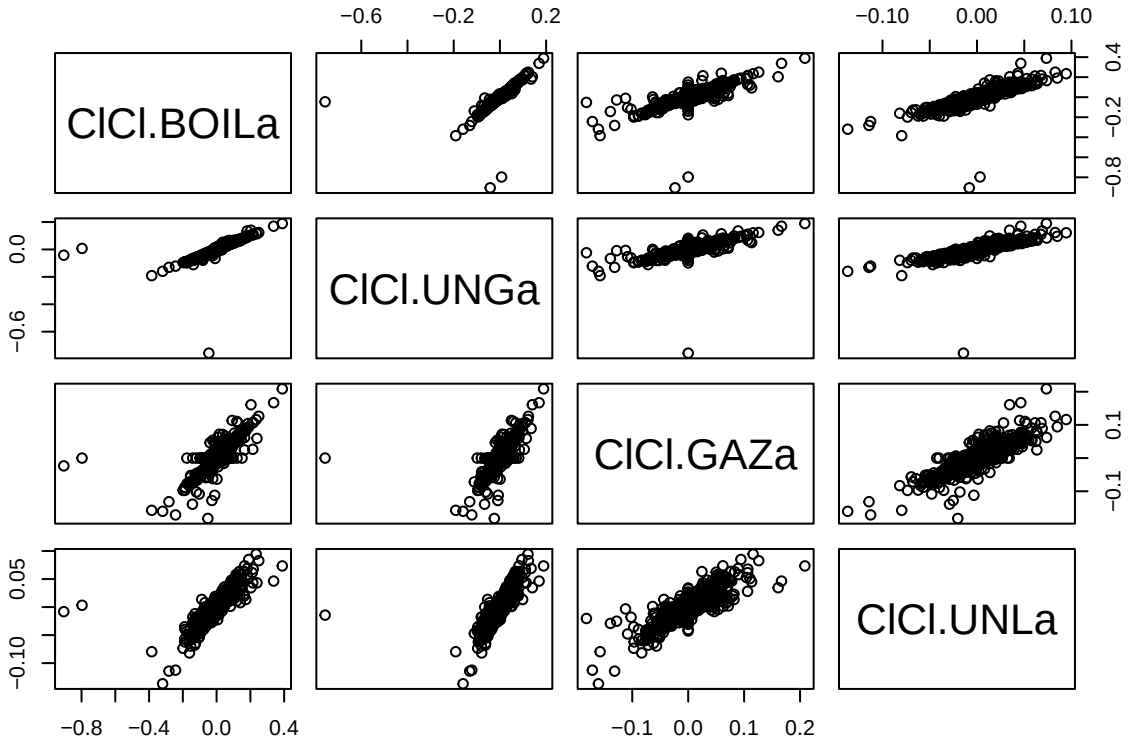
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
# Combine all the returns in a matrix
all_returns = cbind(    ClCl(BOILa),
                                ClCl(UNGa),
                                ClCl(GAZa),
                                ClCl(UNLa))
head(all_returns)
```

```
##              ClCl.BOILa    ClCl.UNGa    ClCl.GAZa    ClCl.UNLa
## 2017-08-10           NA           NA           NA           NA
## 2017-08-11 -0.002178649  0.000000000  0.000000000  0.000000000
## 2017-08-14 -0.004366812 -0.005979073  0.000000000 -0.006030151
## 2017-08-15 -0.012061404 -0.010526316  0.000000000 -0.004044489
## 2017-08-16 -0.021087680 -0.013677850  0.000000000 -0.006091371
## 2017-08-17  0.018140590  0.009245070 -0.008226846  0.004085802
```

```
all_returns = as.matrix(na.omit(all_returns))
pairs(all_returns) #plots are fairly highly correlated
```

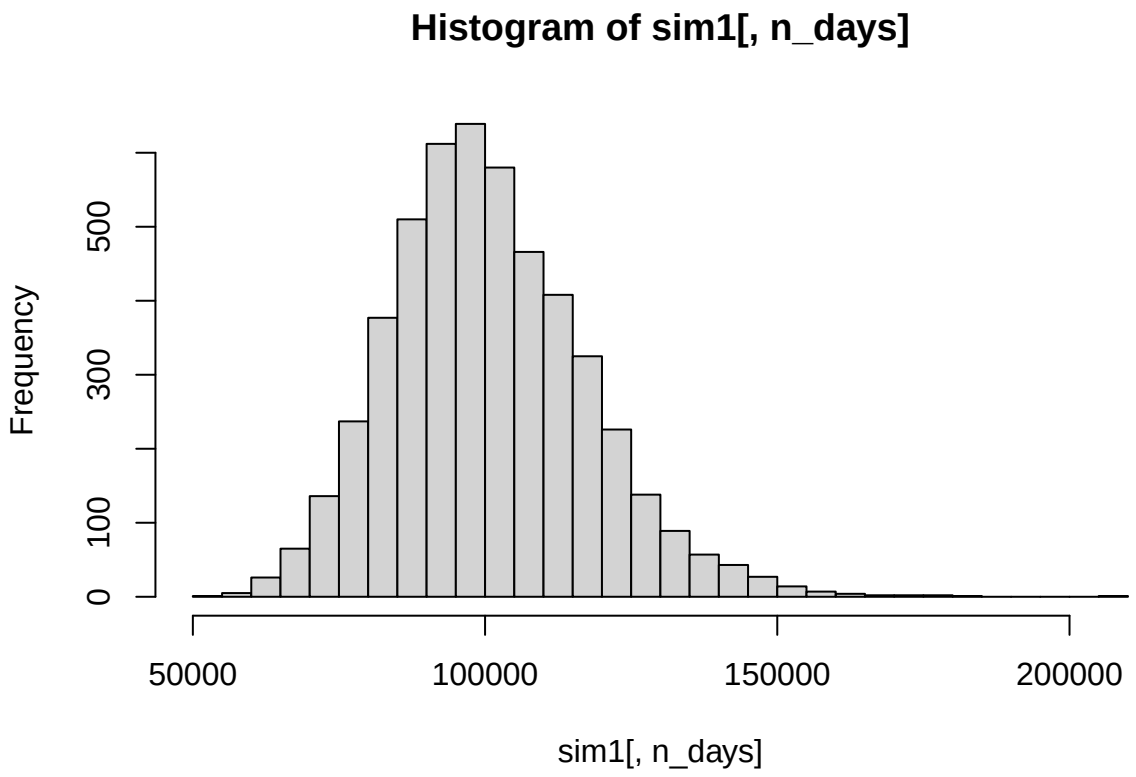



```
initial_wealth = 100000 #changed wealth
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.25, 0.25, 0.25, 0.25)
    holdings = weights * total_wealth
  n_days = 20 #changed days
    wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
    }
    wealthtracker
}
head(sim1)
```

```
##                [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 101745.28 102244.69 100506.44 103070.22 101601.65  97526.81  96553.91
## result.2  97359.82  92796.51  94480.88  95060.43  96480.07  96280.34  97241.85
## result.3  97836.68 100514.60  99274.05  99268.21  94021.34  92879.26  89745.14
## result.4  98536.44 100856.77 109243.35 109723.54 112255.55 116247.77 116453.36
## result.5  98847.64 100168.63  99266.22 100419.60  98581.96  91148.41  90242.99
## result.6 101950.78 103826.85  99655.67 105207.20 114891.71 121505.46 116167.31
##                [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  95338.26  96338.32  95577.37  95950.16  97472.82  97505.45  98718.13
```

```
## result.2  96890.59  86303.95  89972.61  92386.64  92187.25  91123.58  91809.58
## result.3  90572.97  91476.80  87580.68  97511.33  97651.60  94727.44  95746.96
## result.4 118569.62 119948.57 119081.73 116619.89 111634.79 110989.09 108056.23
## result.5  89498.71  88200.59  84783.78  83879.20  82851.75  84403.43  77656.36
## result.6 114611.13 112530.74 109784.79 113209.13 110204.42 111494.56 114723.24
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 100021.81 102751.36 101483.11 104636.17 100384.51 101194.97
## result.2  93160.31  91952.05  91474.48  87570.30  85309.85  87621.67
## result.3  93302.35  94707.37  96469.26  91035.71  89180.07  91914.04
## result.4 109863.02 107952.50 111158.77 111772.96 112140.05 111733.66
## result.5  75672.54  77164.14  75200.14  77660.77  73922.24  78320.41
## result.6 112208.28 116743.60 120501.43 128271.83 125093.80 123715.81
```

```
hist(sim1[,n_days], 25)
```



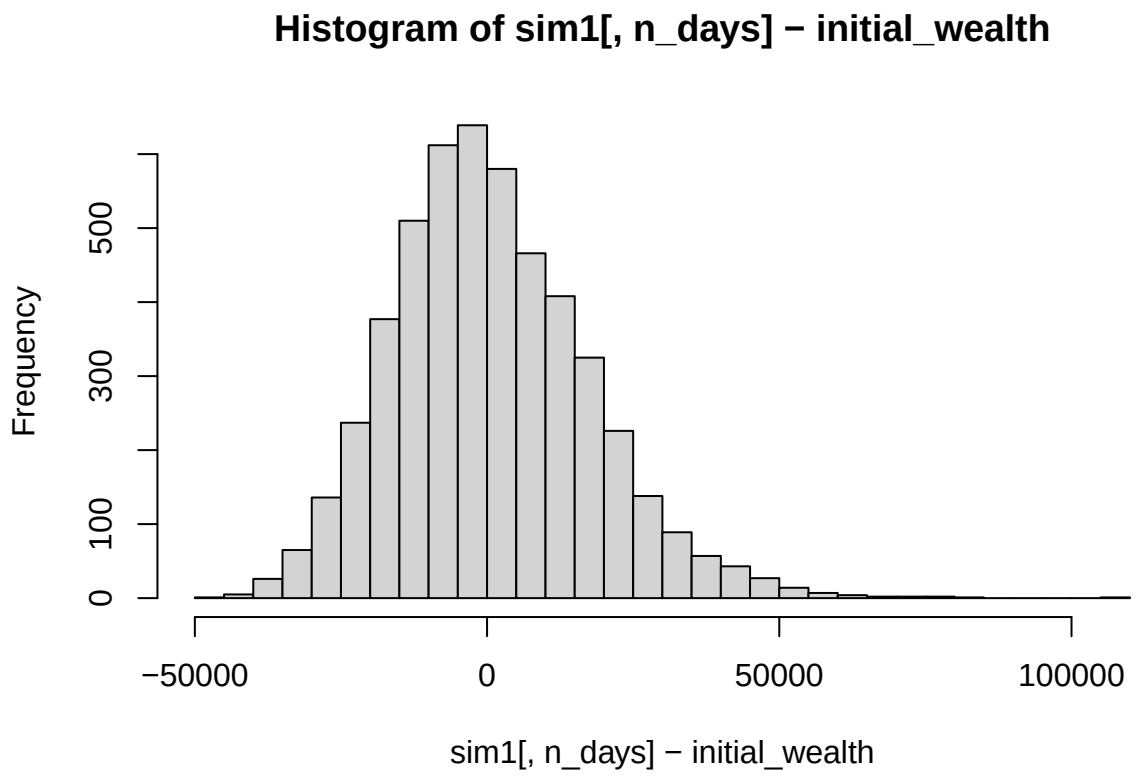

```
# Profit/loss
mean(sim1[,n_days])
```

```
## [1] 100614.2
```

```
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 614.2375
```

```
hist(sim1[,n_days]- initial_wealth, breaks=30)
```




```
# 5% value at risk:
quantile(sim1[,n_days]- initial_wealth, prob=0.05) #-25430, about 1/4 risked
```

```
##        5%
## -24609.46
```

```
#diversify
mystocks = c("UNG","TMV","PXE","GSP","SOYB")
myprices = getSymbols(mystocks, from = "2017-08-10")
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNG?
```

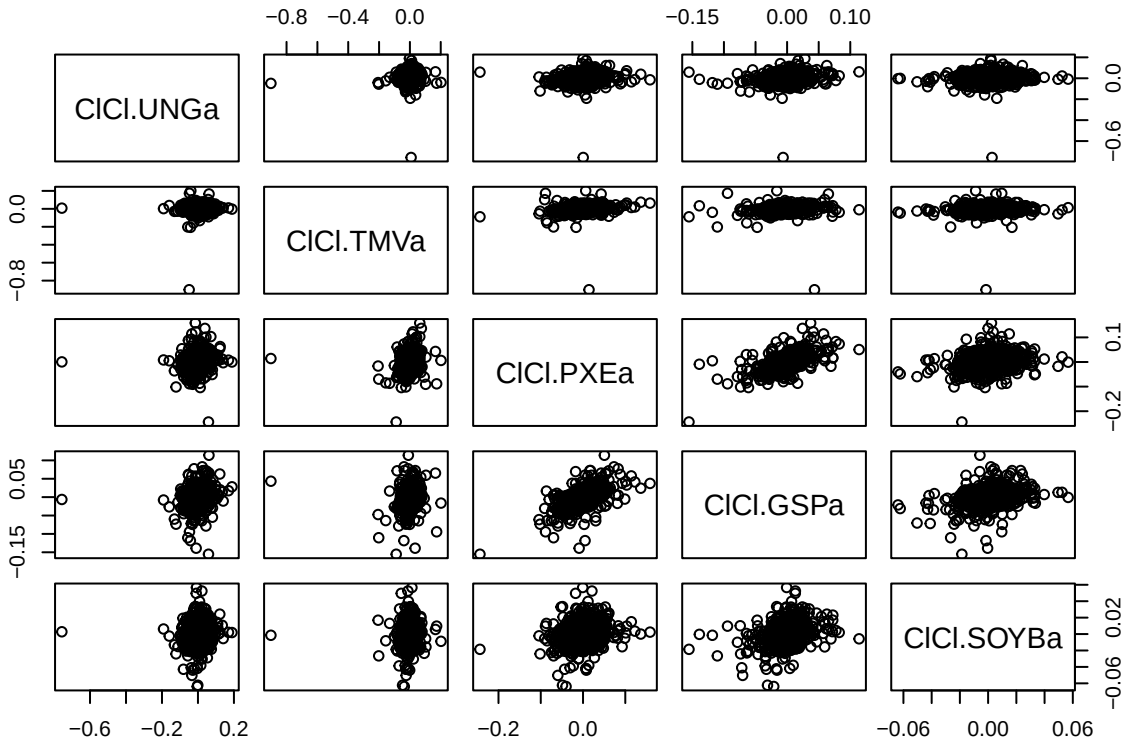
```
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/TMV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/TMV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/PXE?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/PXE?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/GSP?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GSP?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GSP?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SOYB?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SOYB?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SOYB?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
# Combine all the returns in a matrix
all_returns = cbind(    ClCl(UNGa),
                                ClCl(TMVa),
                                ClCl(PXEa),
                                ClCl(GSPa),
                                ClCl(SOYBa))
head(all_returns)
```

```
##               ClCl.UNGa     ClCl.TMVa     ClCl.PXEa     ClCl.GSPa   ClCl.SOYBa
## 2017-08-10           NA            NA            NA            NA           NA
## 2017-08-11  0.000000000 -0.0005364324  0.002750275  0.0075815011  0.005042073
## 2017-08-14 -0.005979073  0.0139559364 -0.004936972 -0.0195635816 -0.008918617
## 2017-08-15 -0.010526316  0.0127051830 -0.007166428 -0.0007674597 -0.011811079
## 2017-08-16 -0.013677850 -0.0088865498 -0.018878401 -0.0084485407  0.000569152
## 2017-08-17  0.009245070 -0.0232067978 -0.011318676 -0.0007745933  0.008532423
```

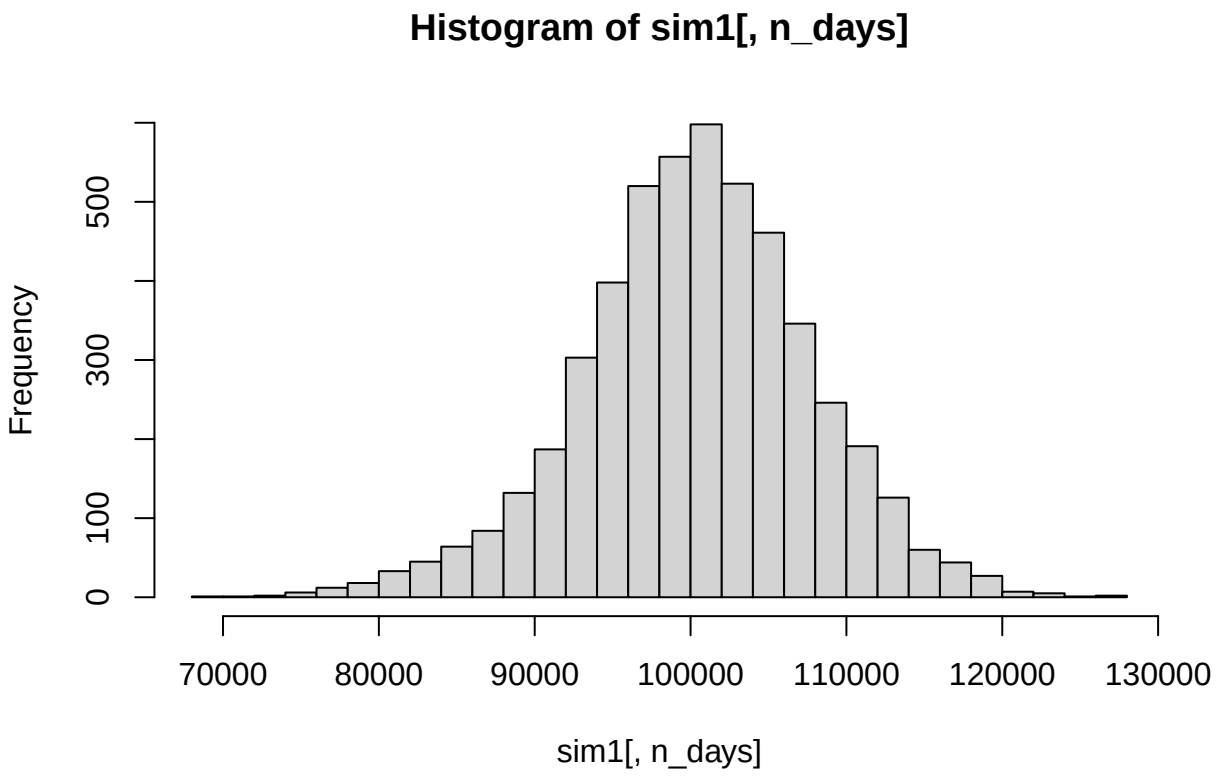
```
all_returns = as.matrix(na.omit(all_returns))
pairs(all_returns) #much weaker correlations between etfs
```



```
initial_wealth = 100000 #changed wealth
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.2, 0.2, 0.2, 0.2,0.2)
    holdings = weights * total_wealth
    n_days = 20 #changed days
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  99945.05 100843.92 102348.90 101311.70 102668.24 103558.01 104821.76
## result.2 100865.38  97538.66  95075.95  92770.24  93698.20  94292.22  94896.25
## result.3 102240.86 102324.05 103283.92 103199.72 102121.83  98226.30 101082.18
## result.4 101701.28 102044.33 102483.25 102423.50 102440.34 103851.80 104103.29
## result.5 100955.07 100778.60 100936.50 101240.27 100538.36 102627.18 100648.68
## result.6  97163.48  95422.13  96339.40  93120.52  92507.07  91547.38  91830.89
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 104004.74 104225.76  94657.15  93990.54  94501.13  95142.03  96782.50
## result.2  92475.62  92158.60  92671.13  93557.18  92985.53  94404.36  95538.07
## result.3 101925.99 101776.61 102698.57 102610.92 102438.61 101339.30 100878.35
## result.4 104404.72 101939.08 102734.37 101285.61 101700.56 102023.75 102286.62
## result.5  99181.59 100325.47 100748.94 100084.10  99586.99 100545.31 102132.44
## result.6  91091.10  90037.12  90089.99  91156.56  90392.51  89156.46  90144.18
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  96922.78  95999.61  96128.98  95577.36  96516.92  97994.25
## result.2  94359.59  94389.41  94445.78  93684.26  94335.14  90587.51
## result.3 100200.26  99923.14 100234.80 100117.33  99773.02  99631.54
## result.4 103519.26 103422.95 103334.64 103776.70 102945.35 102441.84
## result.5 101743.66 101641.19 102114.30 102021.23 101153.04 102804.02
## result.6  89943.30  92506.17  92231.81  92432.06  91347.95  94218.98
```

```
hist(sim1[,n_days], 25)
```



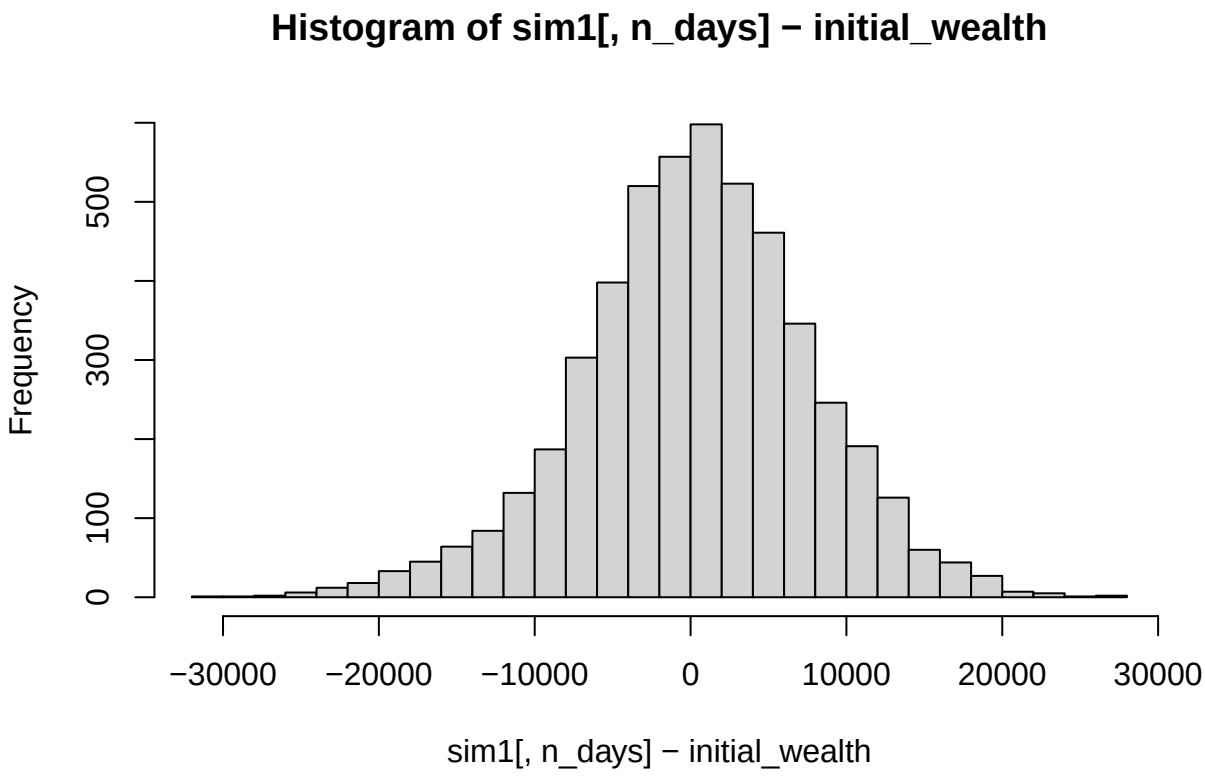

```
# Profit/loss
mean(sim1[,n_days])
```

```
## [1] 100326
```

```
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 326.034
```

```
hist(sim1[,n_days]- initial_wealth, breaks=30)
```




```
# 5% value at risk:
quantile(sim1[,n_days]- initial_wealth, prob=0.05) #-11557, less risked
```

```
##        5%
## -12319.13
```

```
#Best sector
mystocks = c("UNG","GAZ","UNL","RJN","UGA")
myprices = getSymbols(mystocks, from = "2017-08-10")
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

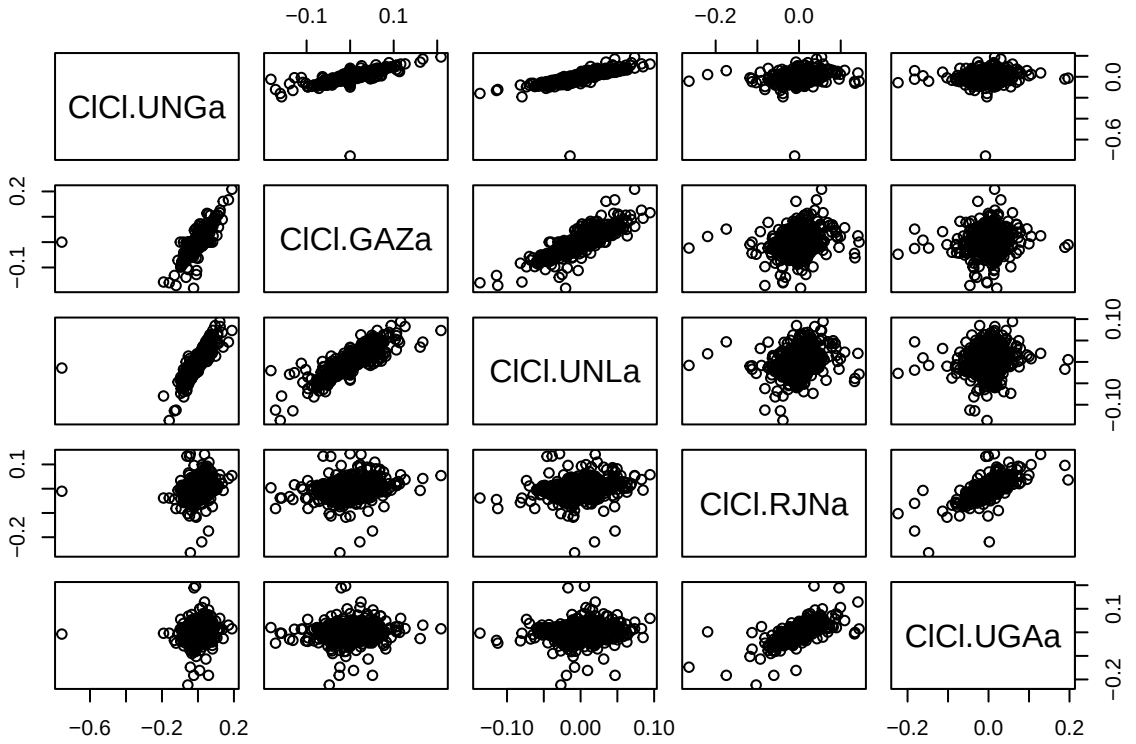
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GAZ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UNL?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/RJN?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/RJN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/RJN?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/UGA?
## period1=-2208988800&period2=1660521600&interval=1d&events=div'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UGA?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/UGA?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
# Combine all the returns in a matrix
all_returns = cbind(    ClCl(UNGa),
                                ClCl(GAZa),
                                ClCl(UNLa),
                                ClCl(RJNa),
                                ClCl(UGAa))
head(all_returns)
```

```
##               ClCl.UNGa    ClCl.GAZa    ClCl.UNLa    ClCl.RJNa    ClCl.UGAa
## 2017-08-10           NA           NA           NA           NA           NA
## 2017-08-11  0.000000000  0.000000000  0.000000000  0.000000000  0.011904723
## 2017-08-14 -0.005979073  0.000000000 -0.006030151 -0.021551724 -0.023529412
## 2017-08-15 -0.010526316  0.000000000 -0.004044489  0.000000000  0.004275165
## 2017-08-16 -0.013677850  0.000000000 -0.006091371  0.000000000 -0.011609868
## 2017-08-17  0.009245070 -0.008226846  0.004085802 -0.004405286  0.018010923
```

```
all_returns = as.matrix(na.omit(all_returns))
pairs(all_returns) # strong correlations between first 3 and weak for others
```

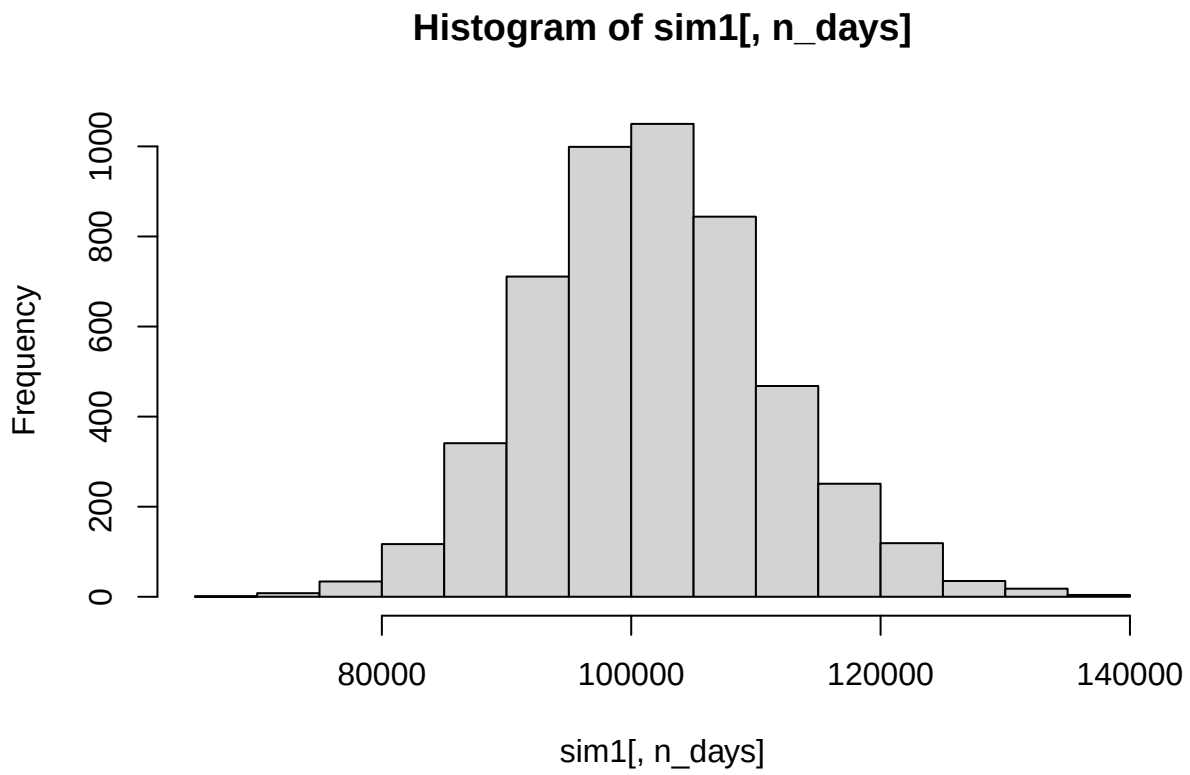



```
initial_wealth = 100000 #changed wealth
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.2, 0.2, 0.2, 0.2,0.2)
    holdings = weights * total_wealth
    n_days = 20 #changed days
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  98697.15 104523.63 104778.14 103937.36 105445.84 107457.00 106397.28
## result.2 101552.75 102399.47 100203.85  98182.90 102002.57 105028.64 105422.49
## result.3 106039.10 105685.47 105663.16 100809.52 102349.07 103921.78 107657.61
## result.4 100485.26 100185.34 101637.84 100402.53  99795.86  99675.44  98576.68
## result.5  99288.51  97044.22  96238.66  95109.92  97643.25 108047.39 108097.67
## result.6 101441.18 100672.88 102248.50 100325.51  99310.03  99654.73 104102.95
##               [,8]      [,9]     [,10]     [,11]     [,12]    [,13]     [,14]
## result.1 107171.55 108943.43 108695.83 110558.76 109448.82 108653.0 108356.15
```

```
## result.2 104150.90 104136.53 104284.75 102399.82 103613.17 102261.1 103611.71
## result.3 111686.39 111730.93 110389.16 108141.78 106837.37 106049.9 104700.80
## result.4  96786.65  96886.12  97238.47  96352.51  93131.36  93088.9  94666.87
## result.5 109029.97 109195.61 110970.44 112678.26 107909.18 108412.4 104295.99
## result.6 101819.16 109362.85 109804.81 107976.86 102163.18 103050.5 103809.65
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 109469.62 110968.46 109133.19 110162.17 109775.76 110463.38
## result.2 103936.04 101971.66  92074.10  93441.37  94241.16  93363.23
## result.3 102946.72 102418.00 101523.17 101538.27 102301.93 101879.07
## result.4  85473.36  85119.81  84637.44  85249.00  85896.76  83982.93
## result.5 100380.85  99953.25  99839.70  99494.86 100636.02  97452.70
## result.6 103979.59 106371.63 104543.66 103770.91 102180.67  99846.18
```

```
hist(sim1[,n_days], 25)
```



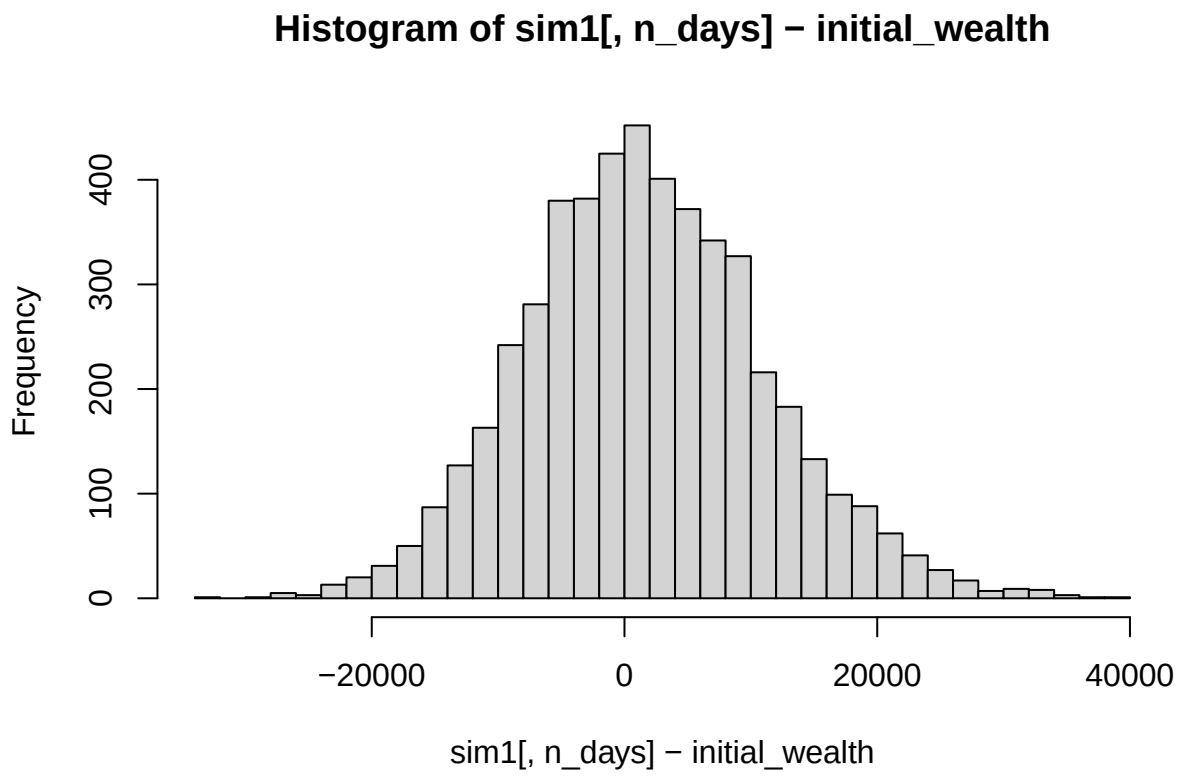

```
# Profit/loss
mean(sim1[,n_days])
```

```
## [1] 101682.3
```

```
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 1682.349
```

```
hist(sim1[,n_days]- initial_wealth, breaks=30)
```




```
# 5% value at risk:
quantile(sim1[,n_days]- initial_wealth, prob=0.05) #-13904,
```

```
##        5%
## -13237.62
```

For some reason the "maximize" strategy had very high correlations despite having ETFs from different categories and it turns out most of the ETFs were natural gas and BOIL was also natural gas but leveraged. Meanwhile, the "best sector" strategy had a mix of natural gas, oil, and energy ETFs which reduced its correlations. Finally, the "diversify" strategy had everything from different sectors which made its correlations the smallest.

The overall ranges of outcomes were:

- Maximize: -60% to +80%, mean 0.6%, risk 25%
- Diversify: -30% to +25%, mean 0.7%, risk 11.5%
- Best sector: -30% to +40%, mean 1.3%, risk 14%

Clearly, the 'maximize' strategy offered the best possible reward, but also the largest possible loss and most risk. Meanwhile, the 'diversify' strategy limited possible losses, gains, and risk, and also did slightly better in terms of mean returns. Finally, the 'best sector' strategy had similarly limited losses but also had more to gain and also had the best mean returns. However, in practice the 'best sector' strategy does seem to rely strongly on luck, and needs to readjust if the best sector changes.

Clustering and PCA

The data in wine.csv contains information on 11 chemical properties of 6500 different bottles of *vinho verde* wine from northern Portugal. In addition, two other variables about each wine are recorded:

- whether the wine is red or white
- the quality of the wine, as judged on a 1-10 scale by a panel of certified wine snobs.

Run both PCA and a clustering algorithm of your choice on the 11 chemical properties (or suitable transformations thereof) and summarize your results. Which dimensionality reduction technique makes more sense to you for this data? Convince yourself (and me) that your chosen method is easily capable of distinguishing the reds from the whites, using only the "unsupervised" information contained in the data on chemical properties. Does your unsupervised technique also seem capable of distinguishing the higher from the lower quality wines?

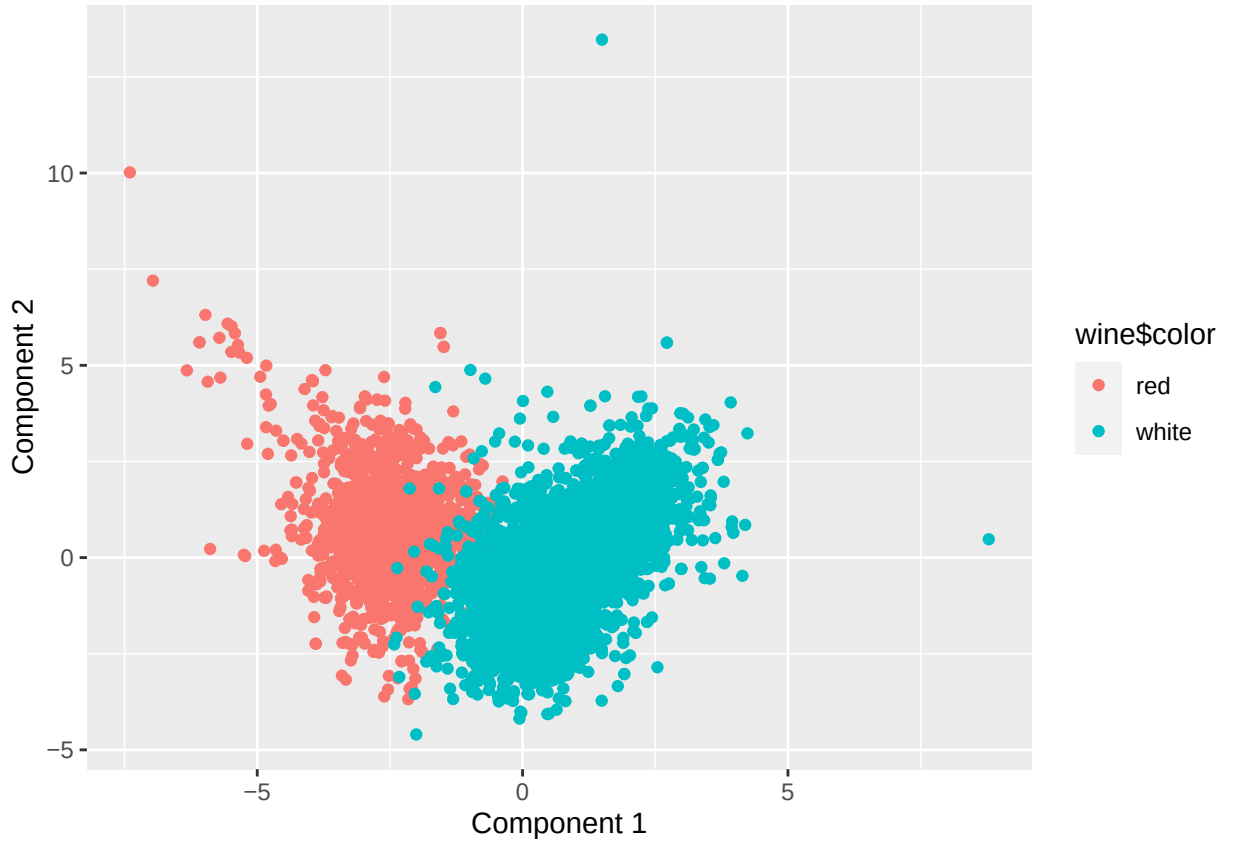
To clarify: I'm not asking you to run an supervised learning algorithms. Rather, I'm asking you to see whether the differences in the labels (red/white and quality score) emerge naturally from applying an unsupervised technique to the chemical properties. This should be straightforward to assess using plots

```
wine = read.csv('Data/wine.csv', header=TRUE)
head(wine) #11 features, then quality and color
```

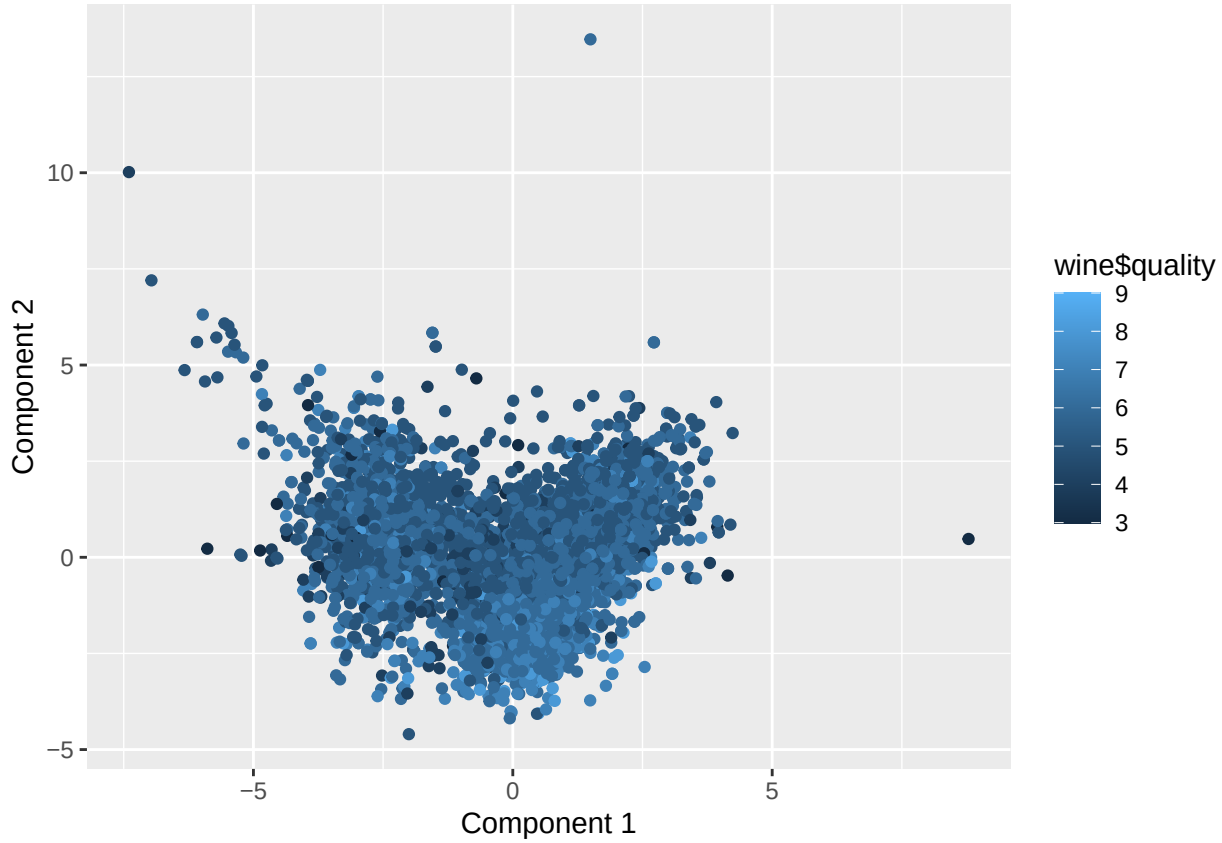
```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality color
## 1       5   red
## 2       5   red
## 3       5   red
## 4       6   red
## 5       5   red
## 6       5   red
```

```
# Center and scale the data
X = wine[,(1:11)]
X = scale(X, center=TRUE, scale=TRUE)
# Extract the centers and scales from the rescaled data (which are named attributes)
mu = attr(X,"scaled:center")
sigma = attr(X,"scaled:scale")
```

```
pc2 = prcomp(X, scale=TRUE, rank=2)
loadings = pc2$rotation
scores = pc2$x

qplot(scores[,1], scores[,2], color=wine$color, xlab='Component 1', ylab='Component 2')
```




```
qplot(scores[,1], scores[,2], color=wine$quality, xlab='Component 1', ylab='Component 2')
```




```
# The top words associated with each component
o1 = order(loadings[,1], decreasing=TRUE)
colnames(X)[head(o1,25)]
```

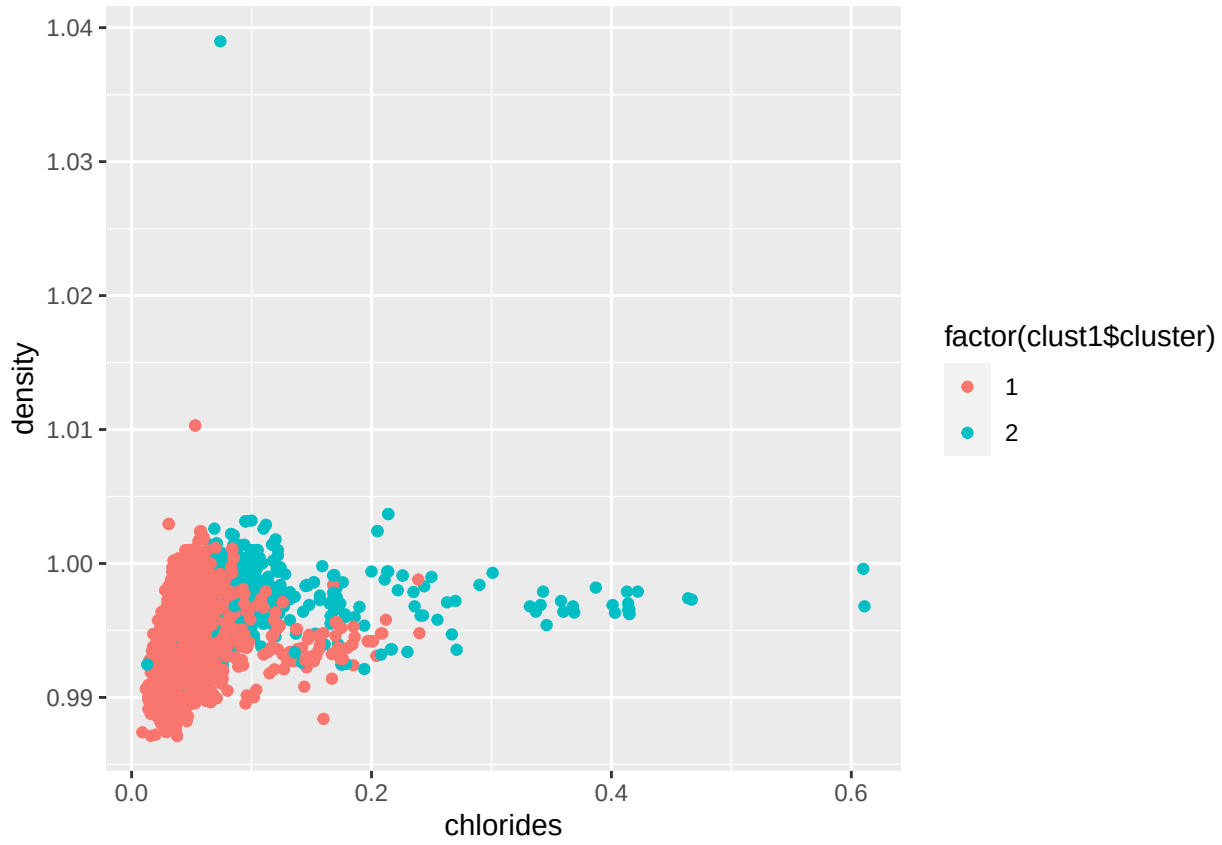
```
##  [1] "total.sulfur.dioxide" "free.sulfur.dioxide"  "residual.sugar"
##  [4] "citric.acid"          "density"              "alcohol"
##  [7] "pH"                   "fixed.acidity"        "chlorides"
## [10] "sulphates"            "volatile.acidity"
```

```
o2 = order(loadings[,2], decreasing=TRUE)
colnames(X)[head(o2,25)]
```

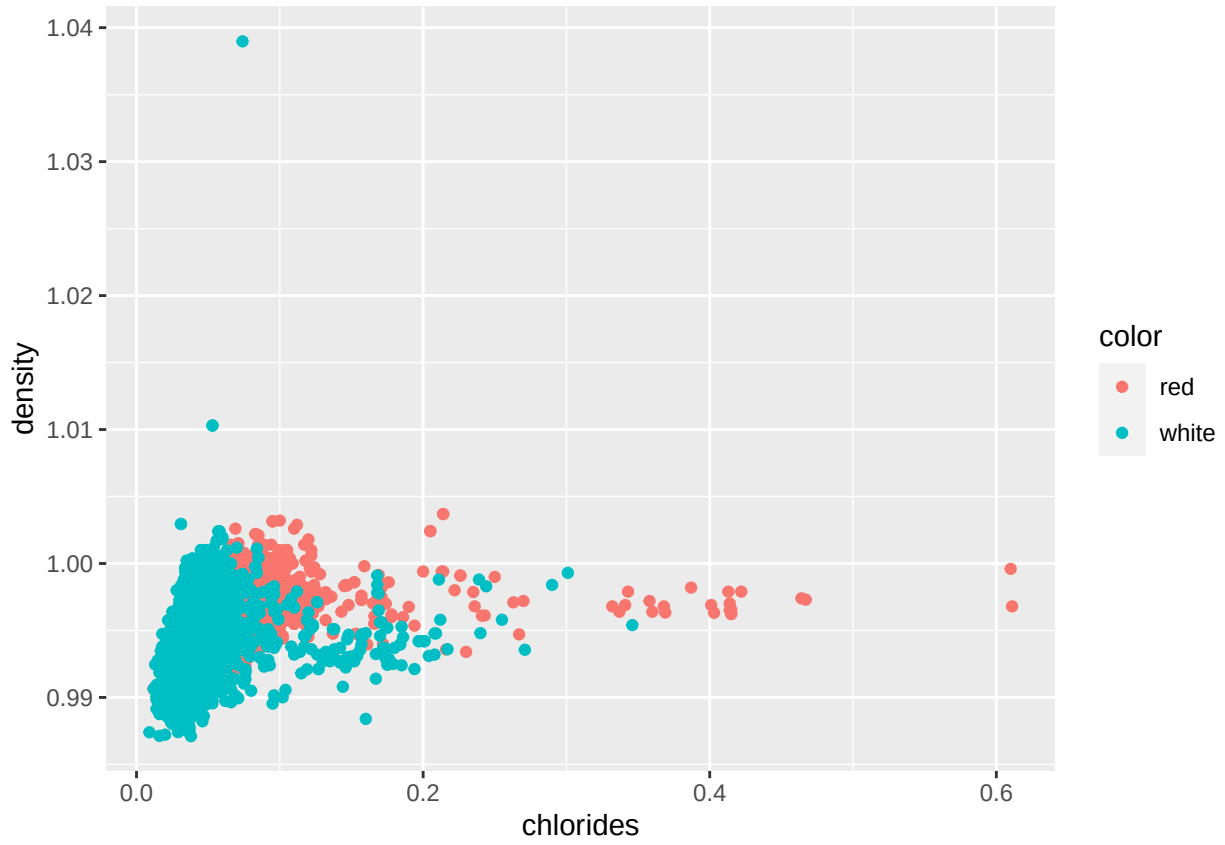
```
##  [1] "density"              "fixed.acidity"        "residual.sugar"
##  [4] "chlorides"            "sulphates"            "citric.acid"
##  [7] "volatile.acidity"     "total.sulfur.dioxide" "free.sulfur.dioxide"
## [10] "pH"                   "alcohol"
```

PCA appears to be able to split the colors fairly well, but the wine quality split seems to be quite a mess.

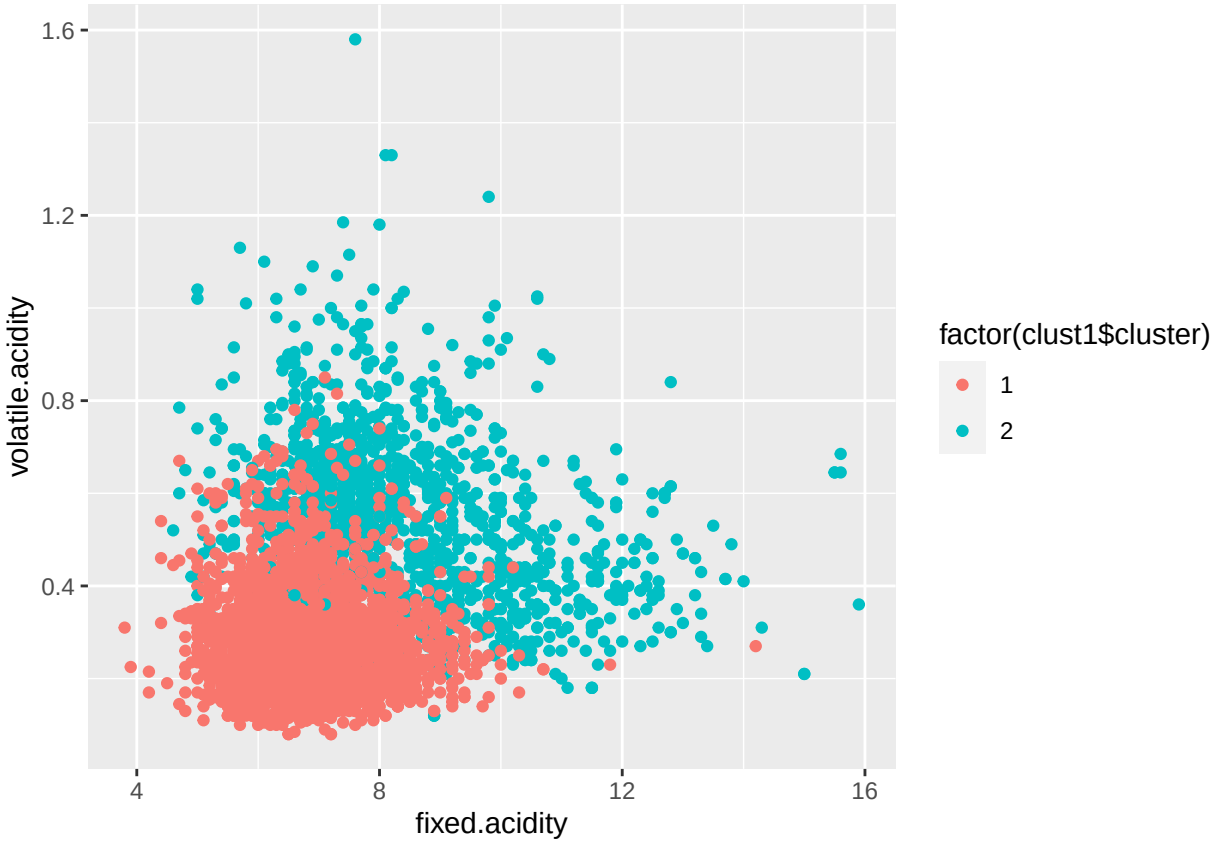
```
clust1 = kmeans(X, 2, nstart=25)
qplot(chlorides, density, data=wine, color=factor(clust1$cluster))
```




```
qplot(chlorides, density, data=wine, color=color)
```



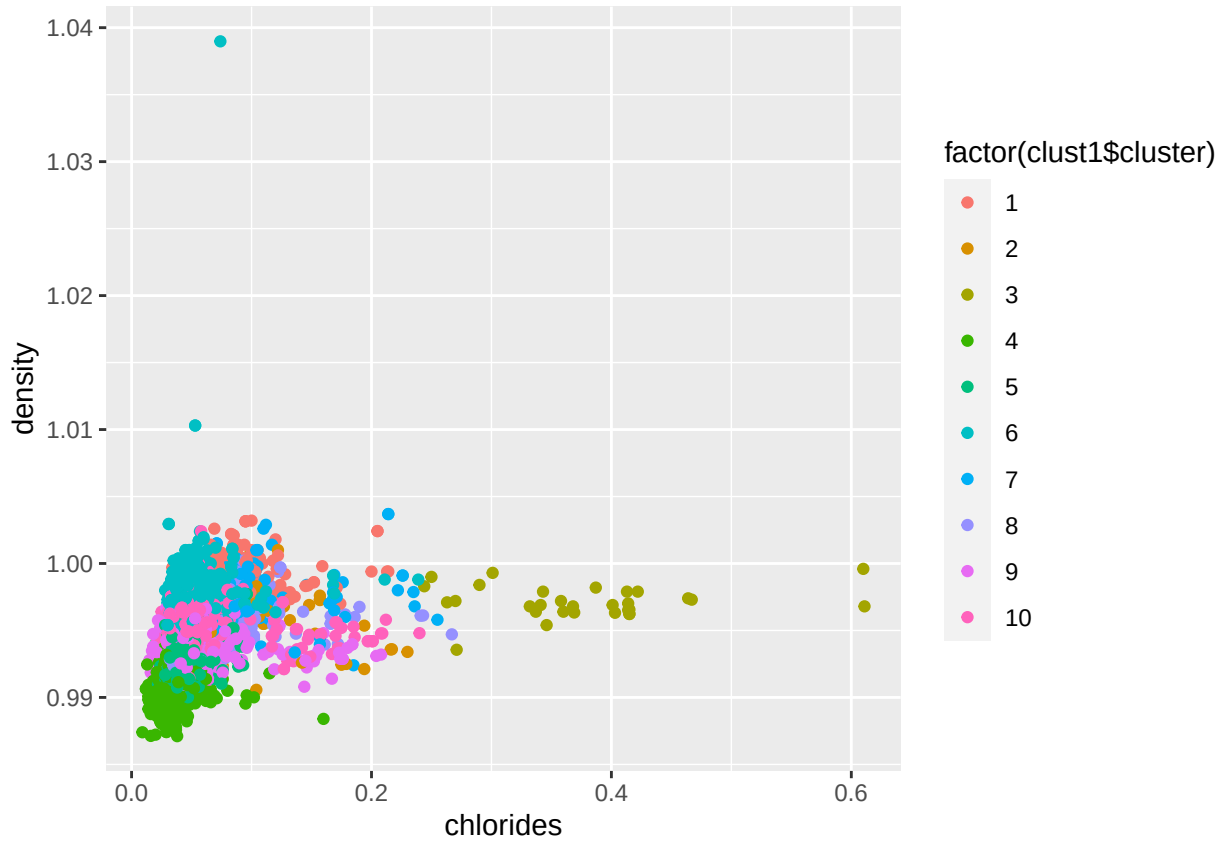

```
qplot(fixed.acidity, volatile.acidity, data=wine, color=factor(clust1$cluster))
```



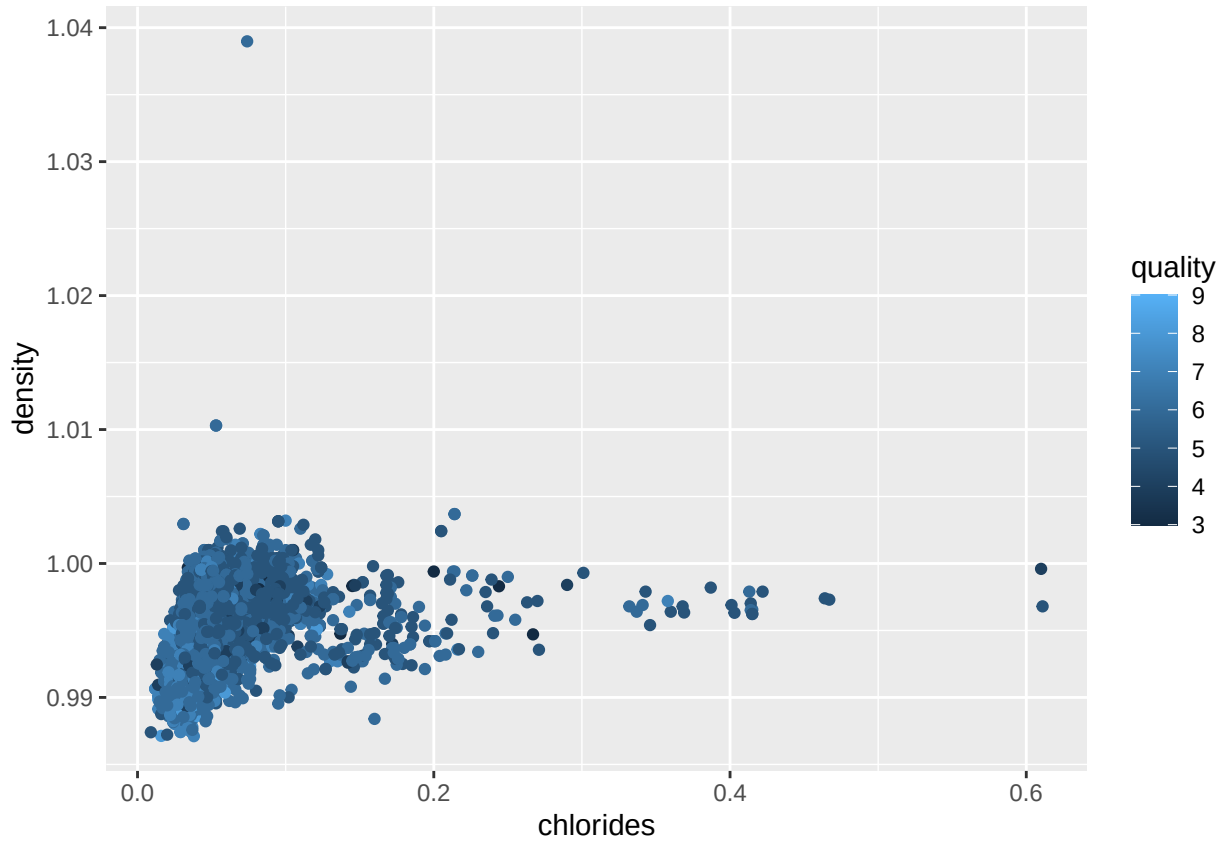

```
qplot(fixed.acidity, volatile.acidity, data=wine, color=color)
```

Using k means, it does seem like red and white are distinguishable. It's not perfect but the overall distribution shapes of the clusters mostly match the distributions of red and white wines. The split seems most obvious comparing the acidities of the wines.

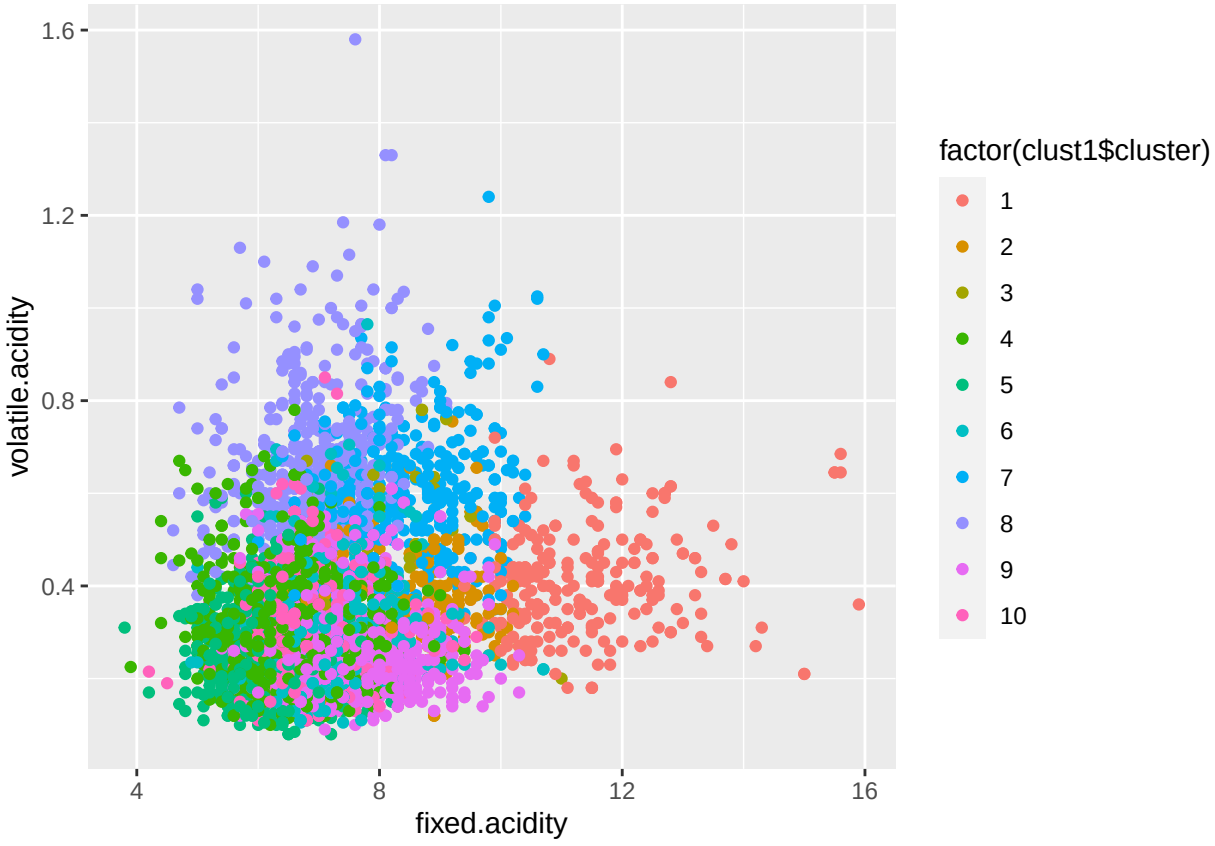
```
clust1 = kmeans(X, 10, nstart=25)
qplot(chlorides, density, data=wine, color=factor(clust1$cluster))
```



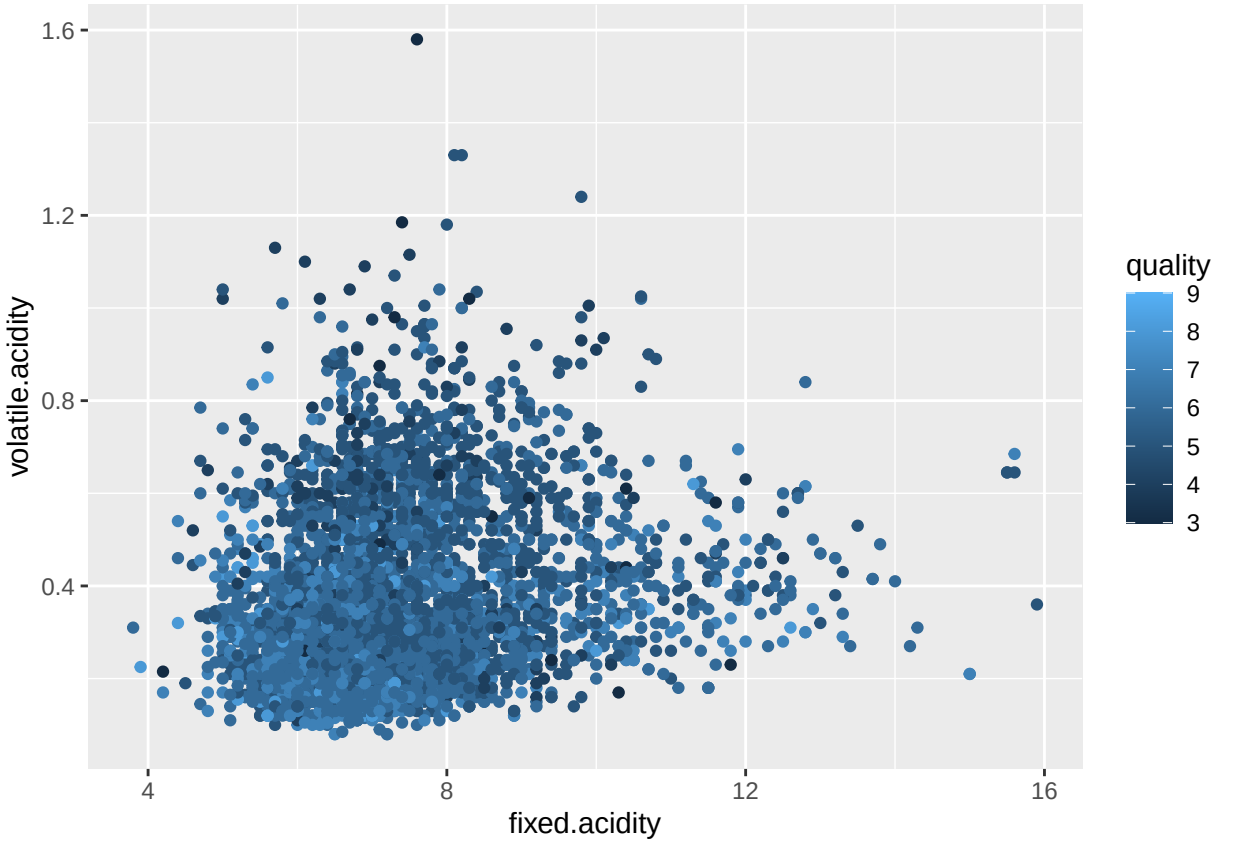

```
qplot(chlorides, density, data=wine, color=quality)
```




```
qplot(fixed.acidity, volatile.acidity, data=wine, color=factor(clust1$cluster))
```




```
qplot(fixed.acidity, volatile.acidity, data=wine, color=quality)
```



Like PCA, K means also seems to be unable to discern the quality of the wines. The qualities also seem to be quite mixed regardless of density or acidity or so on, so the could be other factors(such as simple opinion) at play that make the task less predictable.

Overall, while the plot from PCA seems more clear, K means does actually split the data into groups, while PCA was just colored using the truth values. If I could only use one then I would probably go with K means, because I think the split from PCA might be less obvious without using the truth values from the colors. Something else to try is to do K means on the result of the PCA components but K means does seem to be doing fairly well on its own already.

Market segmentation

Consider the data in social_marketing.csv. This was data collected in the course of a market-research study using followers of the Twitter account of a large consumer brand that shall remain nameless---let's call it "NutrientH20" just to have a label. The goal here was for NutrientH20 to understand its social-media audience a little bit better, so that it could hone its messaging a little more sharply.

A bit of background on the data collection: the advertising firm who runs NutrientH20's online-advertising campaigns took a sample of the brand's Twitter followers. They collected every Twitter post ("tweet") by each of those followers over a seven-day period in June 2014. Every post was examined by a human annotator contracted through Amazon's Mechanical Turk service. Each tweet was categorized based on its content using a pre-specified scheme of 36 different categories, each representing a broad area of interest (e.g. politics, sports, family, etc.) Annotators were allowed to classify a post as belonging to more than one category. For example, a hypothetical post such as "I'm really excited to see grandpa go wreck shop in his geriatic soccer league this Sunday!" might be categorized as both "family" and "sports." You get the picture.

Each row of social_marketing.csv represents one user, labeled by a random (anonymous, unique) 9-digit

alphanumeric code. Each column represents an interest, which are labeled along the top of the data file. The entries are the number of posts by a given user that fell into the given category. Two interests of note here are "spam" (i.e. unsolicited advertising) and "adult" (posts that are pornographic, salacious, or explicitly sexual). There are a lot of spam and pornography "bots" on Twitter; while these have been filtered out of the data set to some extent, there will certainly be some that slip through. There's also an "uncategorized" label. Annotators were told to use this sparingly, but it's there to capture posts that don't fit at all into any of the listed interest categories. (A lot of annotators may used the "chatter" category for this as well.) Keep in mind as you examine the data that you cannot expect perfect annotations of all posts. Some annotators might have simply been asleep at the wheel some, or even all, of the time! Thus there is some inevitable error and noisiness in the annotation process.

Your task to is analyze this data as you see fit, and to prepare a concise report for NutrientH20 that identifies any interesting market segments that appear to stand out in their social-media audience. You have complete freedom in deciding how to pre-process the data and how to define "market segment." (Is it a group of correlated interests? A cluster? A latent factor? Etc.) Just use the data to come up with some interesting, well-supported insights about the audience, and be clear about what you did.

```
social = read.csv('Data/social_marketing.csv', header=TRUE)
head(social)
```

```
##           X chatter current_events travel photo_sharing uncategorized tv_film
## 1 hmjoe4g3k       2              0      2             2             2       1
## 2 clk1m5w8s       3              3      2             1             1       1
## 3 jcsovtak3       6              3      4             3             1       5
## 4 3oeb4hiln       1              5      2             2             0       1
## 5 fd75x1vgk       5              2      0             6             1       0
## 6 h6nvj91yp       6              4      2             7             0       1
##   sports_fandom politics food family home_and_garden music news online_gaming
## 1             1        0    4      1               2     0    0              0
## 2             4        1    2      2               1     0    0              0
## 3             0        2    1      1               1     1    1              0
## 4             0        1    0      1               0     0    0              0
## 5             0        2    0      1               0     0    0              3
## 6             1        0    2      1               1     1    0              0
##   shopping health_nutrition college_uni sports_playing cooking eco computers
## 1        1               17           0              2       5   1         1
## 2        0                0           0              1       0   0         0
## 3        2                0           0              0       2   1         0
## 4        0                0           1              0       0   0         0
## 5        2                0           4              0       1   0         1
## 6        5                0           0              0       0   0         1
##   business outdoors crafts automotive art religion beauty parenting dating
## 1        0        2      1          0   0        1      0         1      1
## 2        1        0      2          0   0        0      0         0      1
## 3        0        0      2          0   8        0      1         0      1
## 4        1        0      3          0   2        0      1         0      0
## 5        0        1      0          0   0        0      0         0      0
## 6        1        0      0          1   0        0      0         0      0
##   school personal_fitness fashion small_business spam adult
## 1      0               11       0              0    0     0
## 2      4                0       0              0    0     0
## 3      0                0       1              0    0     0
## 4      0                0       0              0    0     0
## 5      0                0       0              1    0     0
```

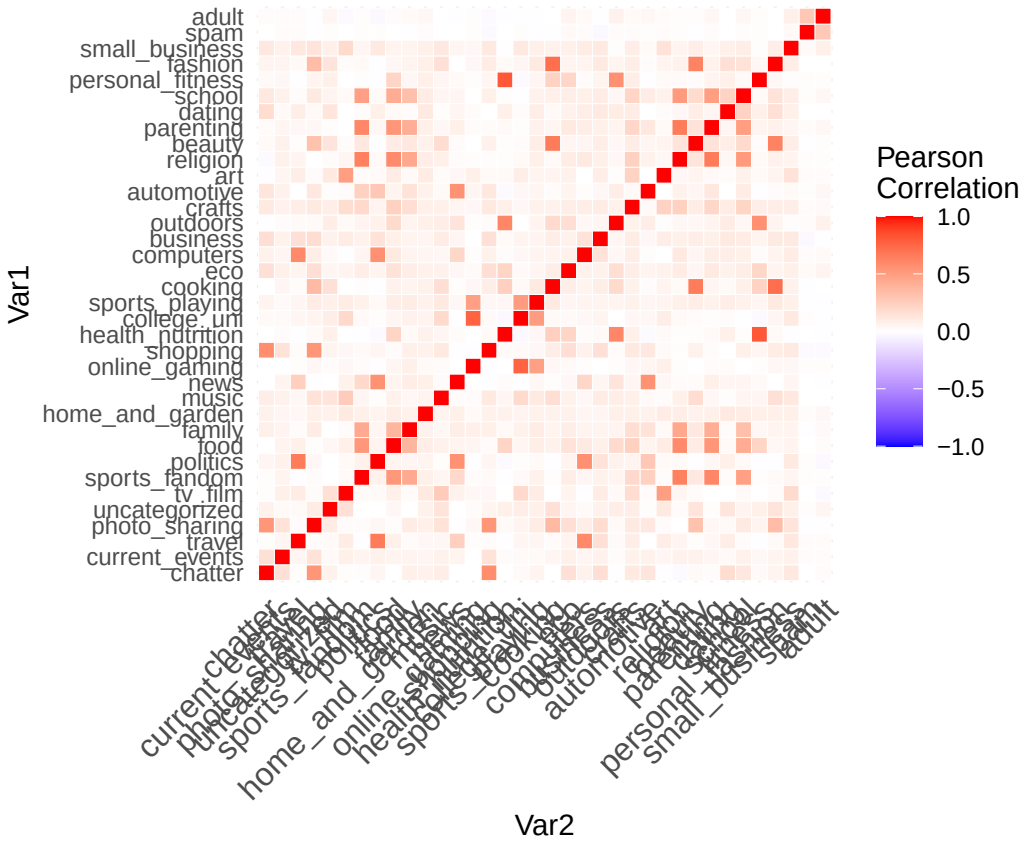
```
## 6      0                 0       0              0    0     0
```

Since X is an ID, it shouldn't be used for prediction so drop it. Everything else appears to be numeric data.

```
social = subset(social, select = -c(X))
#http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visuali
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
cormat = melt(round(cor(social),2)) #correlation matrix
ggplot(data = cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()
```



Looking at the correlation heatmap, it seems most of the data is relatively uncorrelated with each other, and none are negatively correlated. The positively correlated features are:

- fashion, beauty, and eco
- sports_playing, college_uni, and online_gaming
- school, parenting, religon, family, food, and sports_fandom
- health_nutrition, outdoors, and personal_fitness
- shopping, photo_sharing, and chatter
- politics, travel, and computers
- adult, spam
- automotive, news, and politics

So it looks like there should be at least 8 main clusters. Let's round it to 10 to account for the remaining features.

```
library(ClusterR)  # for kmeans++
```

```
## Loading required package: gtools
```

```
##
## Attaching package: 'gtools'
```

```
## The following object is masked from 'package:mosaic':
##
##     logit
```

```
X = scale(social, center=TRUE, scale=TRUE) #scale
clust2 = KMeans_rcpp(X, clusters=10, num_init=25, initializer = 'kmeans++')
```

```
#help(KMeans_rcpp) #unsure how to get clusters from Kmeans++
clust1 = kmeans(X, 10, nstart=25) #kmeans
```

```
## Warning: did not converge in 10 iterations
```

```
round(clust1$center,2)
```

```
##    chatter current_events travel photo_sharing uncategorized tv_film
## 1    -0.09          -0.09  -0.03         -0.01         -0.04    0.10
## 2    -0.13          -0.02  -0.15         -0.11          0.17   -0.15
## 3    -0.04           0.18  -0.05          1.24          0.50   -0.14
## 4    -0.08           0.11   3.27         -0.11         -0.09   -0.07
## 5    -0.38          -0.20  -0.22         -0.43         -0.18   -0.22
## 6     1.48           0.31  -0.20          1.08          0.07   -0.14
## 7    -0.13           0.11  -0.10         -0.10         -0.10   -0.10
## 8    -0.07           0.07  -0.19         -0.22         -0.09   -0.01
## 9    -0.12           0.33   0.22         -0.08          0.69    2.75
## 10    0.07           0.28   0.29         -0.09          0.11   -0.12
##    sports_fandom politics  food family home_and_garden music  news
## 1          -0.13    -0.18 -0.09   0.21            0.07 -0.05 -0.19
## 2          -0.20    -0.20  0.45  -0.09            0.16  0.00 -0.07
```

```
## 3          -0.21    -0.13 -0.20   0.03           0.14  0.55 -0.08
## 4          -0.21     3.12  0.16  -0.09           0.05 -0.04  1.14
## 5          -0.32    -0.30 -0.36  -0.30          -0.20 -0.23 -0.31
## 6          -0.20    -0.15 -0.30  -0.05           0.10  0.13 -0.27
## 7           2.09    -0.22  1.85   1.52           0.16  0.02 -0.11
## 8           0.67     1.23 -0.15   0.24           0.16 -0.09  2.66
## 9          -0.12    -0.09  0.15  -0.11           0.33  1.00  0.00
## 10          0.14     0.15  0.04  -0.06           0.24  0.01  0.00
##    online_gaming shopping health_nutrition college_uni sports_playing cooking
## 1           3.62    -0.14            -0.18        3.31           2.15   -0.12
## 2          -0.11    -0.06             2.22       -0.21          -0.02    0.41
## 3          -0.02     0.20            -0.07       -0.02           0.20    2.82
## 4          -0.17    -0.08            -0.17       -0.05           0.04   -0.19
## 5          -0.23    -0.40            -0.31       -0.26          -0.26   -0.33
## 6          -0.16     1.36            -0.24       -0.10          -0.06   -0.23
## 7          -0.08    -0.02            -0.14       -0.13           0.10   -0.10
## 8          -0.12    -0.19            -0.24       -0.19          -0.08   -0.23
## 9          -0.17     0.02            -0.16        0.37           0.14   -0.14
## 10          0.09    -0.24             0.05        0.13          -0.11   -0.06
##      eco computers business outdoors crafts automotive   art religion beauty
## 1  -0.07     -0.08    -0.10    -0.14   0.03       0.07  0.27    -0.19  -0.22
## 2   0.56     -0.08     0.05     1.73   0.06      -0.18 -0.08    -0.17  -0.21
## 3   0.00      0.06     0.23     0.01   0.08       0.01  0.00    -0.12   2.64
## 4   0.16      2.91     0.56    -0.04   0.20      -0.13 -0.16     0.12  -0.18
## 5  -0.28     -0.26    -0.25    -0.32  -0.29      -0.31 -0.24    -0.30  -0.27
## 6   0.28     -0.03     0.34    -0.25   0.08       0.07 -0.20    -0.26  -0.19
## 7   0.18      0.09     0.10    -0.07   0.70       0.12 -0.03     2.29   0.32
## 8  -0.10     -0.19    -0.12     0.31  -0.16       2.59 -0.16    -0.18  -0.18
## 9   0.10     -0.15     0.35    -0.09   0.74      -0.23  2.64     0.01   0.01
## 10  0.45      0.30    -0.35     0.30   0.22       0.12  0.33     0.12  -0.10
##    parenting dating school personal_fitness fashion small_business  spam adult
## 1      -0.13  -0.01  -0.23            -0.18   -0.07          0.13 -0.08 -0.02
## 2      -0.09   0.19  -0.17             2.16   -0.10         -0.12 -0.08  0.02
## 3      -0.06   0.05   0.17            -0.04    2.73          0.16 -0.08  0.00
## 4       0.02   0.31  -0.11            -0.15   -0.17          0.40 -0.08 -0.14
## 5      -0.32  -0.17  -0.32            -0.33   -0.29         -0.21 -0.08 -0.01
## 6      -0.20   0.30   0.09            -0.19   -0.07          0.17 -0.08 -0.03
## 7       2.17   0.02   1.69            -0.09    0.01          0.09 -0.08 -0.01
## 8       0.04  -0.03   0.02            -0.23   -0.21         -0.16 -0.08 -0.11
## 9      -0.20  -0.06  -0.05            -0.15   -0.02          0.79 -0.08 -0.04
## 10      0.19  -0.01   0.09             0.12   -0.02          0.31 12.42  3.75
```

Looking at the high(>1) values for the 10 clusters, they are:

1. tv_film, music, art
2. sports_fandom, food, family, religion, parenting, school
3. none
4. spam, adult
5. photo_sharing, cooking, fashion
6. online_gaming, college_uni, sports_playing
7. politics, news, automotive
8. politics, news, computers
9. chatter, shopping,
10. health_nutrition, outdoors, personal_fitness

Overall, these are fairly similar to the results from the correlation heatmap. Since NutientH20 seems to sell some kind of vitamin water, I would recommend that they target clusters 6, and 10 which have to do with sports and fitness/nutrition respectively. Secondary targets could be clusters 2, 5, and 9 which have less of a direct interest but still could be potential customers. Finally, I would avoid or reduce advertising to clusters 1, 3, 4, 7, and 8 since they don't seem to have much interest in the product.

The Reuters corpus

Revisit the Reuters C50 text corpus that we briefly explored in class. Your task is simple: tell an interesting story, anchored in some analytical tools we have learned in this class, using this data. For example:

- you could cluster authors or documents and tell a story about what you find.
- you could look for common factors using PCA.
- you could train a predictive model and assess its accuracy. (Yes, this is a supervised learning task, but it potentially draws on a lot of what you know about unsupervised learning, since constructing features for a document might involve dimensionality reduction.)
- you could do anything else that strikes you as interesting with this data.

Describe clearly what question you are trying to answer, what models you are using, how you pre-processed the data, and so forth. Make sure you include at least *one* really interesting plot (although more than one might be necessary, depending on your question and approach.)

Format your write-up in the following sections, some of which might be quite short:

- Question: What question(s) are you trying to answer?
- Approach: What approach/statistical tool did you use to answer the questions?
- Results: What evidence/results did your approach provide to answer the questions? (E.g. any numbers, tables, figures as appropriate.)
- Conclusion: What are your conclusions about your questions? Provide a written interpretation of your results, understandable to stakeholders who might plausibly take an interest in this data set.

Regarding the data itself: In the C50train directory, you have 50 articles from each of 50 different authors (one author per directory). Then in the C50test directory, you have another 50 articles from each of those same 50 authors (again, one author per directory). This train/test split is obviously intended for building predictive models, but to repeat, you need not do that on this problem. You can tell any story you want using any methods you want. Just make it compelling!

Note: if you try to build a predictive model, you will need to figure out a way to deal with words in the test set that you never saw in the training set. This is a nontrivial aspect of the modeling exercise. (E.g. you might simply ignore those new words.)

This question will be graded according to three criteria:

1. the overall "interesting-ness" of your question and analysis.
2. the clarity of your description. We will be asking ourselves: could your analysis be reproduced by a competent data scientist based on what you've said? (That's good.) Or would that person have to wade into the code in order to understand what, precisely, you've done? (That's bad.)
3. technical correctness (i.e. did you make any mistakes in execution or interpretation?)

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
```

```
## The following object is masked from 'package:mosaic':
##
##     inspect
```

```
library(tidyverse)
library(slam)
library(proxy)
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
readerPlain = function(fname){
                readPlain(elem=list(content=readLines(fname)),
                          id=fname, language='en') }
file_list = Sys.glob('Data/ReutersC50/C50train/SarahDavison/*.txt')
simon = lapply(file_list, readerPlain) #reads in data
# Clean up the file names
mynames = file_list %>%
    { strsplit(., '/', fixed=TRUE) } %>%
    { lapply(., tail, n=2) } %>%
    { lapply(., paste0, collapse = '') } %>%
    unlist
names(simon) = mynames
```

```
documents_raw = Corpus(VectorSource(simon))
## Some pre-processing/tokenization steps.
## tm_map just maps some function to every document in the corpus
my_documents = documents_raw
my_documents = tm_map(my_documents, content_transformer(tolower)) # make everything lowercase
```

```
## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(tolower)):
## transformation drops documents
```

```
my_documents = tm_map(my_documents, content_transformer(removeNumbers)) # remove numbers
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removeNumbers)): transformation drops documents
```

```
my_documents = tm_map(my_documents, content_transformer(removePunctuation)) # remove punctuation
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removePunctuation)): transformation drops documents
```

```
my_documents = tm_map(my_documents, content_transformer(stripWhitespace)) ## remove excess white-space
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(stripWhitespace)): transformation drops documents
```

```
## Remove stopwords.  Always be careful with this!
#stopwords("en")
#stopwords("SMART")
#?stopwords
my_documents = tm_map(my_documents, content_transformer(removeWords), stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(removeWords), :
## transformation drops documents
```

```
## create a doc-term-matrix
DTM_simon = DocumentTermMatrix(my_documents)
DTM_simon # some basic summary statistics
```

```
## <<DocumentTermMatrix (documents: 50, terms: 4200)>>
## Non-/sparse entries: 12449/197551
## Sparsity           : 94%
## Maximal term length: 39
## Weighting          : term frequency (tf)
```

```
class(DTM_simon)  # a special kind of sparse matrix format
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
## You can inspect its entries...
inspect(DTM_simon[1:10,1:20])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 20)>>
## Non-/sparse entries: 52/148
## Sparsity           : 74%
## Maximal term length: 13
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs acquisition already also alternative asia asian asias assets banks billion
##   1            1       1    1           1    3     1     1      2     1       3
##   10           1       0    0           0    2     1     1      0     1       0
##   2            2       0    1           0    0     0     0      0     0       0
##   3            0       0    1           0    2     0     1      0     0       0
##   4            0       0    0           0    0     1     0      2     1       1
##   5            0       0    1           0    1     1     0      0     0       0
##   6            0       1    0           1    1     0     0      0     0       0
##   7            0       1    1           0    1     0     0      0     0       1
##   8            0       0    1           1    0     0     0      0     0       1
##   9            0       0    1           1    0     0     0      0     0       1
```

```
## ...find words with greater than a min count...
findFreqTerms(DTM_simon, 50)
```

```
##  [1] "also"
##  [2] "asia"
##  [3] "billion"
##  [4] "character"
##  [5] "china"
##  [6] "datareutersccctrainsarahdavisonnewsmltxt"
##  [7] "datetimestamp"
##  [8] "description"
##  [9] "heading"
## [10] "hong"
## [11] "hour"
## [12] "isdst"
## [13] "kong"
## [14] "language"
## [15] "listcontent"
## [16] "mday"
## [17] "meta"
## [18] "min"
## [19] "mon"
## [20] "new"
## [21] "one"
## [22] "origin"
## [23] "percent"
## [24] "said"
## [25] "wday"
## [26] "will"
## [27] "yday"
## [28] "year"
```

```
## [29] "can"
## [30] "fund"
## [31] "investment"
## [32] "last"
## [33] "management"
## [34] "market"
## [35] "domestic"
## [36] "foreign"
## [37] "funds"
## [38] "mutual"
## [39] "economic"
## [40] "markets"
## [41] "many"
## [42] "chinese"
## [43] "kongs"
```

```
## ...or find words whose count correlates with a specified word.
findAssocs(DTM_simon, "genetic", .5)
```

```
## $genetic
## numeric(0)
```

```
DTM_simon = removeSparseTerms(DTM_simon, 0.95)
DTM_simon # now ~ 1000 terms (versus ~3000 before)
```

```
## <<DocumentTermMatrix (documents: 50, terms: 1218)>>
## Non-/sparse entries: 8644/52256
## Sparsity           : 86%
## Maximal term length: 39
## Weighting          : term frequency (tf)
```

```
# construct TF IDF weights
tfidf_simon = weightTfIdf(DTM_simon)

####
# Compare documents
####
inspect(tfidf_simon[1,])
```

```
## <<DocumentTermMatrix (documents: 1, terms: 1218)>>
## Non-/sparse entries: 126/1092
## Sparsity           : 90%
## Maximal term length: 39
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##     Terms
## Docs       cash    company      deal   electric     option       owns     power
##    1 0.06565507 0.04136264 0.0492413 0.03656661 0.03656661 0.03656661 0.1071834
##     Terms
## Docs    shares  southern     stake
##    1 0.0383311 0.1346728 0.0275576
```

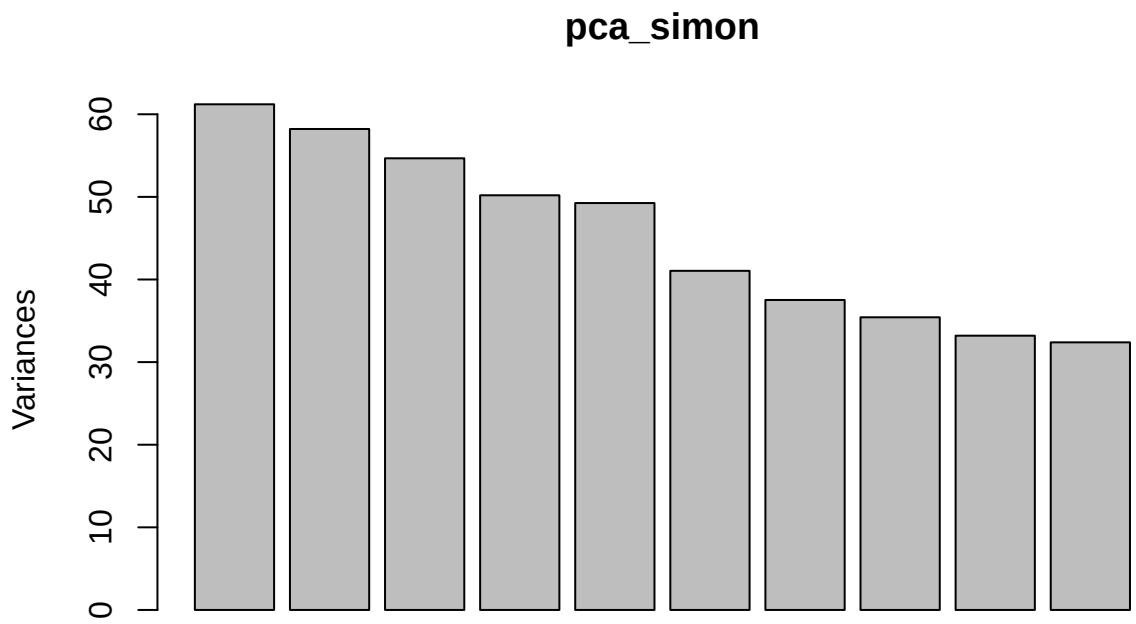
```
X = as.matrix(tfidf_simon)
summary(colSums(X))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.05208 0.07104 0.08575 0.10093 0.61881
```

```
scrub_cols = which(colSums(X) == 0)
X = X[,-scrub_cols] #remove 0 frequencies
dim(X)
```

```
## [1]   50 1200
```

```
pca_simon = prcomp(X, rank=4, scale=TRUE)
plot(pca_simon)
```




```
#observing top words per component
print("Component 1")
```

```
## [1] "Component 1"
```

```
round(pca_simon$rotation[order(abs(pca_simon$rotation[,1]),decreasing=TRUE),1][1:25],3)
```

```
##           kong     regulation           hong     transition         record
##         -0.086          0.083        -0.082         -0.079         -0.077
##          china          wants   requirements      everybody         mutual
##         -0.077         -0.073          0.072         -0.070          0.070
##        travers       offshore       onshore           good            tax
##          0.069          0.069          0.069         -0.069          0.068
##         doesnt         luring      marketing      countries          funds
##         -0.067          0.067          0.067          0.067          0.067
##   institutions       american        chinese infrastructure        beijing
##          0.066         -0.066         -0.065          0.065         -0.065
```

```
print("Component 2")
```

```
## [1] "Component 2"
```

```
round(pca_simon$rotation[order(abs(pca_simon$rotation[,2]),decreasing=TRUE),2][1:25],3)
```

```
##    offshore     inflows    external     travers speculators   marketing
##      -0.091       0.090       0.090      -0.089       0.089      -0.089
##   situation     onshore     kingdom      retail        pace      europe
##       0.088      -0.088      -0.088      -0.088       0.087      -0.086
##      united       eased    taxation   elsewhere    frequent   camdessus
##      -0.086       0.086      -0.085       0.085       0.085       0.085
##         tax        kept     banking      raised appropriate    deficits
##      -0.084       0.084       0.084       0.084      -0.084       0.083
##         net
##      -0.083
```

```
print("Component 3")
```

```
## [1] "Component 3"
```

```
round(pca_simon$rotation[order(abs(pca_simon$rotation[,3]),decreasing=TRUE),2][1:25],3)
```

```
##  inflation    salomon     prices  confident      gains  benchmark economists
##      0.007      0.009      0.004      0.004      0.004      0.006      0.007
##     luxury   consumer   property  supported   downside     export      slump
##      0.002      0.005      0.008      0.005      0.007      0.007      0.007
##      wrote turnaround      fears  stimulate   brothers regardless  austerity
##      0.001      0.007      0.005      0.008      0.013      0.012      0.000
##      proof  forecasts    overall   evidence
##      0.000      0.006      0.007     -0.051
```

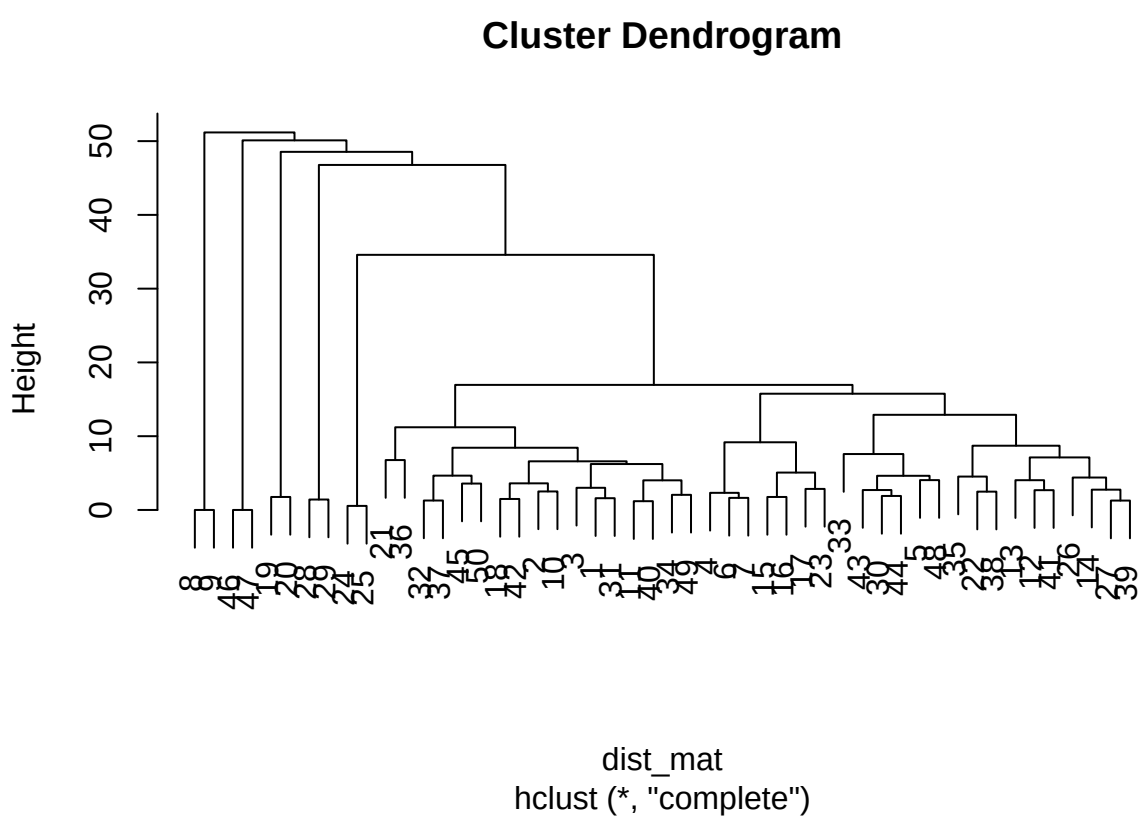
```
print("Component 4")
```

```
## [1] "Component 4"
```

```
round(pca_simon$rotation[order(abs(pca_simon$rotation[,4]),decreasing=TRUE),2][1:25],3)
```

```
##       star    western  according     review   accident      ferry      rival
##     -0.007     -0.007     -0.041     -0.020     -0.007     -0.007      0.000
##      stand   congress      cards       hurt    worried     patten    holiday
##      0.000     -0.019     -0.005     -0.018     -0.018     -0.005     -0.005
##     marked determined       born       rich     seemed   separate   proposed
##     -0.005     -0.002     -0.005     -0.005     -0.005     -0.005     -0.007
##  president     status   attempts  treatment
##     -0.002     -0.019     -0.012     -0.012
```

```
dist_mat = dist(pca_simon$x)
tree_simon = hclust(dist_mat)
plot(tree_simon)
```




```
clust5 = cutree(tree_simon, k=4)
```

```
print("Cluster 1")
```

```
## [1] "Cluster 1"
```

```
which(clust5 == 1)
```

```
##  1  2  3  4  5  6  7 10 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 29 30
##  1  2  3  4  5  6  7 10 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 29 30
## 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 48 49 50
## 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 48 49 50
```

```
print("Cluster 2")
```

```
## [1] "Cluster 2"
```

```
which(clust5 == 2)
```

```
## 8 9
## 8 9
```

```
print("Cluster 3")
```

```
## [1] "Cluster 3"
```

```
which(clust5 == 3)
```

```
## 19 20
## 19 20
```

```
print("Cluster 4")
```

```
## [1] "Cluster 4"
```

```
which(clust5 == 4)
```

```
## 46 47
## 46 47
```

```
print("Cluster 1")
```

```
## [1] "Cluster 1"
```

```
content(simon[[1]])[1]
```

```
## [1] "The largest electricity provider in the United States marched into Asia on Thursday with a US$2
```

```
content(simon[[50]])[1]
```

```
## [1] "Hong Kong’s hefty cash reserves offer a comfortable cushion for children born in 1997, the year
```

```
print("Cluster 2")
```

```
## [1] "Cluster 2"
```

```
content(simon[[8]])[1]
```

```
## [1] "Zero tax and limited regulation are luring billions of dollars into offshore mutual funds, spur
```

```
content(simon[[9]])[1]
```

```
## [1] "Zero tax and limited regulation are luring billions of dollars into offshore mutual funds, spur
```

```
print("Cluster 3")
```

```
## [1] "Cluster 3"
```

```
content(simon[[19]])[1]
```

```
## [1] "Hong Kong business groups hit back on Wednesday at attempts to link China's permanent most favo
```

```
content(simon[[20]])[1]
```

```
## [1] "Hong Kong business groups hit back on Wednesday at U.S. attempts to link China's most favoured
```

```
print("Cluster 4")
```

```
## [1] "Cluster 4"
```

```
content(simon[[46]])[1]
```

```
## [1] "Thailand's banking sector crisis brought repeated calls for an upgrade of Asia's immature finan
```

```
content(simon[[47]])[1]
```

```
## [1] "Thailand's banking sector crisis brought repeated calls for an upgrade of Asia's immature finan
```

Question:

Google News has several categories - US, World, Local, Business, Technology, Entertainment, Sports, Science, and Health. Would it be possible to cluster articles into similar categories without being given labels, only the text?

Approach:

First we should clearly define our target categories. It is probably difficult to fit smaller, more specific categories compared to large ones, so we could try making broader categories. So we could use the same categories as above and try to combine similar ones. For example, US, World, and Local news could perhaps be lumped into a "Government" category. Technology, science, and health are pretty similar so they could be combined into "Science". Finally, sports is more or less a form of entertainment, so combine those as well. In the end, we have Government, Business, Science, and Entertainment as our target categories.

After reading in the data(Sarah Davison's articles), we have to first do preprocessing. So after making everything lowercase, removing numbers, punctuation, whitespace, and stopwords, the data is now stored in a dataframe and so it is structured and usable. Other preprocessing steps could include using n-grams of 2 or 3 and seeing which works better, lemmatizing or stemming to combine words with the same root, matching parts of speech, and combining common words from phrases. But we didn't cover how to do these in R in class so we will move forward with unstemmed unigram words.

So now we have a document term matrix, which just basically counts the number of occurrences per word per article in each row. This could be used directly, but if we want to do clustering, it has to be scaled at a minimum. Term frequency does this by dividing the count of each word by the total number of words in the document, thus limiting the range of each value from 0 to 1. However, some words could just occur more often than others, and thus skew the results. So using inverse document frequency, words that occur everywhere are limited in their impact.

After TFIDF, the data is ready to be used. We could cluster it directly, or use PCA for dimensionality reduction. However, even after removing sparse terms, there are still 1105 features. I decided to use a rank 4 PCA to try to match the 4 target categories. We can see, however, that the weights for components 3 and 4 are much smaller than for 1 and 2, which is presumably why we went with rank 2 in class. Overall, component 1 seems to deal with foreign countries, component 2 with the european economy, component 3 with the general economy, and component 4 with vacation/travel.

Finally, we used hierarchical clustering with k=4 to see what the clusters are.

Results:

Cluster 1 seems to have to do with Hong Kong, Cluster 2 with the Cayman Islands, Cluster 3 with Hong Kong again, and Cluster 4 with Thailand. Every cluster had to do with foreign affairs.

Conclusion:

Trying to match the target categories did not go very well. It seems that this particular author(along with Peter Humphrey and others that I tried in the background) tend to talk about one specific area and not really about anything else. It does make sense that individual reporters would specialize in certain topics, however, so while clustering on one particular person might not be very helpful, clustering on the full corpus across all of the given authors could perhaps fit the categories better. It still might not work though, if Reuters just covers different material from Google News, so there would be some trial and error to find good categories.

Association rule mining

Revisit the notes on association rule mining and the R example on music playlists: playlists.R and playlists.csv. Then use the data on grocery purchases in groceries.txt and find some interesting association rules for these shopping baskets. The data file is a list of shopping baskets: one person's basket for each row, with multiple items per row separated by commas. Pick your own thresholds for lift and confidence; just be clear what these thresholds are and say why you picked them. Do your discovered item sets make sense? Present your discoveries in an interesting and visually appealing way.

Notes:

- This is an exercise in visual and numerical story-telling. Do be clear in your description of what you've done, but keep the focus on the data, the figures, and the insights your analysis has drawn from the data, rather than technical details.
- The data file is a list of baskets: one row per basket, with multiple items per row separated by commas. You'll have to cobble together your own code for processing this into the format expected by the "arules" package. This is not intrinsically all that hard, but it is the kind of data-wrangling wrinkle you'll encounter frequently on real problems, where your software package expects data in one format and the data comes in a different format. Figuring out how to bridge that gap is part of the assignment, and so we won't be giving tips on this front.

```
library(tidyverse)
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following object is masked from 'package:gtools':
##
##     permute

## The following object is masked from 'package:mosaic':
##
##     compare

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(arules)  # has a big ecosystem of packages built around it
```

```
##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:mosaic':
##
##     inspect, lhs, rhs

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
gro = readLines(paste("Data/groceries.txt", sep = ""))
head(gro)
```

```
## [1] "citrus fruit,semi-finished bread,margarine,ready soups"
## [2] "tropical fruit,yogurt,coffee"
## [3] "whole milk"
## [4] "pip fruit,yogurt,cream cheese ,meat spreads"
## [5] "other vegetables,whole milk,condensed milk,long life bakery product"
## [6] "whole milk,butter,yogurt,rice,abrasive cleaner"
```

```
grocols = unique(unlist(strsplit(gro,","))) #columns
length(grocols)
```

```
## [1] 169
```

```
df = c()
for(line in gro){ #for row
  newline = rep(0,169) #make new list
  for(i in strsplit(line,",")){ #for item in line
    newline[match(i,grocols)]=1 #set to 1
  }
  df <- rbind(df, newline) #add newline to df
}
df = data.frame(df)
colnames(df) = grocols
head(df)
```

```
##           citrus fruit semi-finished bread margarine ready soups tropical fruit
## newline              1                   1         1           1              0
## newline.1            0                   0         0           0              1
## newline.2            0                   0         0           0              0
## newline.3            0                   0         0           0              0
## newline.4            0                   0         0           0              0
## newline.5            0                   0         0           0              0
##           yogurt coffee whole milk pip fruit cream cheese  meat spreads
## newline        0      0          0         0             0            0
## newline.1      1      1          0         0             0            0
## newline.2      0      0          1         0             0            0
## newline.3      1      0          0         1             1            1
## newline.4      0      0          1         0             0            0
## newline.5      1      0          1         0             0            0
##           other vegetables condensed milk long life bakery product butter rice
## newline                  0              0                        0      0    0
## newline.1                0              0                        0      0    0
## newline.2                0              0                        0      0    0
## newline.3                0              0                        0      0    0
## newline.4                1              1                        1      0    0
```

```
## newline.5               0             0                       0      1    1
##           abrasive cleaner rolls/buns UHT-milk bottled beer liquor (appetizer)
## newline                 0          0        0            0                  0
## newline.1               0          0        0            0                  0
## newline.2               0          0        0            0                  0
## newline.3               0          0        0            0                  0
## newline.4               0          0        0            0                  0
## newline.5               1          0        0            0                  0
##           pot plants cereals white bread bottled water chocolate curd flour
## newline            0       0           0             0         0    0     0
## newline.1          0       0           0             0         0    0     0
## newline.2          0       0           0             0         0    0     0
## newline.3          0       0           0             0         0    0     0
## newline.4          0       0           0             0         0    0     0
## newline.5          0       0           0             0         0    0     0
##           dishes beef frankfurter soda chicken sugar fruit/vegetable juice
## newline        0    0           0    0       0     0                    0
## newline.1      0    0           0    0       0     0                    0
## newline.2      0    0           0    0       0     0                    0
## newline.3      0    0           0    0       0     0                    0
## newline.4      0    0           0    0       0     0                    0
## newline.5      0    0           0    0       0     0                    0
##           newspapers packaged fruit/vegetables specialty bar butter milk pastry
## newline            0                         0             0           0      0
## newline.1          0                         0             0           0      0
## newline.2          0                         0             0           0      0
## newline.3          0                         0             0           0      0
## newline.4          0                         0             0           0      0
## newline.5          0                         0             0           0      0
##           processed cheese detergent root vegetables frozen dessert
## newline                 0         0              0              0
## newline.1               0         0              0              0
## newline.2               0         0              0              0
## newline.3               0         0              0              0
## newline.4               0         0              0              0
## newline.5               0         0              0              0
##           sweet spreads salty snack waffles candy bathroom cleaner canned beer
## newline               0           0       0     0                0           0
## newline.1             0           0       0     0                0           0
## newline.2             0           0       0     0                0           0
## newline.3             0           0       0     0                0           0
## newline.4             0           0       0     0                0           0
## newline.5             0           0       0     0                0           0
##           sausage brown bread shopping bags beverages hamburger meat spices
## newline         0           0             0         0              0      0
## newline.1       0           0             0         0              0      0
## newline.2       0           0             0         0              0      0
## newline.3       0           0             0         0              0      0
## newline.4       0           0             0         0              0      0
## newline.5       0           0             0         0              0      0
##           hygiene articles napkins pork berries whipped/sour cream
## newline                 0       0    0       0                 0
## newline.1               0       0    0       0                 0
## newline.2               0       0    0       0                 0
```

```
## newline.3               0       0    0       0                 0
## newline.4               0       0    0       0                 0
## newline.5               0       0    0       0                 0
##           artif. sweetener grapes dessert zwieback domestic eggs spread cheese
## newline                 0      0       0        0             0             0
## newline.1               0      0       0        0             0             0
## newline.2               0      0       0        0             0             0
## newline.3               0      0       0        0             0             0
## newline.4               0      0       0        0             0             0
## newline.5               0      0       0        0             0             0
##           misc. beverages hard cheese cat food ham turkey baking powder
## newline                0           0        0   0      0             0
## newline.1              0           0        0   0      0             0
## newline.2              0           0        0   0      0             0
## newline.3              0           0        0   0      0             0
## newline.4              0           0        0   0      0             0
## newline.5              0           0        0   0      0             0
##           pickled vegetables oil chewing gum chocolate marshmallow ice cream
## newline                   0   0           0                     0         0
## newline.1                 0   0           0                     0         0
## newline.2                 0   0           0                     0         0
## newline.3                 0   0           0                     0         0
## newline.4                 0   0           0                     0         0
## newline.5                 0   0           0                     0         0
##           frozen vegetables canned fish seasonal products curd cheese
## newline                  0           0                 0           0
## newline.1                0           0                 0           0
## newline.2                0           0                 0           0
## newline.3                0           0                 0           0
## newline.4                0           0                 0           0
## newline.5                0           0                 0           0
##           red/blush wine frozen potato products specialty fat
## newline                0                      0            0
## newline.1              0                      0            0
## newline.2              0                      0            0
## newline.3              0                      0            0
## newline.4              0                      0            0
## newline.5              0                      0            0
##           specialty chocolate candles flower (seeds) sparkling wine salt
## newline                     0       0              0              0    0
## newline.1                   0       0              0              0    0
## newline.2                   0       0              0              0    0
## newline.3                   0       0              0              0    0
## newline.4                   0       0              0              0    0
## newline.5                   0       0              0              0    0
##           frozen meals canned vegetables onions herbs white wine brandy
## newline              0                 0      0     0          0      0
## newline.1            0                 0      0     0          0      0
## newline.2            0                 0      0     0          0      0
## newline.3            0                 0      0     0          0      0
## newline.4            0                 0      0     0          0      0
## newline.5            0                 0      0     0          0      0
##           photo/film sliced cheese pasta softener cling film/bags fish
## newline            0             0     0        0               0    0
```

```
## newline.1          0             0     0        0              0    0
## newline.2          0             0     0        0              0    0
## newline.3          0             0     0        0              0    0
## newline.4          0             0     0        0              0    0
## newline.5          0             0     0        0              0    0
##           male cosmetics canned fruit Instant food products soft cheese honey
## newline               0            0                     0           0     0
## newline.1             0            0                     0           0     0
## newline.2             0            0                     0           0     0
## newline.3             0            0                     0           0     0
## newline.4             0            0                     0           0     0
## newline.5             0            0                     0           0     0
##           dental care popcorn cake bar snack products flower soil/fertilizer
## newline             0       0        0              0                      0
## newline.1           0       0        0              0                      0
## newline.2           0       0        0              0                      0
## newline.3           0       0        0              0                      0
## newline.4           0       0        0              0                      0
## newline.5           0       0        0              0                      0
##           specialty cheese finished products cocoa drinks dog food prosecco
## newline                  0                 0           0       0        0
## newline.1                0                 0           0       0        0
## newline.2                0                 0           0       0        0
## newline.3                0                 0           0       0        0
## newline.4                0                 0           0       0        0
## newline.5                0                 0           0       0        0
##           frozen fish make up remover cleaner female sanitary products
## newline             0               0       0                        0
## newline.1           0               0       0                        0
## newline.2           0               0       0                        0
## newline.3           0               0       0                        0
## newline.4           0               0       0                        0
## newline.5           0               0       0                        0
##           dish cleaner cookware meat tea mustard house keeping products
## newline              0        0    0   0       0                      0
## newline.1            0        0    0   0       0                      0
## newline.2            0        0    0   0       0                      0
## newline.3            0        0    0   0       0                      0
## newline.4            0        0    0   0       0                      0
## newline.5            0        0    0   0       0                      0
##           skin care potato products liquor pet care soups rum salad dressing
## newline          0               0      0        0     0   0              0
## newline.1        0               0      0        0     0   0              0
## newline.2        0               0      0        0     0   0              0
## newline.3        0               0      0        0     0   0              0
## newline.4        0               0      0        0     0   0              0
## newline.5        0               0      0        0     0   0              0
##           sauces vinegar soap hair spray instant coffee roll products
## newline        0       0    0          0              0             0
## newline.1      0       0    0          0              0             0
## newline.2      0       0    0          0              0             0
## newline.3      0       0    0          0              0             0
## newline.4      0       0    0          0              0             0
## newline.5      0       0    0          0              0             0
```

```
##           mayonnaise rubbing alcohol syrup liver loaf baby cosmetics
## newline            0               0     0          0              0
## newline.1          0               0     0          0              0
## newline.2          0               0     0          0              0
## newline.3          0               0     0          0              0
## newline.4          0               0     0          0              0
## newline.5          0               0     0          0              0
##           organic products nut snack kitchen towels frozen chicken light bulbs
## newline                  0         0              0              0           0
## newline.1                0         0              0              0           0
## newline.2                0         0              0              0           0
## newline.3                0         0              0              0           0
## newline.4                0         0              0              0           0
## newline.5                0         0              0              0           0
##           ketchup jam decalcifier nuts/prunes liqueur organic sausage cream
## newline         0   0           0           0       0                0     0
## newline.1       0   0           0           0       0                0     0
## newline.2       0   0           0           0       0                0     0
## newline.3       0   0           0           0       0                0     0
## newline.4       0   0           0           0       0                0     0
## newline.5       0   0           0           0       0                0     0
##           toilet cleaner specialty vegetables baby food pudding powder tidbits
## newline                0                    0         0              0       0
## newline.1              0                    0         0              0       0
## newline.2              0                    0         0              0       0
## newline.3              0                    0         0              0       0
## newline.4              0                    0         0              0       0
## newline.5              0                    0         0              0       0
##           whisky frozen fruits bags cooking chocolate sound storage medium
## newline        0             0    0                 0                    0
## newline.1      0             0    0                 0                    0
## newline.2      0             0    0                 0                    0
## newline.3      0             0    0                 0                    0
## newline.4      0             0    0                 0                    0
## newline.5      0             0    0                 0                    0
##           kitchen utensil preservation products
## newline                 0                    0
## newline.1               0                    0
## newline.2               0                    0
## newline.3               0                    0
## newline.4               0                    0
## newline.5               0                    0
```

```
#https://www.learnbymarketing.com/1043/working-with-arules-transactions-and-read-transactions/
## Cast this variable as a special arules "transactions" class.
# Using a CSV ####
# Create a temporary directory
dir.create(path = "tmp", showWarnings = FALSE)
# Write our data.frame to a csv
write.csv(df, "./tmp/newgroceries.csv")
# Read that csv back in
playtrans <- read.transactions(
        file = "./tmp/newgroceries.csv",
        sep = ",",
```

```
        rm.duplicates = T
)
```

```
## distribution of transactions with duplicates:
##  167
## 9835
```

```
summary(playtrans)
```

```
## transactions as itemMatrix in sparse format with
##  9836 rows (elements/itemsets/transactions) and
##  10006 columns (items) and a density of 0.0003015068
##
## most frequent items:
##                0                1 abrasive cleaner artif. sweetener
##             9835             9835                1                1
##   baby cosmetics          (Other)
##                1            10001
##
## element (itemset/transaction) length distribution:
## sizes
##    3  169
## 9835    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   3.017   3.000 169.000
##
## includes extended item information - examples:
##             labels
## 1                0
## 2                1
## 3 abrasive cleaner
```

```
musicrules = apriori(playtrans,
    parameter=list(support=.005, confidence=.1, maxlen=4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##       4  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[10006 item(s), 9836 transaction(s)] done [0.01s].
```

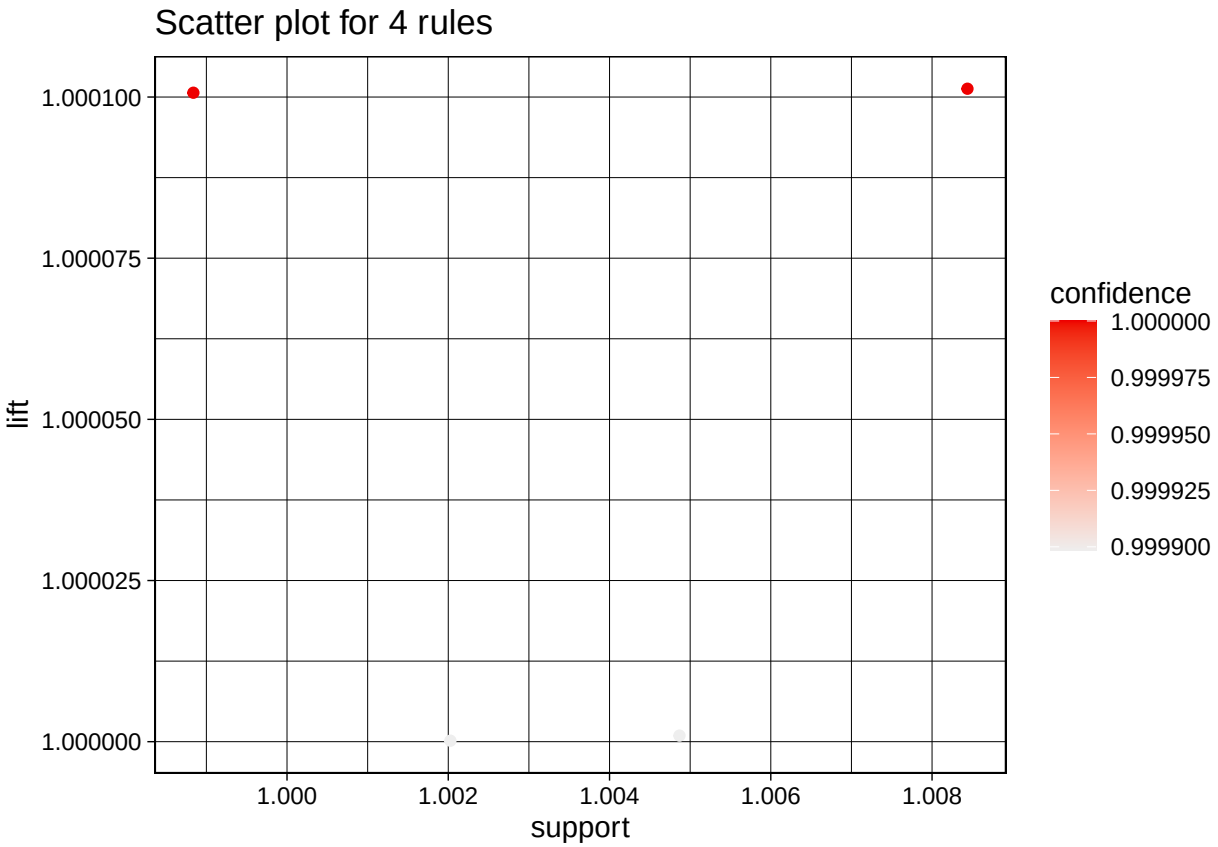
```
## sorting and recoding items ... [2 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [4 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Look at the output... so many rules!
inspect(musicrules)
```

```
##     lhs    rhs support   confidence coverage  lift     count
## [1] {}  => {1} 0.9998983 0.9998983  1.0000000 1.000000 9835
## [2] {}  => {0} 0.9998983 0.9998983  1.0000000 1.000000 9835
## [3] {1} => {0} 0.9998983 1.0000000  0.9998983 1.000102 9835
## [4] {0} => {1} 0.9998983 1.0000000  0.9998983 1.000102 9835
```

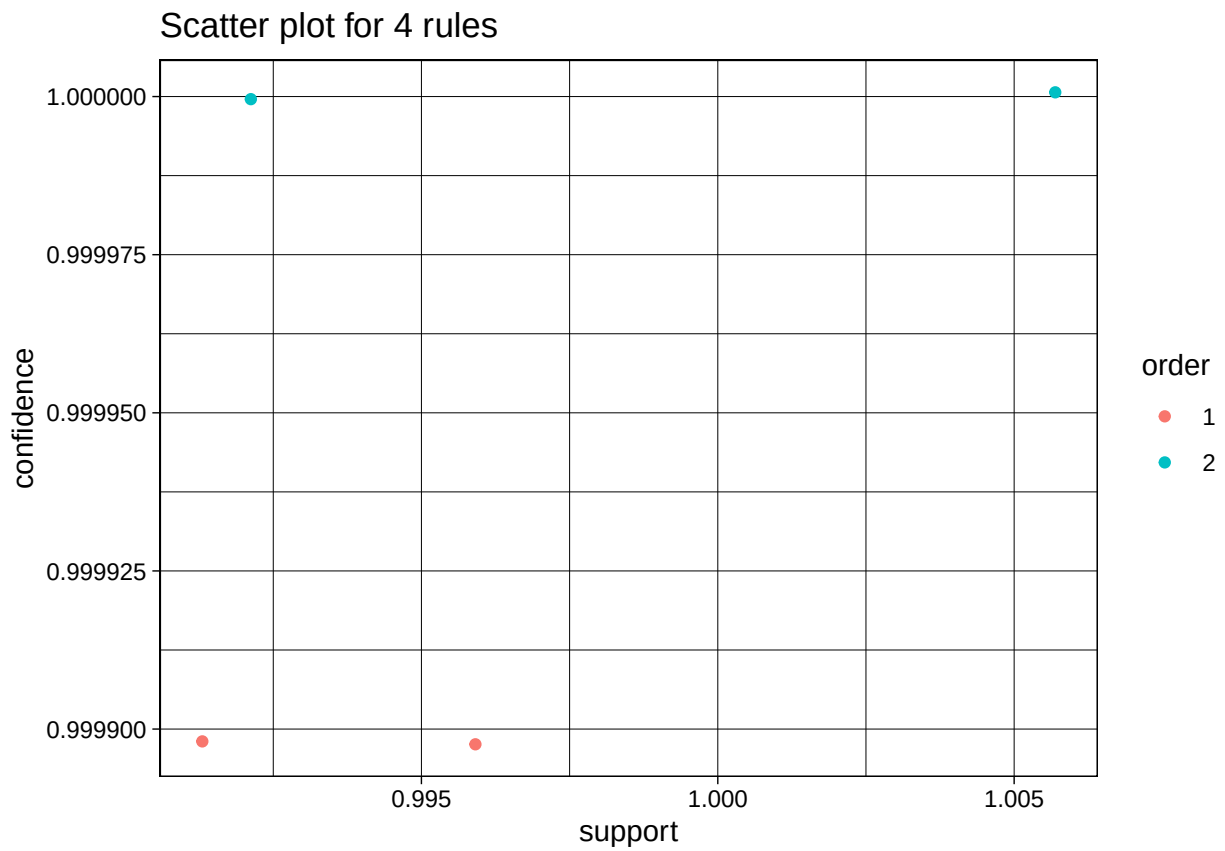
```
plot(musicrules, measure = c("support", "lift"), shading = "confidence")
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



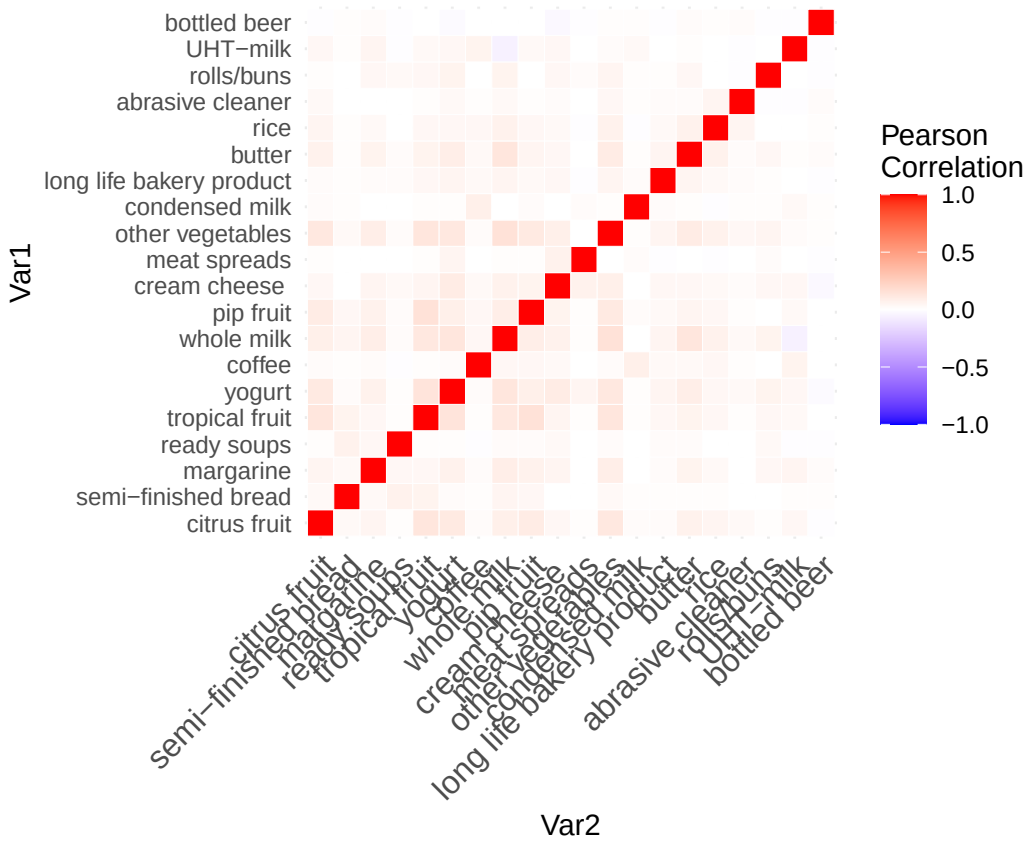

```
# "two key" plot: coloring is by size (order) of item set
plot(musicrules, method='two-key plot')
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```

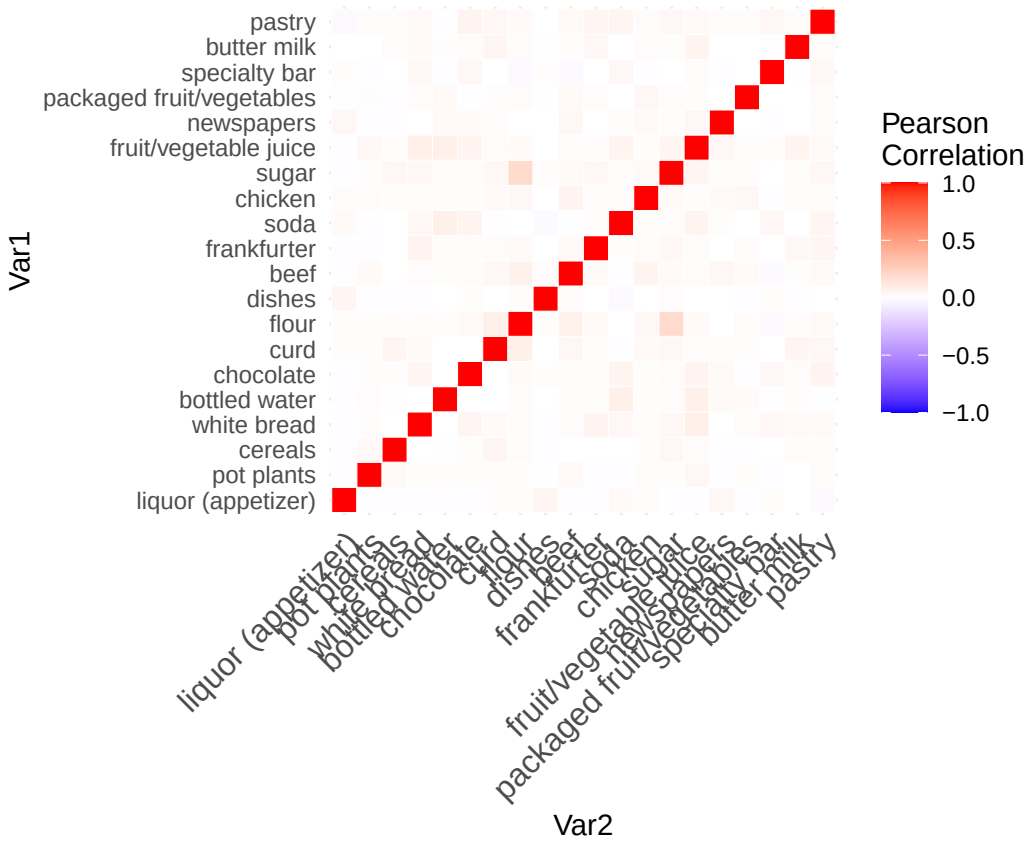



```
playlists_graph = associations2igraph(subset(musicrules, lift>1.005), associationsAsNodes = FALSE)
igraph::write_graph(playlists_graph, file='playlists.graphml', format = "graphml")
```

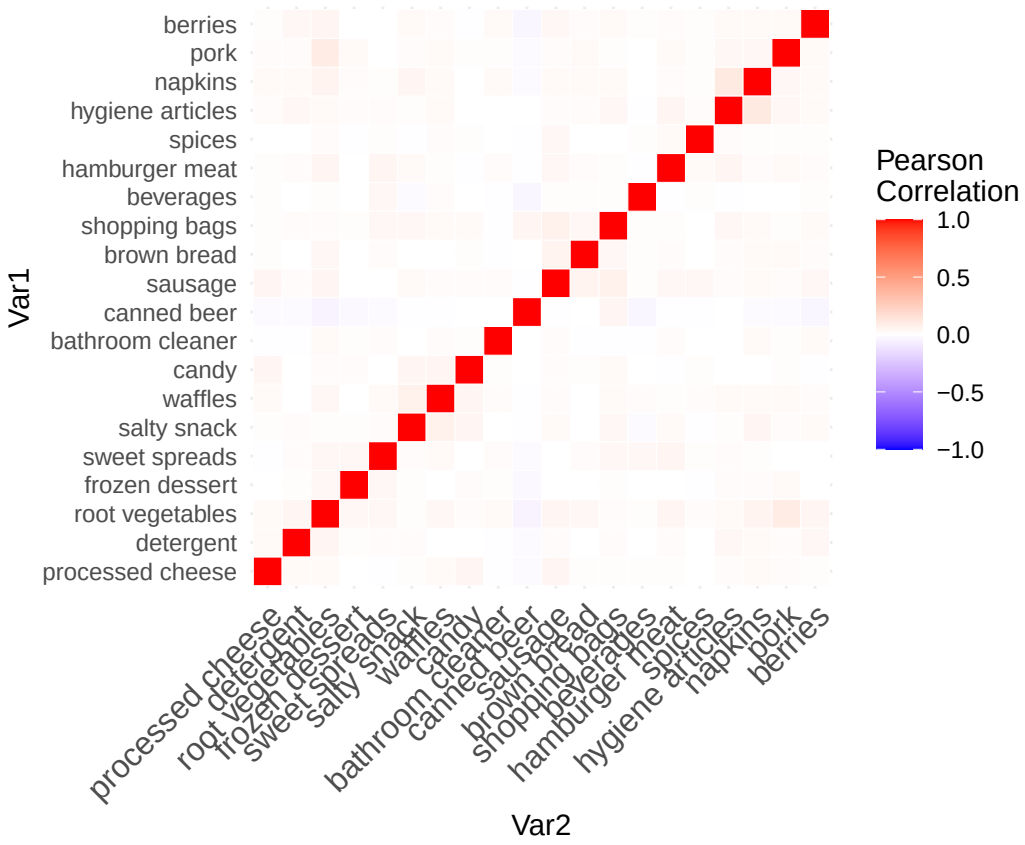
```
library(reshape2)
makemap = function(start,stop){
  cormat = melt(round(cor(df[start:stop]),2)) #correlation matrix
ggplot(data = cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()
}
makemap(1,20)
```



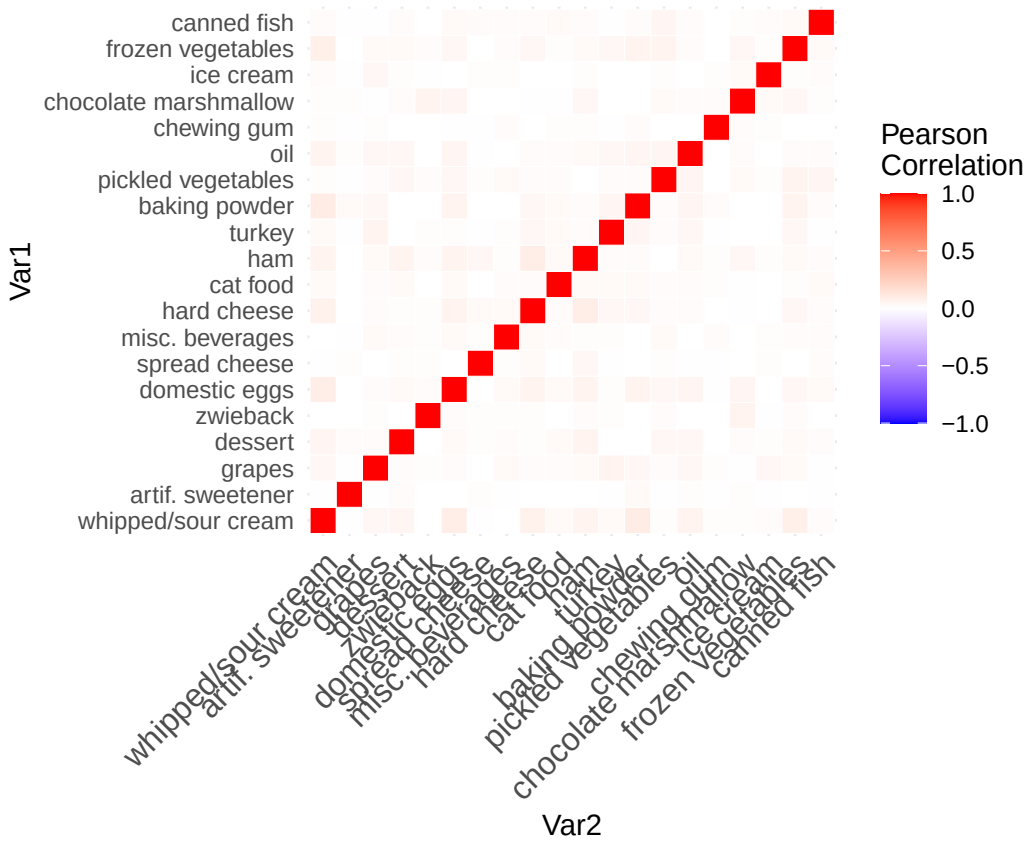

```
makemap(21,40)
```



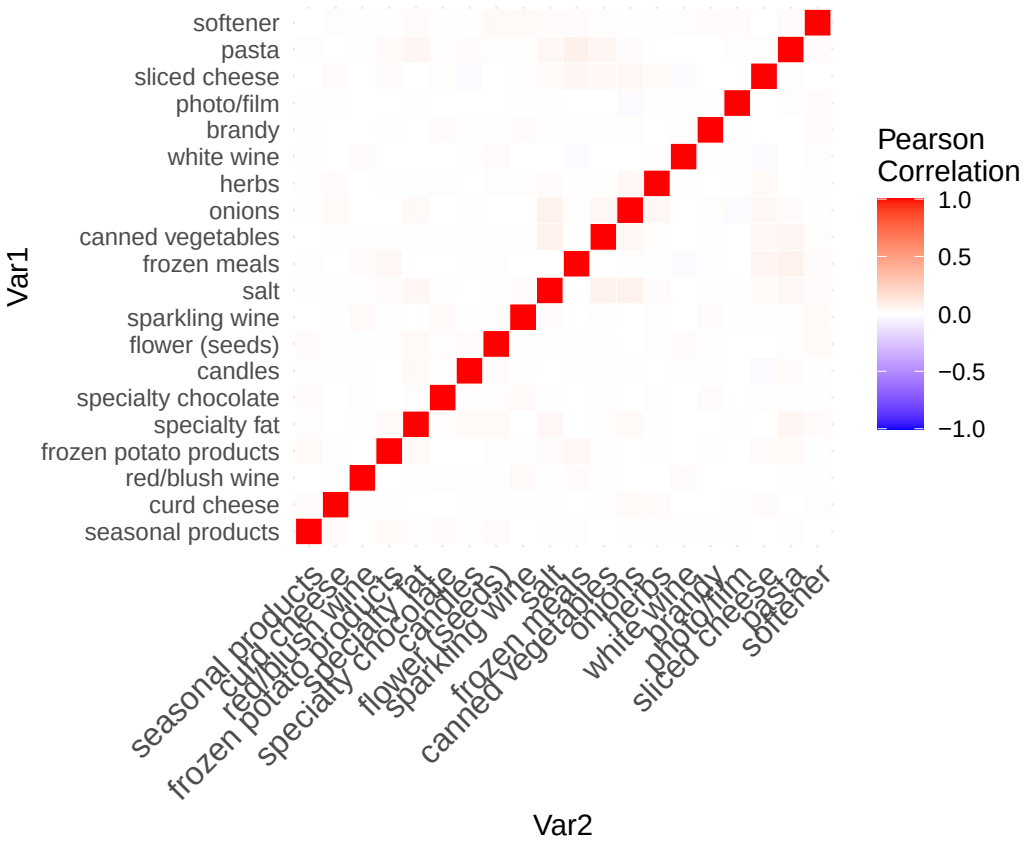

```
makemap(41,60)
```



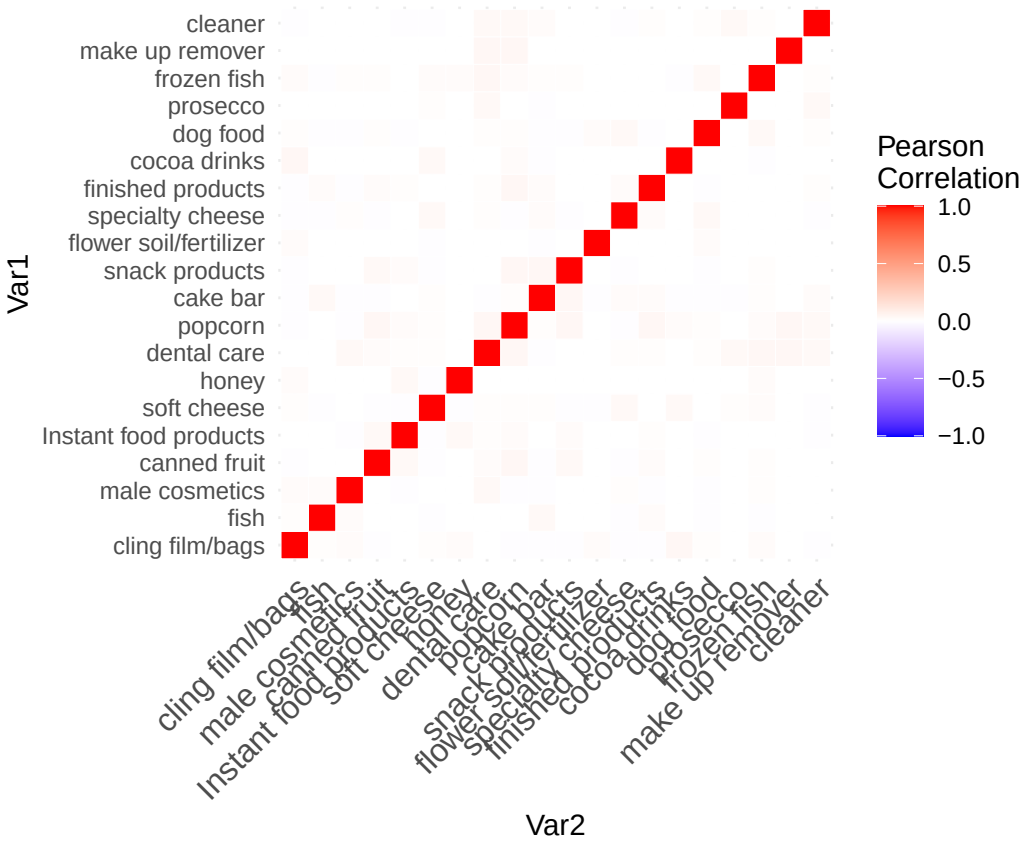

```
makemap(61,80)
```



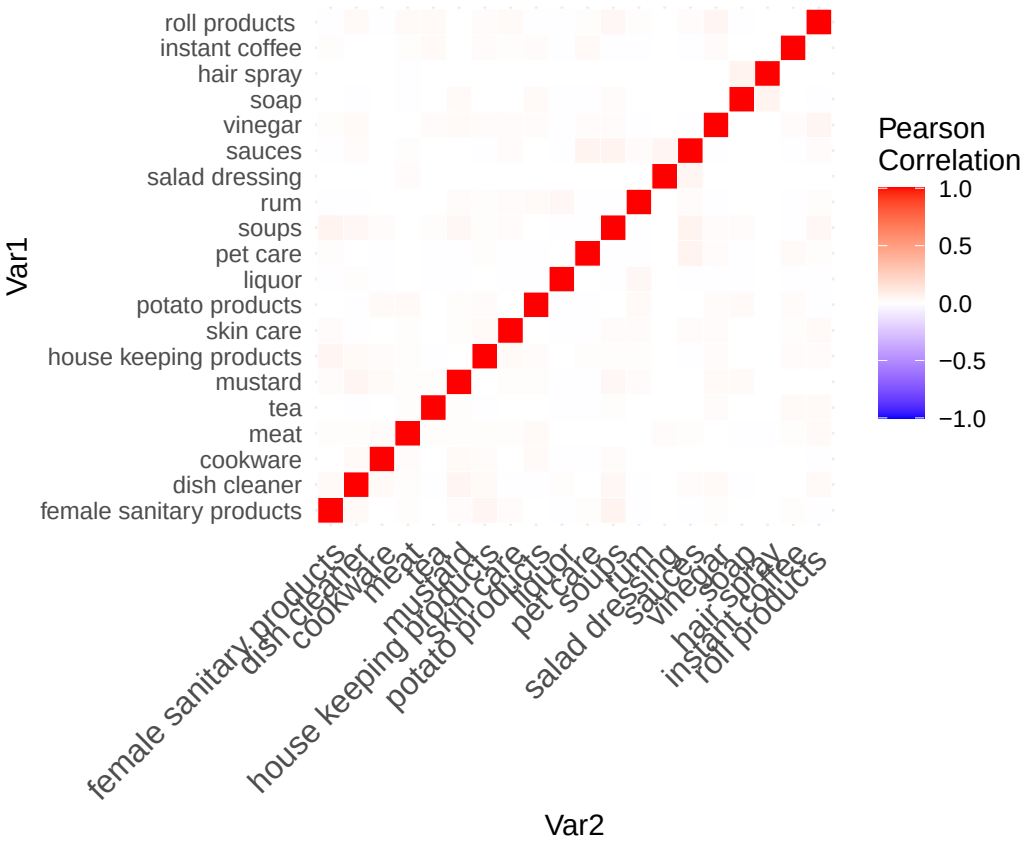

```
makemap(81,100)
```



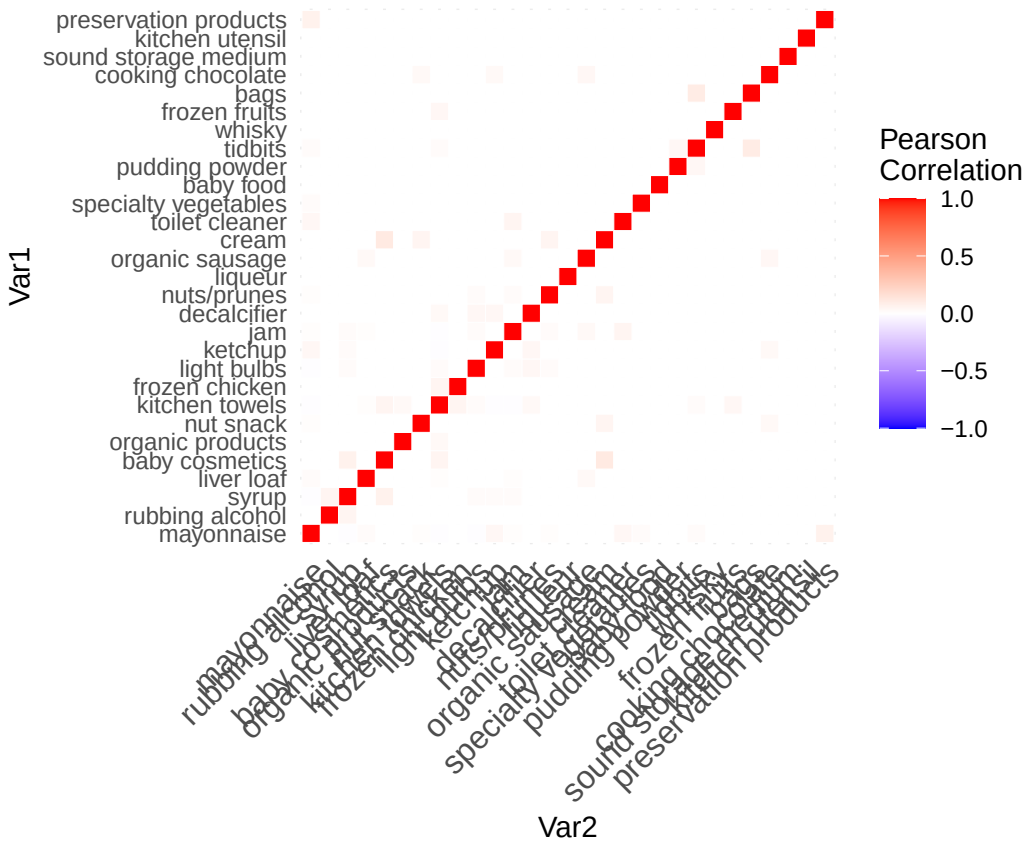

```
makemap(101,120)
```




```
makemap(121,140)
```




```
makemap(141,169)
```



I observed that the data was stored line by line with commas separating items. So, I one hot encoded them row by row into a dataframe which was shown as above to work properly. However, converting to the transactions class seemed to have some major issues, since both the class code and the code I found online reduced the 169 distinct items into only 4 columns.

Plotting heatmaps instead, it looks like the follow groups have associations:

- dairy, fruits, and vegetables
- sugar and flour
- napkins and hygiene articles
- pork and root vegetables
- sour cream, oil, baking powder, cheese, and eggs
- cream and baby cosmetics
- bags and tidbits