Candy Mini-Project*

Exploratory Analysis of Halloween Candy

Barry Grant

2021-10-26

Background



In this mini-project, you will explore FiveThirtyEight's Halloween Candy dataset. FiveThirtyEight, sometimes rendered as just **538**, is an American website that focuses mostly on opinion poll analysis, politics, economics, and sports blogging. They recently ran a rather large poll to

*http://thegrantlab.org/teaching/

determine which candy their readers like best. From their website: *"While we don't know who exactly voted, we do know this: 8,371 different IP addresses voted on about 269,000 randomly generated candy match-ups".*

So what is the top ranked snack-sized Halloween candy? What made some candies more desirable than others? Was it price? Maybe it was just sugar content? Were they chocolate? Did they contain peanuts or almonds? How about crisped rice or other biscuit-esque component, like a Kit Kat or malted milk ball? Was it fruit flavored? Was it made of hard candy, like a lollipop or a strawberry bon bon? Was there nougat? What even is nougat? I know I like nougat, but I still have no real clue what the damn thing is.

> Your task is to explore their candy dataset to find out answers to these types of questions - but most of all your job is to have fun, learn by doing hands on data analysis, and hopefully make this type of analysis less frightening for the future! **Let's get started.**

Learning Objectives

Building on your work with PCA in the previous labs (UK Foods, RNA-seq and beast cancer biopsy analysis), this mini-project gives you the opportunity to apply your data analysis skills to a new, tastier dataset. By the end of this lab, you should be able to:

- Import and explore an unfamiliar dataset to quickly characterize its structure and identify variables requiring special handling;
- Create and iteratively refine effective visualizations including ranked bar plots and scatter plots with `ggrepel()` labels, and interactive `plotly()` graphics to explore and communicate patterns in data;
- Generate and interpret correlation matrices to identify relationships between variables;
- Conduct principal component analysis (PCA) with appropriate scaling, interpret variance explained, and create score plots (PC1 vs PC2) to reveal clustering structure in high-dimensional data;
- Interpret PCA loadings to understand which original variables drive separation along principal components, connecting these back to the correlation structure;
- Synthesize multiple analyses (exploration, correlation, PCA) to draw substantive, data-driven conclusions.

Importing candy data

First things first, let's get the data from the FiveThirtyEight GitHub repo. You can either read from the URL directly or download this candy-data.csv file and place it in your project directory. Either way we need to load it up with `read.csv()` and inspect the data to see exactly what we're dealing with.

```
candy_file <- "candy-data.csv"

candy = ____(____, row.names=___)
head(candy)
```

```
             chocolate fruity caramel peanutyalmondy nougat crispedricewafer
100 Grand            1      0       1              0      0                 1
3 Musketeers         1      0       0              0      1                 0
One dime             0      0       0              0      0                 0
One quarter          0      0       0              0      0                 0
Air Heads            0      1       0              0      0                 0
Almond Joy           1      0       0              1      0                 0
             hard bar pluribus sugarpercent pricepercent winpercent
100 Grand       0   1        0        0.732        0.860   66.97173
3 Musketeers    0   1        0        0.604        0.511   67.60294
One dime        0   0        0        0.011        0.116   32.26109
One quarter     0   0        0        0.011        0.511   46.11650
Air Heads       0   0        0        0.906        0.511   52.34146
Almond Joy      0   1        0        0.465        0.767   50.34755
```

What is in the dataset?

The dataset includes all sorts of information about different kinds of candy. For example, is a candy chocolaty? Does it have nougat? How does its cost compare to other candies? How many people prefer one candy over another?

According to 538 the columns in the dataset include:

- **chocolate**: Does it contain chocolate?
- **fruity**: Is it fruit flavored?
- **caramel**: Is there caramel in the candy?
- **peanutyalmondy**: Does it contain peanuts, peanut butter or almonds?
- **nougat**: Does it contain nougat?
- **crispedricewafer**: Does it contain crisped rice, wafers, or a cookie component?
- **hard**: Is it a hard candy?
- **bar**: Is it a candy bar?
- **pluribus**: Is it one of many candies in a bag or box?
- **sugarpercent**: The percentile of sugar it falls under within the data set.
- **pricepercent**: The unit price percentile compared to the rest of the set.
- **winpercent**: The overall win percentage according to 269,000 match-ups (more on this in a moment).

We will take a whirlwind tour of this dataset and in the process answer the questions highlighted in red throughout this page that aim to guide your exploration process. We will then wrap up by trying Principal Component Analysis (PCA) on this dataset to get yet more experience with this important multivariate method. It will yield a kind of *"Map of Halloween Candy Space"*. How cool is that! Let's explore...

- **Q1**. How many different candy types are in this dataset?
- **Q2**. How many fruity candy types are in the dataset?

> Tip
>
> The functions `dim()`, `nrow()`, `table()` and `sum()` may be useful for answering the first 2 questions.

What is your favorite candy?

One of the most interesting variables in the dataset is `winpercent`. For a given candy this value is the percentage of people who prefer this candy over another randomly chosen candy from the dataset (what 538 term a match-up). Higher values indicate a more popular candy.

We can find the `winpercent` value for Twix by using its name to access the corresponding row of the dataset. This is because the dataset has each candy name as `rownames` (recall that we set this when we imported the original CSV file). For example the code for Twix is:

```
candy["Twix", ]$winpercent
```

```
[1] 81.64291
```

An alternate way to do this is with the help of the **dplyr** package that we have seen previously:

```
library(dplyr)

candy |>
  filter(row.names(candy)=="Twix") |>
  select(winpercent)
```

```
     winpercent
Twix   81.64291
```

- **Q3**. What is your favorite candy (other than Twix) in the dataset and what is it's `winpercent` value?
- **Q4**. What is the `winpercent` value for "Kit Kat"?
- **Q5**. What is the `winpercent` value for "Tootsie Roll Snack Bars"?

Side-note: the skimr::skim() function

There is a useful `skim()` function in the **skimr** package that can help give you a quick overview of a given dataset. Let's install this package and try it on our candy data.

```
library("skimr")
skim(candy)
```

Click to show output

Table 1: Data summary

Name	candy
Number of rows	85
Number of columns	12
_____	
Column type frequency:	
numeric	12
_____	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
chocolate	0	1	0.44	0.50	0.00	0.00	0.00	1.00	1.00	
fruity	0	1	0.45	0.50	0.00	0.00	0.00	1.00	1.00	
caramel	0	1	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
peanutyalmondy	0	1	0.16	0.37	0.00	0.00	0.00	0.00	1.00	
nougat	0	1	0.08	0.28	0.00	0.00	0.00	0.00	1.00	
crispedricewafer	0	1	0.08	0.28	0.00	0.00	0.00	0.00	1.00	
hard	0	1	0.18	0.38	0.00	0.00	0.00	0.00	1.00	
bar	0	1	0.25	0.43	0.00	0.00	0.00	0.00	1.00	
pluribus	0	1	0.52	0.50	0.00	0.00	1.00	1.00	1.00	
sugarpercent	0	1	0.48	0.28	0.01	0.22	0.47	0.73	0.99	
pricepercent	0	1	0.47	0.29	0.01	0.26	0.47	0.65	0.98	
winpercent	0	1	50.32	14.71	22.45	39.14	47.83	59.86	84.18	

From your use of the `skim()` function use the output to answer the following:

- **Q6**. Is there any variable/column that looks to be on a different scale to the majority of the other columns in the dataset?
- **Q7**. What do you think a zero and one represent for the `candy$chocolate` column?

> Tip
>
> Look at the "Variable type" print out from the `skim()` function. Most variables (i.e. columns) are on the zero to one scale but not all. Note that quite a few columns, such as `chocolate`, are exclusively either zero or one values.

Exploratory analysis

A good place to start any exploratory analysis is with a histogram. You can do this most easily with the base R function `hist()`. Alternatively, you can use `ggplot()` with `geom_hist()`. Either works well in this case and (as always) its your choice.

- **Q8**. Plot a histogram of `winpercent` values using both base R and **ggplot2**.
- **Q9**. Is the distribution of `winpercent` values symmetrical?
- **Q10**. Is the center of the distribution above or below 50%?
- **Q11**. On average is chocolate candy higher or lower ranked than fruit candy?
- **Q12**. Is this difference statistically significant?

> Tip
>
> **Hint**: The `chocolate`, `fruity`, `nougat` etc. columns indicate if a given candy has this feature (i.e. one if it has nougat, zero if it does not etc.). We can turn these into **logical** (a.k.a. TRUE/FALSE) values with the `as.logical()` function. We can then use this logical vector to access the corresponding candy rows (those with TRUE values). For example to get the `winpercent` values for all nougat containing candy we can use the code: `candy$winpercent[as.logical(candy$nougat)]`. In addition the functions `mean()` and `t.test()` should help you answer the last two questions here.

```
    Welch Two Sample t-test

data:  chocolate and fruity
t = 6.2582, df = 68.882, p-value = 2.871e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 11.44563 22.15795
```

```
sample estimates:
mean of x mean of y
 60.92153  44.11974
```

Overall Candy Rankings

Let's use the base R `order()` function together with `head()` to sort the whole dataset by a variable/column of interest. Or if you have been getting into the tidyverse and the **dplyr** package you can use the `arrange()` function together with `head()` to do the same thing and answer the following questions:

- **Q13**. What are the five least liked candy types in this set?
- **Q14**. What are the top 5 all time favorite candy types out of this set?

> Tip
>
> **Hint**: Using base R to sort by `sugarpercent` we could use `head(candy[order(candy$sugarpercent),], n=5)`, whilst using **dplyr** we have: `candy |> arrange(sugarpercent) |> head(5)`. Which approach do you prefer and why?

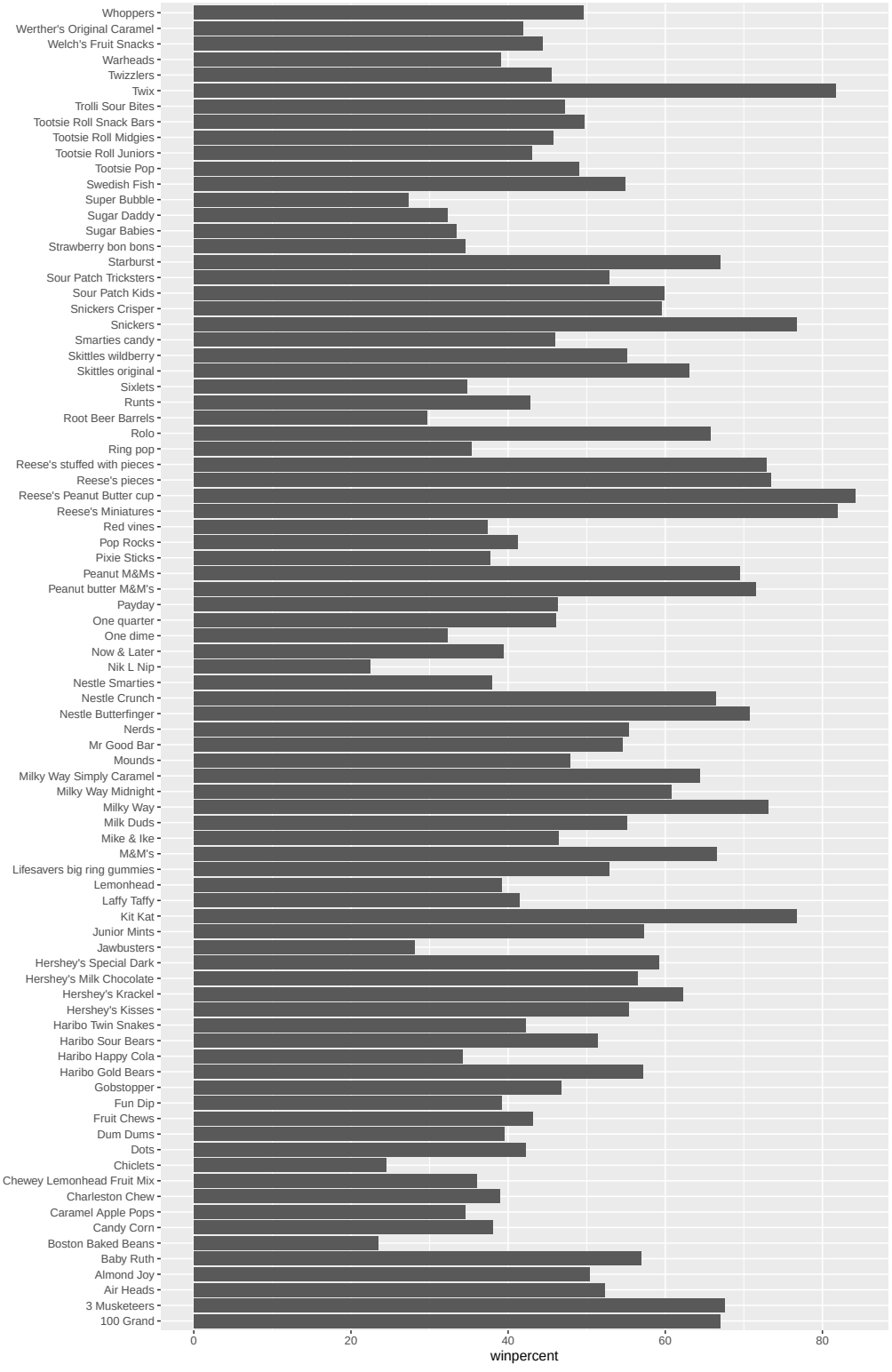
To examine more of the dataset in this vain we can make a barplot to visualize the overall rankings. As always we will use an iterative approach to building a useful visualization by getting a rough starting plot and then refining and adding useful details in a step-wise process.

- **Q15**. Make a first barplot of candy ranking based on `winpercent` values.

> Tip
>
> **HINT**: You can use the `aes(winpercent, rownames(candy))` for your first ggplot like so:

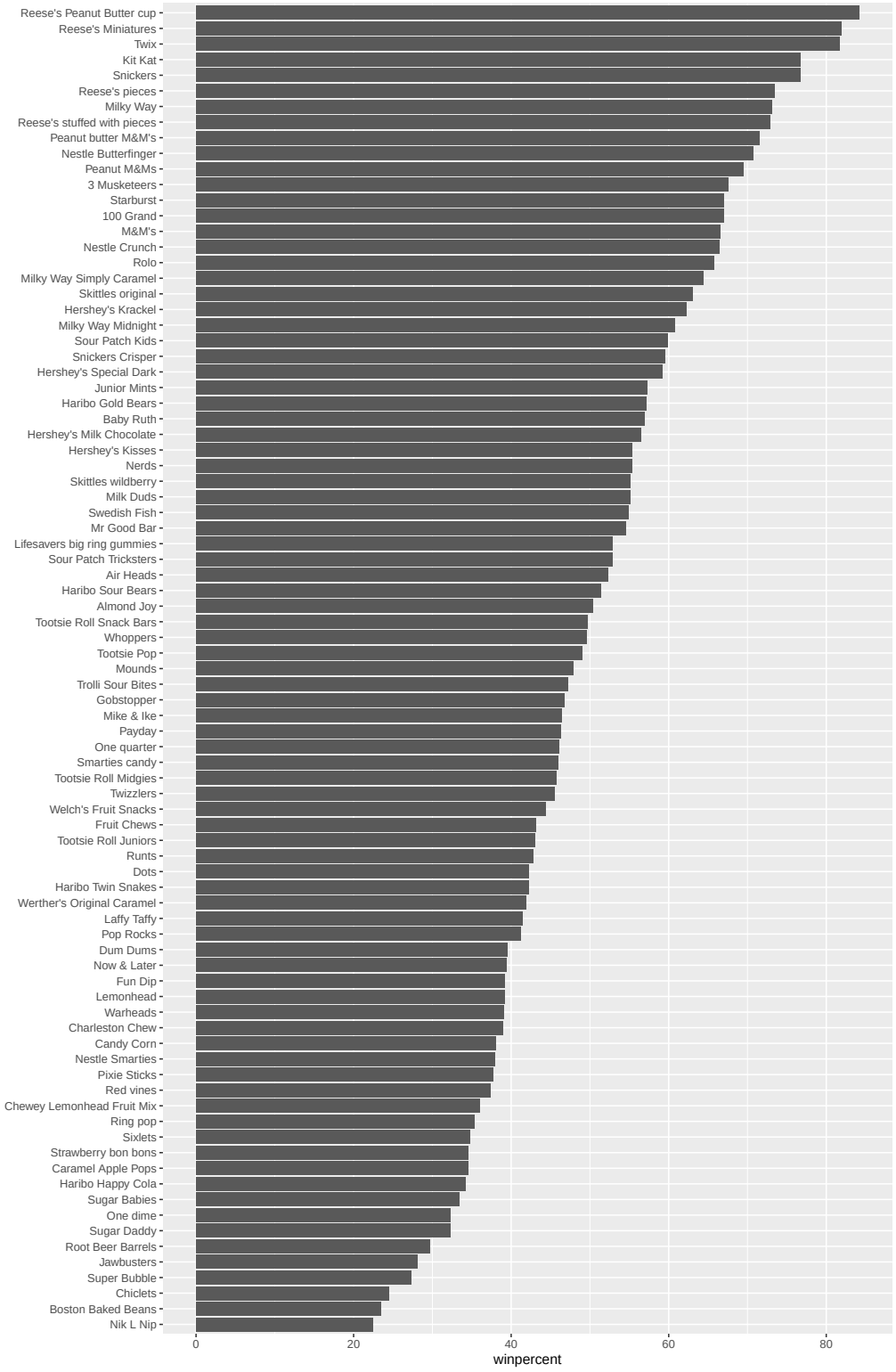
```
library(____)

ggplot(____) +
  ___(___, rownames(candy)) +
  geom_____()
```


Whoppers
Werther's Original Caramel
Welch's Fruit Snacks
Warheads
Twizzlers
Twix
Trolli Sour Bites
Tootsie Roll Snack Bars
Tootsie Roll Midgies
Tootsie Roll Juniors
Tootsie Pop
Swedish Fish
Super Bubble
Sugar Daddy
Sugar Babies
Strawberry bon bons
Starburst
Sour Patch Tricksters
Sour Patch Kids
Snickers Crisper
Snickers
Smarties candy
Skittles wildberry
Skittles original
Sixlets
Runts
Root Beer Barrels
Rolo
Ring pop
Reese's stuffed with pieces
Reese's pieces
Reese's Peanut Butter cup
Reese's Miniatures
Red vines
Pop Rocks
Pixie Sticks
Peanut M&Ms
Peanut butter M&M's
Payday
One quarter
One dime
Now & Later
Nik L Nip
Nestle Smarties
Nestle Crunch
Nestle Butterfinger
Nerds
Mr Good Bar
Mounds
Milky Way Simply Caramel
Milky Way Midnight
Milky Way
Milk Duds
Mike & Ike
M&M's
Lifesavers big ring gummies
Lemonhead
Laffy Taffy
Kit Kat
Junior Mints
Jawbusters
Hershey's Special Dark
Hershey's Milk Chocolate
Hershey's Krackel
Hershey's Kisses
Haribo Twin Snakes
Haribo Sour Bears
Haribo Happy Cola
Haribo Gold Bears
Gobstopper
Fun Dip
Fruit Chews
Dum Dums
Dots
Chiclets
Chewey Lemonhead Fruit Mix
Charleston Chew
Caramel Apple Pops
Candy Corn
Boston Baked Beans
Baby Ruth
Almond Joy
Air Heads
3 Musketeers
100 Grand
0
20
40
60
80
winpercent

- **Q16**. This is quite ugly, use the `reorder()` function to get the bars sorted by `winpercent`?

> Tip
>
> **HINT**: You can use `aes(winpercent, reorder(rownames(candy),winpercent))` to improve your plot.


Reese's Peanut Butter cup
Reese's Miniatures
Twix
Kit Kat
Snickers
Reese's pieces
Milky Way
Reese's stuffed with pieces
Peanut butter M&M's
Nestle Butterfinger
Peanut M&Ms
3 Musketeers
Starburst
100 Grand
M&M's
Nestle Crunch
Rolo
Milky Way Simply Caramel
Skittles original
Hershey's Krackel
Milky Way Midnight
Sour Patch Kids
Snickers Crisper
Hershey's Special Dark
Junior Mints
Haribo Gold Bears
Baby Ruth
Hershey's Milk Chocolate
Hershey's Kisses
Nerds
Skittles wildberry
Milk Duds
Swedish Fish
Mr Good Bar
Lifesavers big ring gummies
Sour Patch Tricksters
Air Heads
Haribo Sour Bears
Almond Joy
Tootsie Roll Snack Bars
Whoppers
Tootsie Pop
Mounds
Trolli Sour Bites
Gobstopper
Mike & Ike
Payday
One quarter
Smarties candy
Tootsie Roll Midgies
Twizzlers
Welch's Fruit Snacks
Fruit Chews
Tootsie Roll Juniors
Runts
Dots
Haribo Twin Snakes
Werther's Original Caramel
Laffy Taffy
Pop Rocks
Dum Dums
Now & Later
Fun Dip
Lemonhead
Warheads
Charleston Chew
Candy Corn
Nestle Smarties
Pixie Sticks
Red vines
Chewey Lemonhead Fruit Mix
Ring pop
Sixlets
Strawberry bon bons
Caramel Apple Pops
Haribo Happy Cola
Sugar Babies
One dime
Sugar Daddy
Root Beer Barrels
Jawbusters
Super Bubble
Chiclets
Boston Baked Beans
Nik L Nip
0
20
40
60
80
winpercent

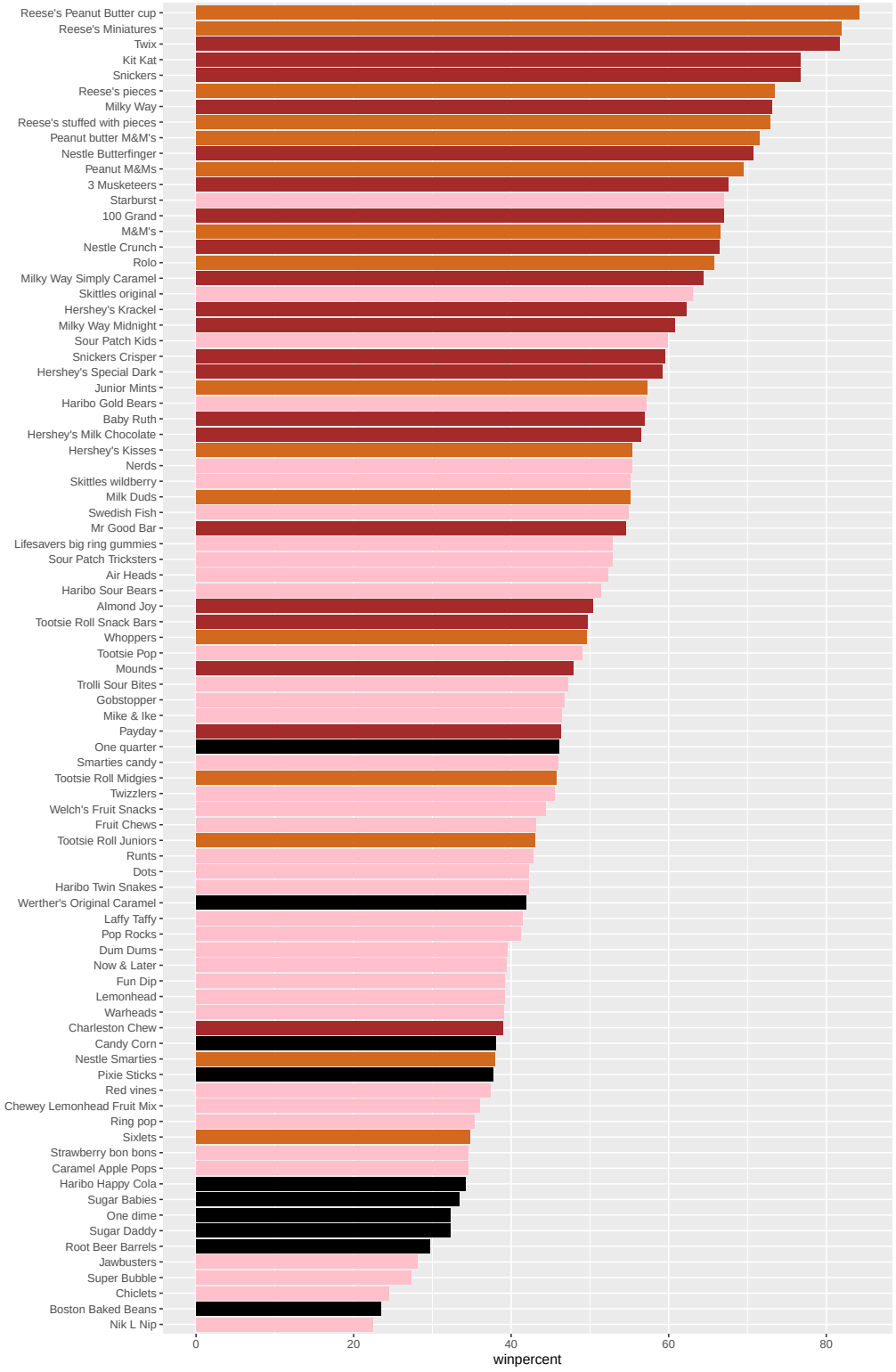
Time to add some useful color

Let's setup a color vector (that signifies candy type) that we can then use for some future plots. We start by making a vector of all black values (one for each candy). Then we overwrite chocolate (for chocolate candy), brown (for candy bars) and red (for fruity candy) values.

```
my_cols=rep("black", ___)
my_cols[as.logical(___)] = "chocolate"
my_cols[___] = "brown"
my_cols[___] = ___
```

Now let's try our barplot with these colors. Note that we use `fill=my_cols` for `geom_col()`. Experiment to see what happens if you use `col=mycols`.

```
ggplot(candy) +
  aes(winpercent, reorder(rownames(candy),winpercent)) +
  geom_col(fill=my_cols) +
  ylab("")
```


Reese's Peanut Butter cup
Reese's Miniatures
Twix
Kit Kat
Snickers
Reese's pieces
Milky Way
Reese's stuffed with pieces
Peanut butter M&M's
Nestle Butterfinger
Peanut M&Ms
3 Musketeers
Starburst
100 Grand
M&M's
Nestle Crunch
Rolo
Milky Way Simply Caramel
Skittles original
Hershey's Krackel
Milky Way Midnight
Sour Patch Kids
Snickers Crisper
Hershey's Special Dark
Junior Mints
Haribo Gold Bears
Baby Ruth
Hershey's Milk Chocolate
Hershey's Kisses
Nerds
Skittles wildberry
Milk Duds
Swedish Fish
Mr Good Bar
Lifesavers big ring gummies
Sour Patch Tricksters
Air Heads
Haribo Sour Bears
Almond Joy
Tootsie Roll Snack Bars
Whoppers
Tootsie Pop
Mounds
Trolli Sour Bites
Gobstopper
Mike & Ike
Payday
One quarter
Smarties candy
Tootsie Roll Midgies
Twizzlers
Welch's Fruit Snacks
Fruit Chews
Tootsie Roll Juniors
Runts
Dots
Haribo Twin Snakes
Werther's Original Caramel
Laffy Taffy
Pop Rocks
Dum Dums
Now & Later
Fun Dip
Lemonhead
Warheads
Charleston Chew
Candy Corn
Nestle Smarties
Pixie Sticks
Red vines
Chewey Lemonhead Fruit Mix
Ring pop
Sixlets
Strawberry bon bons
Caramel Apple Pops
Haribo Happy Cola
Sugar Babies
One dime
Sugar Daddy
Root Beer Barrels
Jawbusters
Super Bubble
Chiclets
Boston Baked Beans
Nik L Nip
0
20
40
60
80
winpercent

Now, for the first time, using this plot we can answer questions like:

- **Q17**. What is the worst ranked chocolate candy?
- **Q18**. What is the best ranked fruity candy?

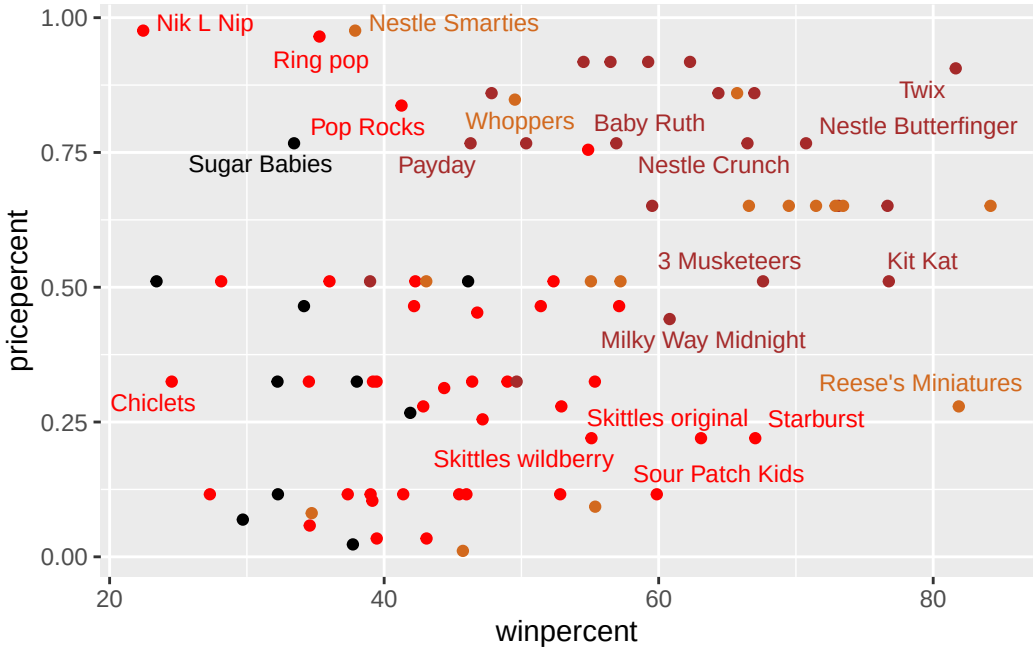
Taking a look at pricepercent

What about value for money? What is the best candy for the least money? One way to get at this would be to make a plot of `winpercent` vs the `pricepercent` variable. The `pricepercent` variable records the percentile rank of the candy's price against all the other candies in the dataset. Lower values are less expensive and higher values are more expensive.

To this plot we will add text labels so we can more easily identify a given candy. There is a regular `geom_label()` that comes with ggplot2. However, as there are quite a few candies in our dataset lots of these labels will be overlapping and hard to read. To help with this we can use the `geom_text_repel()` function from the **ggrepel** package.

```
library(ggrepel)

# How about a plot of win vs price
ggplot(___) +
  aes(___, ___, label=rownames(candy)) +
  geom_point(col=my_cols) +
  geom_text_repel(col=my_cols, size=3.3, max.overlaps = 5)
```



- **Q19**. Which candy type is the highest ranked in terms of `winpercent` for the least money - i.e. offers the most bang for your buck?
- **Q20**. What are the top 5 most expensive candy types in the dataset and of these which is the least popular?

Tip

Hint: To see which candy is the most expensive (and which is the least expensive) we can `order()` the dataset by `pricepercent`.

```
ord <- order(candy$pricepercent, decreasing = TRUE)
head( candy[ord,c(11,12)], n=5 )
```

```
                           pricepercent winpercent
Nik L Nip                         0.976   22.44534
Nestle Smarties                   0.976   37.88719
Ring pop                          0.965   35.29076
Hershey's Krackel                 0.918   62.28448
Hershey's Milk Chocolate          0.918   56.49050
```

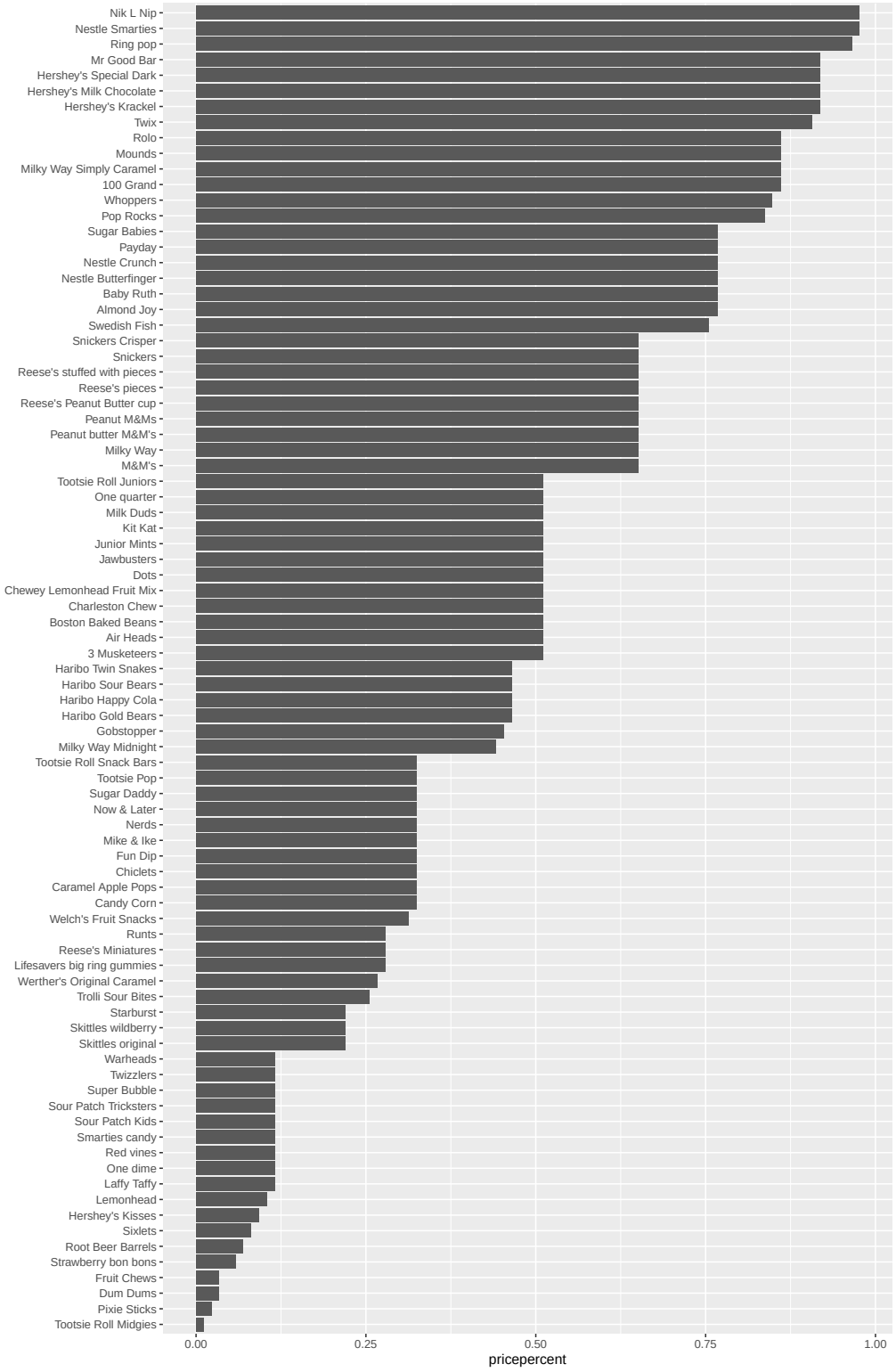
Using the **dplyr** package we can do the same thing

```
library(dplyr)
candy |>
  arrange(-pricepercent) |>
  select(pricepercent, winpercent) |>
  head(n=5)
```

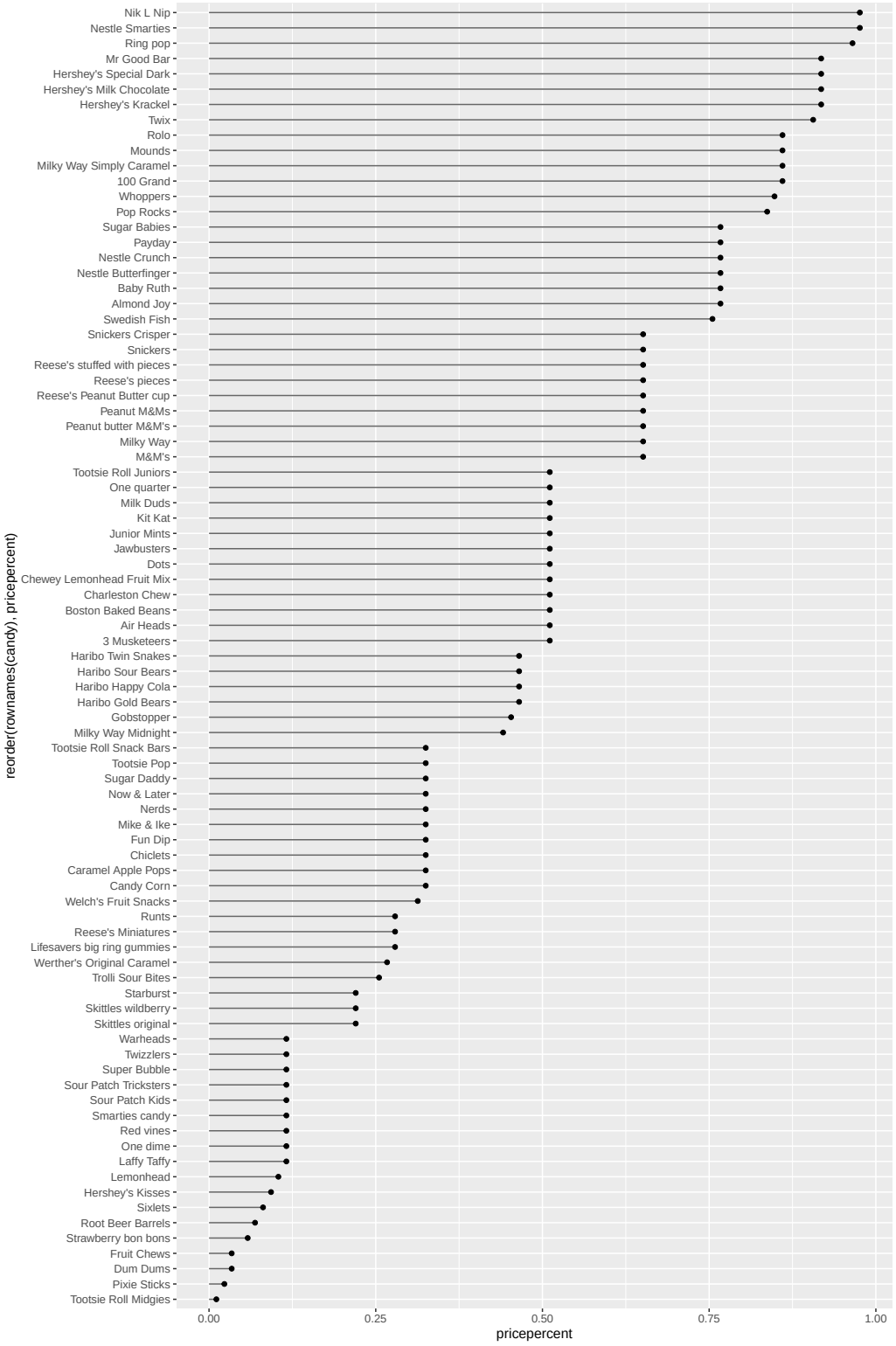
```
                          pricepercent winpercent
Nik L Nip                        0.976   22.44534
Nestle Smarties                  0.976   37.88719
Ring pop                         0.965   35.29076
Hershey's Krackel                0.918   62.28448
Hershey's Milk Chocolate         0.918   56.49050
```

Optional

- **Q21**. Make a barplot again with `geom_col()` this time using `pricepercent` and then improve this step by step, first ordering the x-axis by value and finally making a so called "dot chat" or "lollipop" chart by swapping `geom_col()` for `geom_point()` + `geom_segment()`.


Nik L Nip
Nestle Smarties
Ring pop
Mr Good Bar
Hershey's Special Dark
Hershey's Milk Chocolate
Hershey's Krackel
Twix
Rolo
Mounds
Milky Way Simply Caramel
100 Grand
Whoppers
Pop Rocks
Sugar Babies
Payday
Nestle Crunch
Nestle Butterfinger
Baby Ruth
Almond Joy
Swedish Fish
Snickers Crisper
Snickers
Reese's stuffed with pieces
Reese's pieces
Reese's Peanut Butter cup
Peanut M&Ms
Peanut butter M&M's
Milky Way
M&M's
Tootsie Roll Juniors
One quarter
Milk Duds
Kit Kat
Junior Mints
Jawbusters
Dots
Chewey Lemonhead Fruit Mix
Charleston Chew
Boston Baked Beans
Air Heads
3 Musketeers
Haribo Twin Snakes
Haribo Sour Bears
Haribo Happy Cola
Haribo Gold Bears
Gobstopper
Milky Way Midnight
Tootsie Roll Snack Bars
Tootsie Pop
Sugar Daddy
Now & Later
Nerds
Mike & Ike
Fun Dip
Chiclets
Caramel Apple Pops
Candy Corn
Welch's Fruit Snacks
Runts
Reese's Miniatures
Lifesavers big ring gummies
Werther's Original Caramel
Trolli Sour Bites
Starburst
Skittles wildberry
Skittles original
Warheads
Twizzlers
Super Bubble
Sour Patch Tricksters
Sour Patch Kids
Smarties candy
Red vines
One dime
Laffy Taffy
Lemonhead
Hershey's Kisses
Sixlets
Root Beer Barrels
Strawberry bon bons
Fruit Chews
Dum Dums
Pixie Sticks
Tootsie Roll Midgies
0.00
0.25
0.50
0.75
1.00
pricepercent

```
# Make a lollipop chart of pricepercent
ggplot(candy) +
  aes(pricepercent, reorder(rownames(candy), pricepercent)) +
  geom_segment(aes(yend = reorder(rownames(candy), pricepercent),
                   xend = 0), col="gray40") +
    geom_point()
```


Nik L Nip
Nestle Smarties
Ring pop
Mr Good Bar
Hershey's Special Dark
Hershey's Milk Chocolate
Hershey's Krackel
Twix
Rolo
Mounds
Milky Way Simply Caramel
100 Grand
Whoppers
Pop Rocks
Sugar Babies
Payday
Nestle Crunch
Nestle Butterfinger
Baby Ruth
Almond Joy
Swedish Fish
Snickers Crisper
Snickers
Reese's stuffed with pieces
Reese's pieces
Reese's Peanut Butter cup
Peanut M&Ms
Peanut butter M&M's
Milky Way
M&M's
Tootsie Roll Juniors
One quarter
Milk Duds
Kit Kat
Junior Mints
Jawbusters
Dots
Chewey Lemonhead Fruit Mix
Charleston Chew
Boston Baked Beans
Air Heads
3 Musketeers
Haribo Twin Snakes
Haribo Sour Bears
Haribo Happy Cola
Haribo Gold Bears
Gobstopper
Milky Way Midnight
Tootsie Roll Snack Bars
Tootsie Pop
Sugar Daddy
Now & Later
Nerds
Mike & Ike
Fun Dip
Chiclets
Caramel Apple Pops
Candy Corn
Welch's Fruit Snacks
Runts
Reese's Miniatures
Lifesavers big ring gummies
Werther's Original Caramel
Trolli Sour Bites
Starburst
Skittles wildberry
Skittles original
Warheads
Twizzlers
Super Bubble
Sour Patch Tricksters
Sour Patch Kids
Smarties candy
Red vines
One dime
Laffy Taffy
Lemonhead
Hershey's Kisses
Sixlets
Root Beer Barrels
Strawberry bon bons
Fruit Chews
Dum Dums
Pixie Sticks
Tootsie Roll Midgies
reorder(rownames(candy), pricepercent)
0.00
0.25
0.50
0.75
1.00
pricepercent

One of the most interesting aspects of this chart is that a lot of the candies share the same ranking, so it looks like quite a few of them are the same price.

Exploring the correlation structure

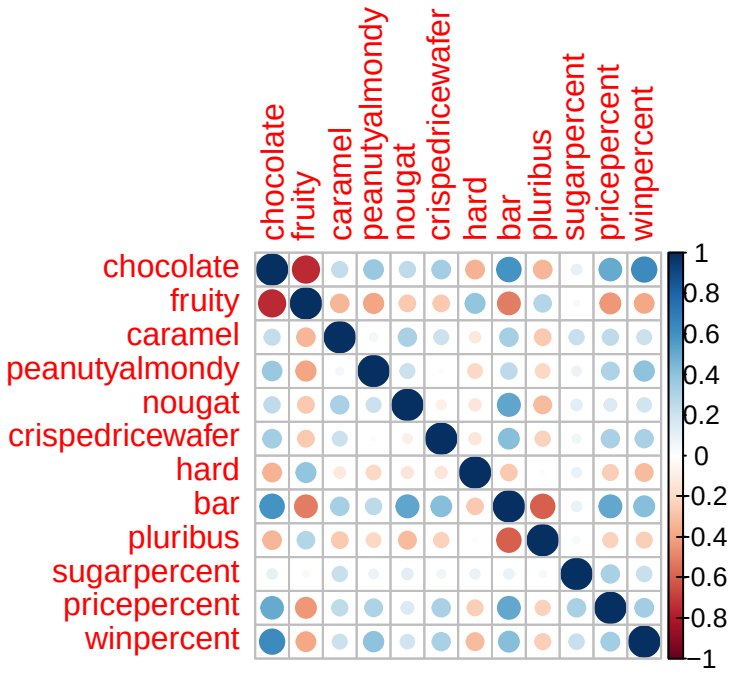
Before we dive into PCA in the next section, it's worth pausing to look more closely at the correlations between all pairs of variables.

We have already seen that being a chocolate candy tends to imply being relatively expensive and popular - in other words, these variables are *positively correlated* with one another. Here we will examine the magnitude and direction of all variable pair combinations to see how they relate to each other more broadly.

> **Side-note**: This matters especially because PCA operates not on raw data but on the *relationships between variables*. Specifically, it **finds the directions that capture the most co-variation in your data**.
>
> This is also why we will use `scale=TRUE` when calling `prcomp()` in the next section: it ensures all variables contribute equally by working from correlations (standardized and scale-free) rather than raw covariances, which would be dominated by whichever variable happens to have the largest numeric range. Can you guess which variable in our dataset that would be?
>
> An important point here is to think of the analysis below as a preview of your upcoming PCA results: the patterns we spot here should reappear in the loadings plot at the end of the next section.

The code below calculates the pairwise correlation between all variables and views the results with the **corrplot** package to plot a correlation matrix.

```
library(corrplot)
cij <- cor(candy)
corrplot(cij)
```



A few things to notice about this correlation matrix plot. First, the diagonal is always perfectly blue (correlation = 1.0) — every variable is perfectly correlated with itself. Second, the matrix is symmetric: the correlation between `chocolate` and `bar` is the same as between `bar` and `chocolate`, so the upper and lower triangles are mirror images. Most importantly, this matrix is exactly what PCA will decompose mathematically. When we run `prcomp()` on scaled data, it is performing an eigen-decomposition of this correlation matrix — finding the directions (eigen-vectors) that explain the most variance (eigen-values). The loading values you will see shortly are those eigen-vectors, and the proportion of variance explained by each PC corresponds to its eigen-value. The corrplot is therefore not just an exploratory tool, it is the structure that PCA will summarize.

- **Q22**. Examining this plot what two variables are anti-correlated (i.e. have minus values)?
- **Q23**. Use your corrplot result to make a prediction: which variables do you expect will have the largest contributions (i.e. loadings) to PC1 (i.e., drive the most separation between candies along the first principal component)?

  HINT: Do you like chocolaty fruity candies?

  Side-note: A quick note on terminology, **Covariance** measures whether two variables tend to increase and decrease together, but its magnitude depends on the scale of each variable. A variable measured in percentages will naturally produce larger covariances than one measured as a binary 0/1. **Correlation** is simply a standardized version of covariance, rescaled to always fall between -1 and +1, making comparisons across variable pairs meaningful.

Principal Component Analysis

Let's apply PCA using the `prcomp()` function to our candy dataset remembering to set the `scale=TRUE` argument.

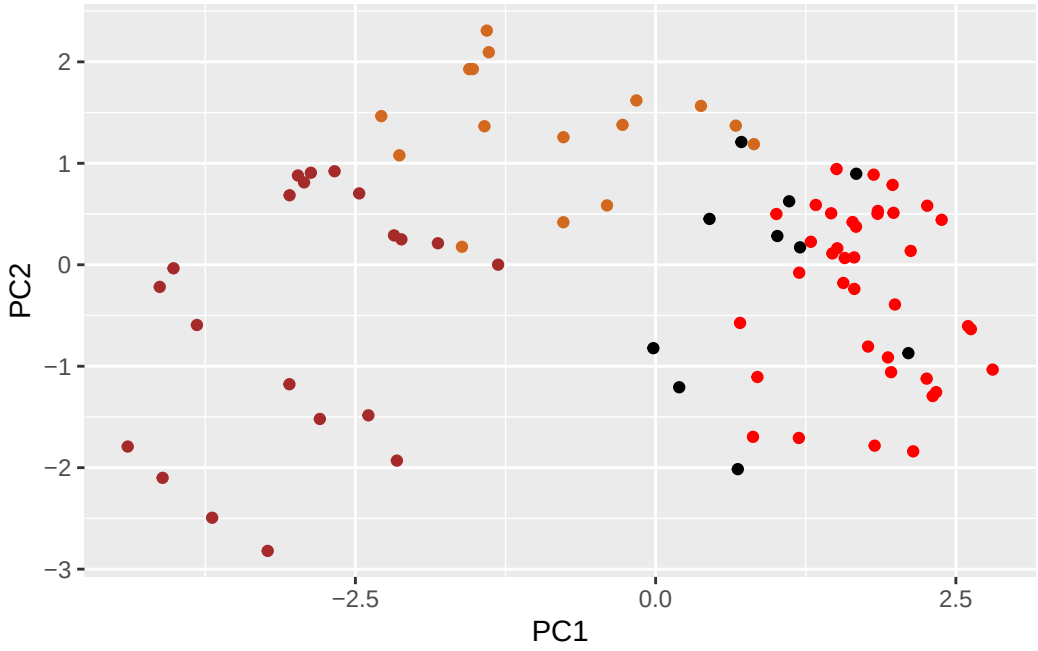
> **Side-note**: Feel free to examine what happens if you leave this argument out (i.e. use the default `scale=FALSE`). Then examine the `summary(pca)` and `pca$rotation[,1]` component and see that it is dominated by `winpercent` (which is after all measured on a very different scale than the other variables).

```
pca <- ____(candy, ____)
summary(___)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4    PC5     PC6     PC7
Standard deviation     2.0788 1.1378 1.1092 1.07533 0.9518 0.81923 0.81530
Proportion of Variance 0.3601 0.1079 0.1025 0.09636 0.0755 0.05593 0.05539
Cumulative Proportion  0.3601 0.4680 0.5705 0.66688 0.7424 0.79830 0.85369
                           PC8     PC9    PC10    PC11    PC12
Standard deviation     0.74530 0.67824 0.62349 0.43974 0.39760
Proportion of Variance 0.04629 0.03833 0.03239 0.01611 0.01317
Cumulative Proportion  0.89998 0.93832 0.97071 0.98683 1.00000
```

Now we can plot our main PCA score plot of PC1 vs PC2 using our `my_cols` colors for each candy:

```
ggplot(pca$x) +
  ___ +
  ___
```

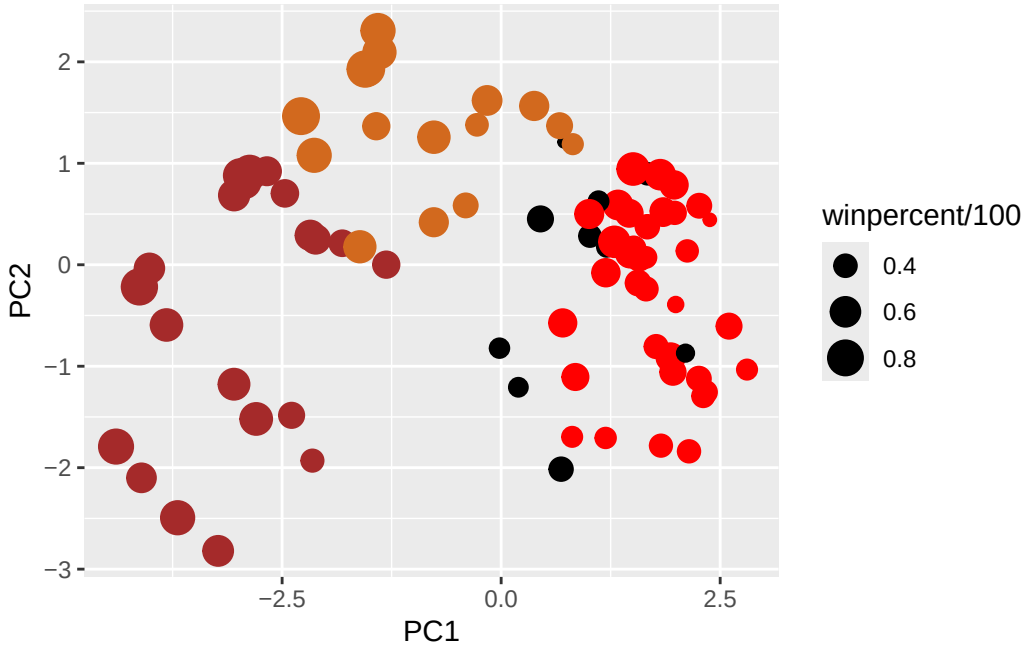


We can make a much nicer plot by adding extra annotation data (like candy names and winpercent values for example) but it is important to note that ggplot works best when you supply an input data.frame that includes a separate column for each of the aesthetics you would like displayed in your final plot. To accomplish this we can make a new data.frame here that contains our PCA results with all the rest of our candy data. We will then use this for making our plots below

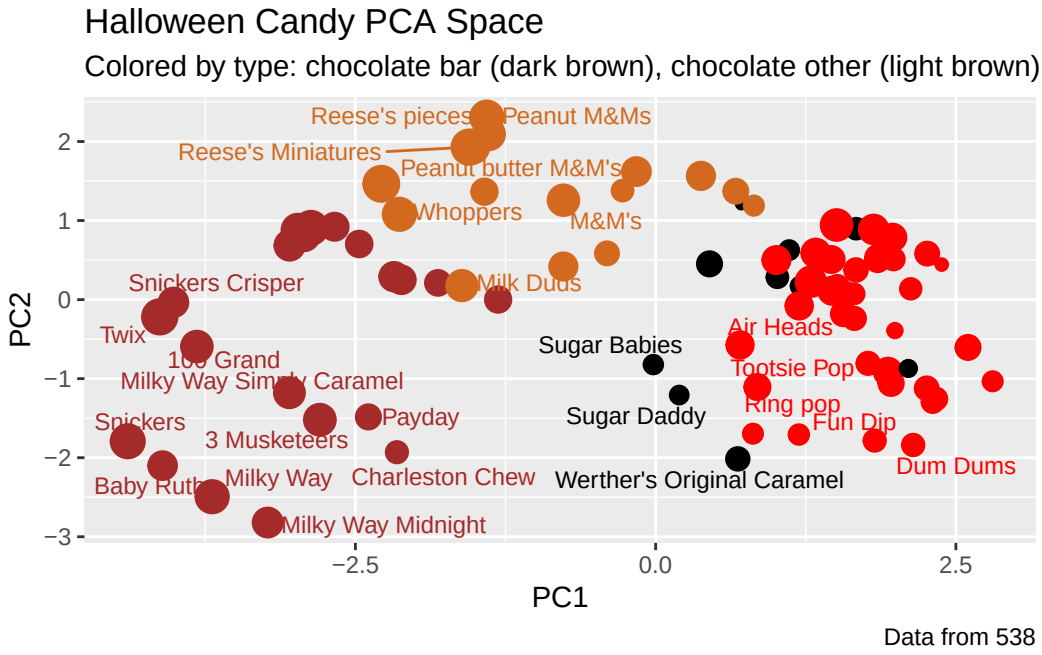
```
# Make a new data-frame with our PCA results and candy data
my_data <- cbind(candy, pca$x)
```

```
p <- ggplot(my_data) +
        aes(x=___, y=___,
            size=winpercent/100,
            text=rownames(my_data),
            label=rownames(my_data)) +
        geom_point(col=___)

p
```



Again we can use the **ggrepel** package and the function `ggrepel::geom_text_repel()` to label up the plot with non overlapping candy names:



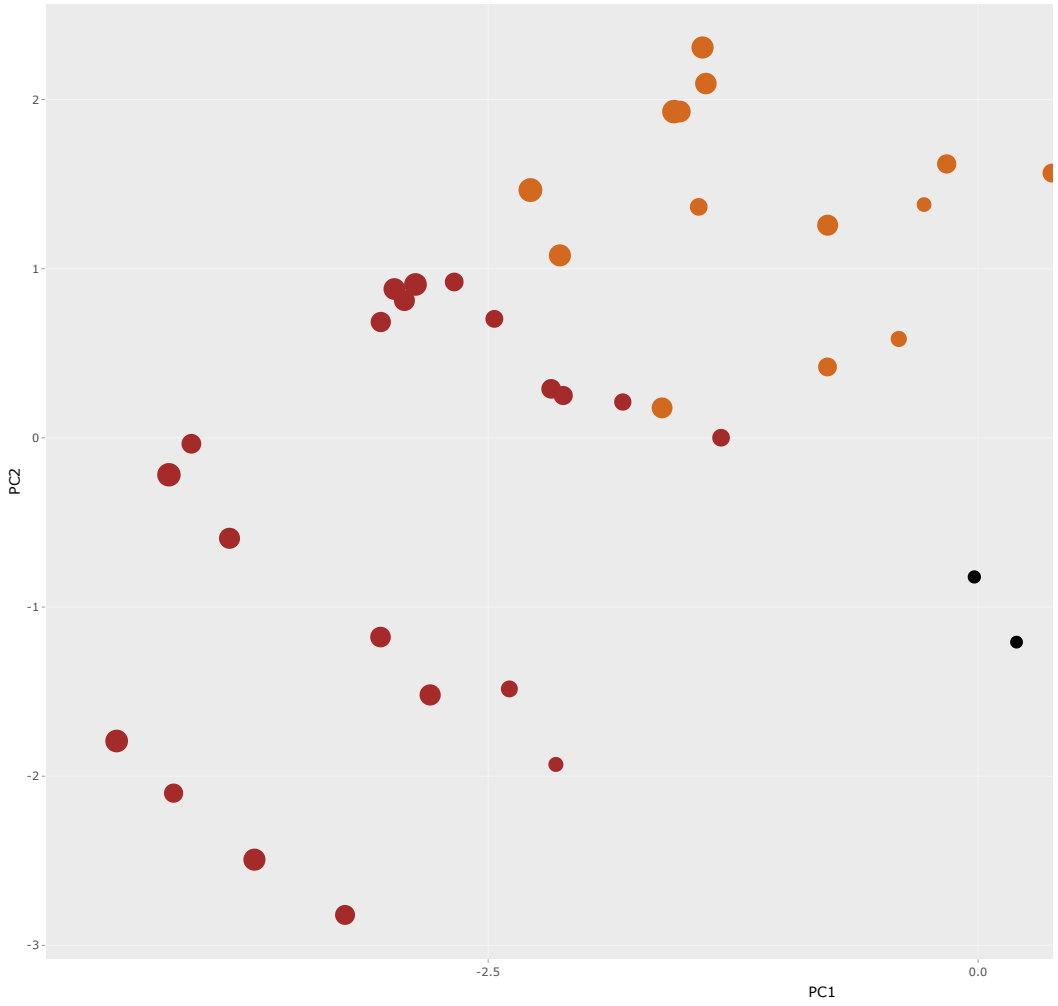
- **Q24**. Complete the code to generate the "Halloween Candy PCA Space Map" above. Be

sure to have legable labels along with a plot title and subtitle:

```
p + geom_text_repel(size=3.3, col=___, max.overlaps = 7)  +
  theme(legend.position = "none") +
  labs(___,
       ___,
       ___)
```

If you want to see more candy labels you can change the `max.overlaps` value to allow more overlapping labels or pass the ggplot object `p` to **plotly** like so to generate an interactive plot that you can mouse over to see all the labels:

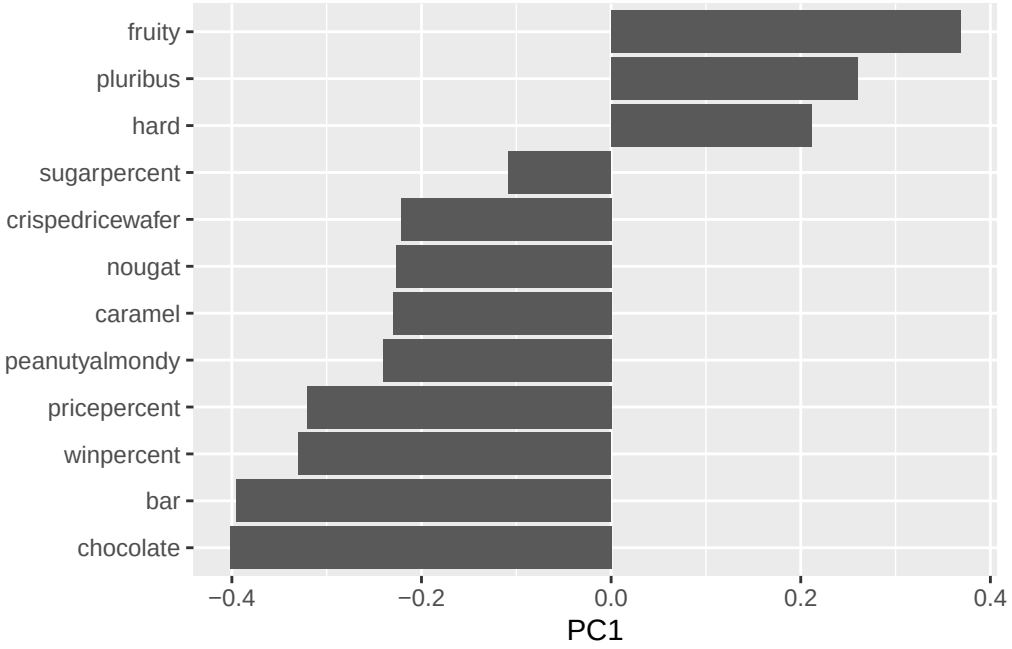
```
library(plotly)
ggplotly(p)
```


2
1
0
-1
-2
-3
PC2
-2.5
0.0
PC1

N.B. Note that this plot is an *"interactive plot"*, you can mouse over the points to obtain more information on them (called a "tooltip" in nerd speak) and "pan" to select a sub-region of the plot to zoom in to see more details. Give it try! Also note that this interactivity will only work with HTML output and not in PDFs or MSWord documents for example. This will become important later when you render your lab report for GradeScope

Let's finish this section by taking a quick look at our PCA loadings. Do these make sense to you? Notice the opposite effects of `chocolate` and `fruity` and the similar effects of `chocolate` and `bar` (i.e. we already know they are correlated).

```
ggplot(pca$rotation) +
  aes(___, ___) +
  geom____()
```



- **Q25**. Complete the code to generate the loadings plot above. What original variables are picked up strongly by PC1 in the positive direction? Do these make sense to you? Where did you see this relationship highlighted previously?

- **Q26**. Based on your exploratory analysis, correlation findings, and PCA results, what combination of characteristics appears to make a "winning" candy? How do these different analyses (visualization, correlation, PCA) support or complement each other in reaching this conclusion?

PCA from Scratch on a mystery dataset

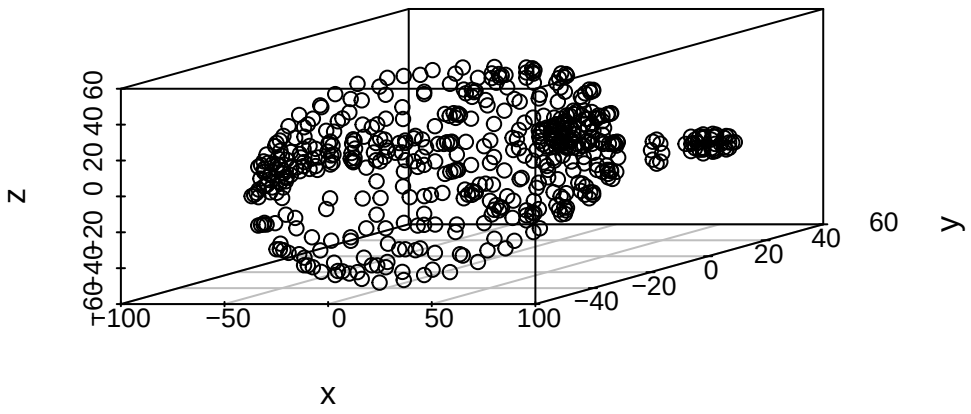
So far we have used R's built-in `prcomp()` function to perform PCA. But what is it actually doing mathematically? In this optional section we will perform PCA "by hand" using matrix algebra and connect this back to what `prcomp()` computes for us automatically. To make things concrete — and a little more fun — we will use a mystery 3D dataset.

```
# Read a mystery 3D dataset - what could it be?
m <- read.csv("mystery.csv")
head(m)
```

```
     x    y    z
1 40.6 28.3 -1.1
2 40.1 30.4 -1.1
3 40.7 31.1 -1.1
4 42.0 30.4 -1.1
5 43.5 28.3 -1.1
6 37.5 28.3 14.5
```

Let's start by visualizing the data as a scatter plot in 3D:

```
library(scatterplot3d)
scatterplot3d(m)
```



- **Q27.** Hmm… can you tell what the mystery dataset represents from this 3D plot? Hint: It is something that goes well with chocolate in Barry's opnion.

Perhaps it is not particularly clear yet so let's perform PCA step by step. Recall from our candy analysis that PCA operates on the covariance structure of the data. Our first step is therefore to compute the covariance matrix with the `cov()` function:

```
# Step 1: Compute the covariance matrix
cij <- cov(m)
cij
```

```
             x           y           z
x 2036.5727580 134.9131571  -0.2923907
y  134.9131571 761.6970573   0.0343026
z   -0.2923907   0.0343026 553.4070615
```

> **Side-note**: Notice we are using `cov()` here rather than `cor()`. Because our x, y, and z coordinates are all measured on the same scale (they are all spatial dimensions of the same object), we do not need to standardize them — so we work with the covariance matrix directly rather than the correlation matrix. This is why we will not use `scale=TRUE` when we call `prcomp()` later on when we check our work. Compare this to our candy analysis where variables were on very different scales and scaling was essential.

PCA finds the directions of maximum variance in the data by computing the eigenvectors of the covariance matrix. The eigenvectors define the new axes (principal components) and the corresponding eigenvalues tell us how much variance each axis captures:

```
# Step 2: Eigen decomposition of the covariance matrix
ans <- eigen(cij)

eigenval <- ans$values   # variance captured by each PC
eigenvec <- ans$vectors  # the new axes (principal components)

# How much variance does each PC capture?
round(cumsum(eigenval) / sum(eigenval) * 100, 2)
```

```
[1]  61.18  83.49 100.00
```

- **Q28.** What percentage of the total variance is captured by the first two principal components together? Does this seem like enough to get a meaningful 2D representation of a 3D object?

As we have a large amount of variance captured in our first few PCs we can get a useful lower dimensional representation (e.g. a 2D representation with a PC1 vs PC2 plot) by "projecting" our original data onto the first two eigenvectors or PCs.

This projection is simply a matrix multiplication — we are asking "where does each data point land (or"score") when we look at the data from the direction of maximum variance¿':

```
# Step 3: Project data onto the top 2 eigenvectors
scores <- as.matrix(m) %*% eigenvec[, 1:2]

# Set some colnames for easier plotting
colnames(scores) <- c("PC1", "PC2")
```

```
ggplot(___) +
  aes(___, ___) +
  geom_point()
```

- **Q29.** What is the mystery dataset? Does it make sense that PCA was able to recover a recognizable 2D view from 3D data? What has been "lost" in the reduction from 3D to 2D?

key-point Note that PCA was able to recover "better" axis of maximal variance that allow us to plot a "better" (i.e. more recognizable) 2D view from our original 3D data.

Finally, let's verify our "by-hand" calculation against `prcomp()` and plot these results with ggplot:

```
# Verify with prcomp() - note no scaling needed here
p <- prcomp(m)
summary(p)
```

```
Importance of components:
                          PC1     PC2     PC3
Standard deviation     45.2846 27.3418 23.5246
Proportion of Variance  0.6118  0.2230  0.1651
Cumulative Proportion   0.6118  0.8349  1.0000
```

And a nicer ggplot version:

```
ggplot(p$x) +
  aes(PC1, PC2) +
  geom_point(size=2, col="steelblue", alpha=0.5) +
  labs(title="Mystery dataset: PCA projection",
       subtitle="3D structure recovered in 2D") +
  theme_bw()
```

- **Q30.** Do the results from prcomp() match your manual eigen decomposition? The scores may differ in sign (direction) — this is normal and does not affect interpretation. Why do you think this is?

Summary

Starting with basic data import and exploration, you characterized the structure of the candy dataset and identified key variables. You then built increasingly sophisticated visualizations—from simple histograms to ranked bar plots to labeled scatter plots—to reveal patterns in candy popularity and pricing.

The correlation analysis showed you how candy attributes relate to each other (chocolate and fruity are essentially mutually exclusive, for instance), and this set the stage for PCA. By reducing the original variables down to a few PCs, you created a "map of candy space" where similar candies cluster together. The PCA loadings revealed that the primary axis of variation separates chocolate-based candies from fruity ones - the same pattern we observed in the correlation matrix, now visualized in a single, interpretable plot.

Optional extension questions

```
losers = candy[which(candy$winpercent < 50),]
winners = candy[which(candy$winpercent >= 50),]
```

- **Q31**. Are popular candies more expensive? In other words: is price significantly different between "winners" and "losers"? List both average values and a P-value along with your answer.
- **Q32**. Are candies with more sugar more likely to be popular? What is your interpretation of the means and P-value in this case?