Class9

Hanhee Jo (PID: A59017994)

Table of contents

Background 1
1. Import the data 1
2. What is your favorite candy? 2
3. Overall Candy Rankings 9
4. Taking a look at pricepercent 17
5. Exploring the correlation structure 27
6. Principal Component Analysis (PCA) 28

Background

Today we are delving into an analysis of Halloween Candy data using ggplot, dplyr, basic stats, correlation analysis and our old friend PCA.

1. Import the data

```
candy <- read.csv("candy-data.csv", row.names=1)
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
```

Q1. How many candy types are in this dataset?

```
nrow(candy)
```

```
[1] 85
```

Q2. How many fruity candy types are in the dataset?

```
sum(candy$fruity)
```

```
[1] 38
```

Q. How many chocolat candy types are in the dataset?

```
sum(candy$chocolate)
```

```
[1] 37
```

2. What is your favorite candy?

Q3. What is your favorite candy in the dataset and what is it's winpercent value?

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

```
[1] 81.64291
```

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

```
[1] 81.64291
```

```
library(dplyr)
```

We can also use the `filter()` and `select()` functions from **dplyr**

```
candy |>
  filter(rownames(candy) == "Twix") |>
  select(winpercent, sugarpercent)
```

```
     winpercent sugarpercent
Twix   81.64291        0.546
```

```
candy |>
  filter(rownames(candy) == "Haribo Gold Bears") |>
  select(winpercent, sugarpercent)
```

```
                 winpercent sugarpercent
Haribo Gold Bears   57.11974        0.465
```

A useful function for a quqick look at a new dataset is found in the **skimr** package.

```
skimr::skim(candy)
```

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	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
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	

Q6. Is there any variable/column that looks to be on a different scale to the majority of the other columns in the dataset?

`winpercent` column is on a different "scale" or range than all the others.

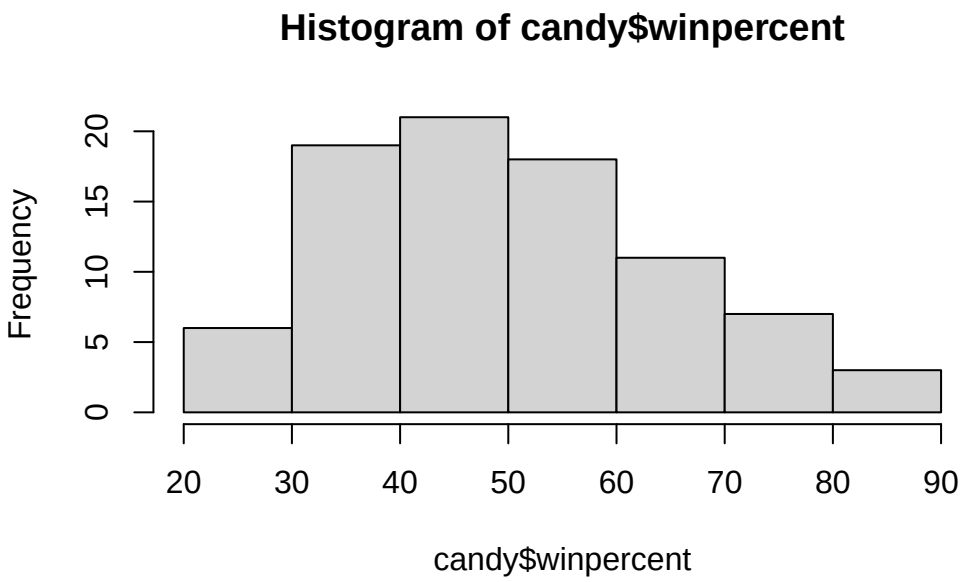
N.B We will need to scale this data before analysis like PCA for example to avoid this one variable dominating our analysis.

Q7. What do you think a zero and one represent for the candy$chocolate column?

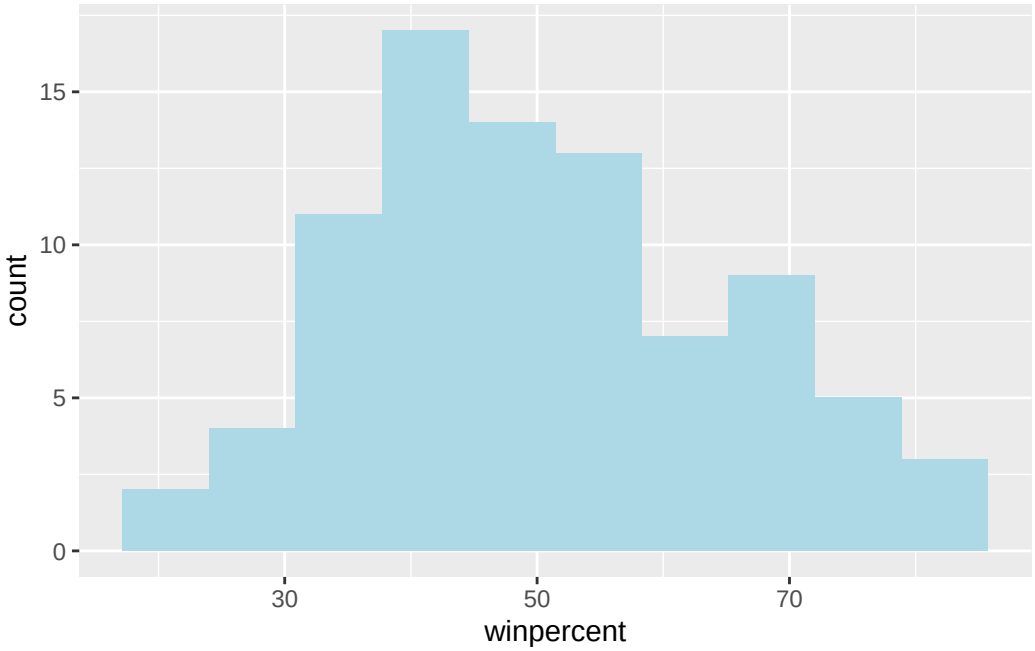
That the candy has the chocolate.

Q8. Plot a histogram of winpercent values usind base R and ggplot.

```
hist(candy$winpercent)
```




```
library(ggplot2)

ggplot(candy, aes(winpercent)) +
  geom_histogram(bins=10, fill = "lightblue")
```



Q9. Is the distribution of winpercent values symmetrical?

No

Q10. Is the center of the distribution above or below 50%?

From the histogram, it looks to be below 50% mark.

```
summary(candy$winpercent)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  22.45   39.14   47.83   50.32   59.86   84.18 
```

```
#The median is below 50% but the mean is above 50%.
```

Q11. On average is chocolate candy higher or lower ranked than fruit candy?

Step 1. Find/extract chocolate candy rows in the dataset. Step 2. Get their `winpercent` values. Step 3. Calculate their mean `winpercent` values.

Step 4. Find/extract fruity candy rows in the dataset. Step 5. Get their `winpercent` values. Step 6. Calculate their mean `winpercent` values.

Step 7. Compare mean chocolate `winpercent` to mean fruity `winpercent` and see which one is larger.

1. Find chocolate candy

```
choc.inds <- candy$chocolate == 1
choc.candy <- candy[choc.inds, ]
```

2. Get their `winpercent` values.

```
choc.win <- choc.candy$winpercent
```

3. Get their mean

```
mean(choc.win)
```

```
[1] 60.92153
```

4. Find fruity candy

```
fruit.inds <- as.logical(candy$fruity)
fruit.candy <- candy[fruit.inds, ]
```

5. Get their `winpercent` values.

```
fruit.win <- fruit.candy$winpercent
```

6. Get their mean

```
mean(fruit.win)
```

```
[1] 44.11974
```

You can also use **dplyr** package.

```
candy |>
  filter(chocolate == 1) |>
  select(winpercent)
```

	winpercent
100 Grand	66.97173
3 Musketeers	67.60294
Almond Joy	50.34755
Baby Ruth	56.91455
Charleston Chew	38.97504
Hershey's Kisses	55.37545
Hershey's Krackel	62.28448
Hershey's Milk Chocolate	56.49050
Hershey's Special Dark	59.23612
Junior Mints	57.21925
Kit Kat	76.76860
Peanut butter M&M's	71.46505
M&M's	66.57458
Milk Duds	55.06407
Milky Way	73.09956
Milky Way Midnight	60.80070
Milky Way Simply Caramel	64.35334
Mounds	47.82975
Mr Good Bar	54.52645
Nestle Butterfinger	70.73564
Nestle Crunch	66.47068
Peanut M&Ms	69.48379
Reese's Miniatures	81.86626
Reese's Peanut Butter cup	84.18029
Reese's pieces	73.43499
Reese's stuffed with pieces	72.88790
Rolo	65.71629
Sixlets	34.72200
Nestle Smarties	37.88719
Snickers	76.67378
Snickers Crisper	59.52925
Tootsie Pop	48.98265
Tootsie Roll Juniors	43.06890
Tootsie Roll Midgies	45.73675
Tootsie Roll Snack Bars	49.65350
Twix	81.64291
Whoppers	49.52411

Q12. Is this difference statistically significant?

Let's use a t.test

```
  t.test(choc.win, fruit.win)
```

```
    Welch Two Sample t-test

data:  choc.win and fruit.win
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
```

3. Overall Candy Rankings

Q13. What are the five least liked candy types in this set?

```
  #sort(candy$winpercent)

  x <- c(10, 1, 100)
  sort(x)
```

```
[1]   1  10 100
```

```
  order(x)
```

```
[1] 2 1 3
```

So I can use the output of `order(winpercent)` to re-arrange (or order) my whole dataset by `winpercent`

```
  order.inds <-order(candy$winpercent)
  head( candy[order.inds, ], 5 )
```

```
                  chocolate fruity caramel peanutyalmondy nougat
Nik L Nip                 0      1       0              0      0
Boston Baked Beans        0      0       0              1      0
Chiclets                  0      1       0              0      0
```

```
Super Bubble               0      1       0              0      0
Jawbusters                 0      1       0              0      0
                  crispedricewafer hard bar pluribus sugarpercent pricepercent
Nik L Nip                        0    0   0        1        0.197        0.976
Boston Baked Beans               0    0   0        1        0.313        0.511
Chiclets                         0    0   0        1        0.046        0.325
Super Bubble                     0    0   0        0        0.162        0.116
Jawbusters                       0    1   0        1        0.093        0.511
                  winpercent
Nik L Nip           22.44534
Boston Baked Beans  23.41782
Chiclets            24.52499
Super Bubble        27.30386
Jawbusters          28.12744
```

```
candy |>
  arrange(winpercent) |>
  head(5)
```

```
                  chocolate fruity caramel peanutyalmondy nougat
Nik L Nip                 0      1       0              0      0
Boston Baked Beans        0      0       0              1      0
Chiclets                  0      1       0              0      0
Super Bubble              0      1       0              0      0
Jawbusters                0      1       0              0      0
                  crispedricewafer hard bar pluribus sugarpercent pricepercent
Nik L Nip                        0    0   0        1        0.197        0.976
Boston Baked Beans               0    0   0        1        0.313        0.511
Chiclets                         0    0   0        1        0.046        0.325
Super Bubble                     0    0   0        0        0.162        0.116
Jawbusters                       0    1   0        1        0.093        0.511
                  winpercent
Nik L Nip           22.44534
Boston Baked Beans  23.41782
Chiclets            24.52499
Super Bubble        27.30386
Jawbusters          28.12744
```

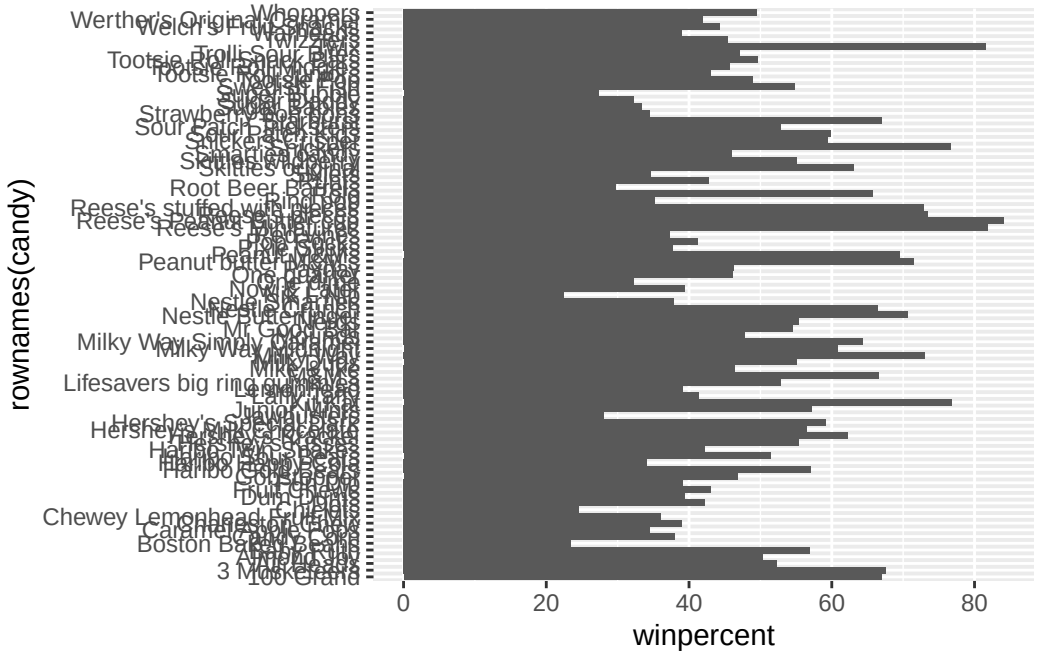
Q14. What are the top 5 all time favorite candy types out of this set?

```
candy |>
  arrange(-winpercent) |>
  head(5)
```

```
                         chocolate fruity caramel peanutyalmondy nougat
Reese's Peanut Butter cup         1      0       0              1      0
Reese's Miniatures                1      0       0              1      0
Twix                              1      0       1              0      0
Kit Kat                           1      0       0              0      0
Snickers                          1      0       1              1      1
                         crispedricewafer hard bar pluribus sugarpercent
Reese's Peanut Butter cup                0    0   0        0        0.720
Reese's Miniatures                       0    0   0        0        0.034
Twix                                     1    0   1        0        0.546
Kit Kat                                  1    0   1        0        0.313
Snickers                                 0    0   1        0        0.546
                         pricepercent winpercent
Reese's Peanut Butter cup        0.651   84.18029
Reese's Miniatures               0.279   81.86626
Twix                             0.906   81.64291
Kit Kat                          0.511   76.76860
Snickers                         0.651   76.67378
```

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

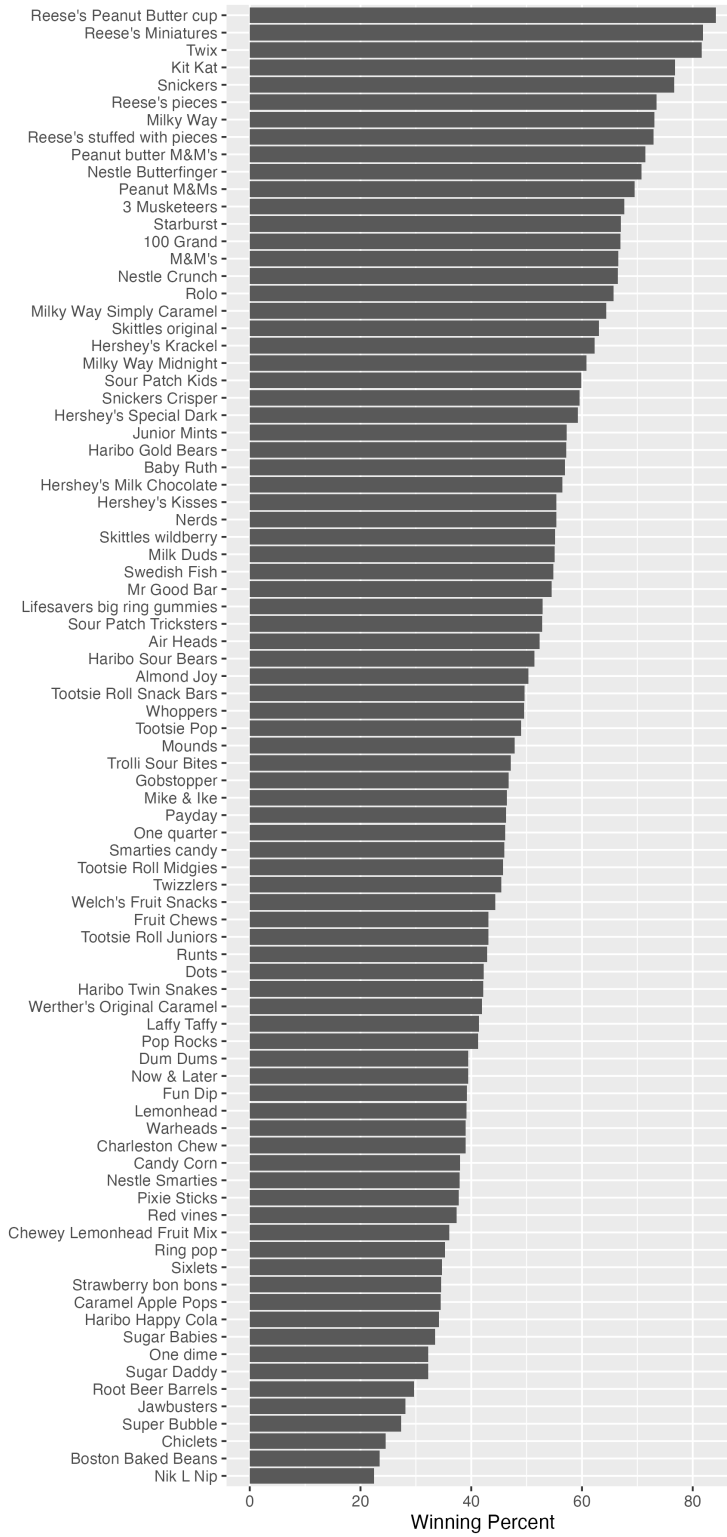
```
ggplot(candy) +
  aes(winpercent, rownames(candy)) +
  geom_col()
```



We can make this plot better by rearranging (ordering) the y-axis by winpercent so the highest scoring candy is at the top and lowest at the bottom.

```
p <- ggplot(candy) +
  aes(x=winpercent,
      y=reorder(rownames(candy), winpercent) ) +
  geom_col() +
  ylab("") +
  xlab("Winning Percent")
```

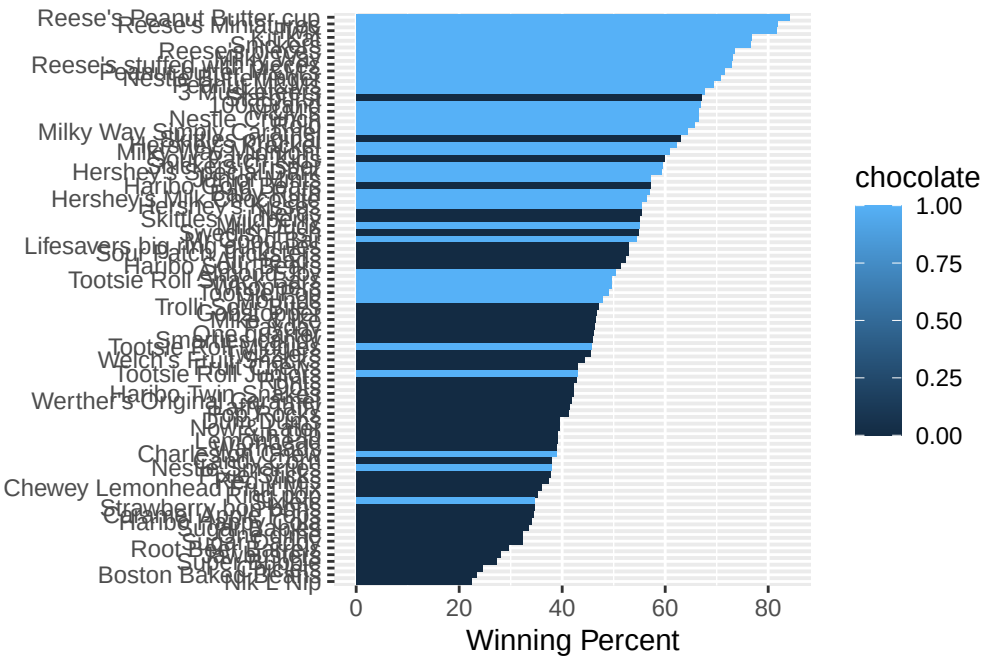
```
ggsave("my_plot.png", height=12, width=6)
```



This will insert the image in the rendered document.

Q. Color your bars by "chocolate"

```
ggplot(candy) +
  aes(x=winpercent,
      y=reorder(rownames(candy), winpercent), fill=chocolate) +
  geom_col() +
  ylab("") +
  xlab("Winning Percent")
```



I want to color chocolate and fruity candy a specified color. To do this, we need to define our own custom color vector that has the exact colot mappings we want.

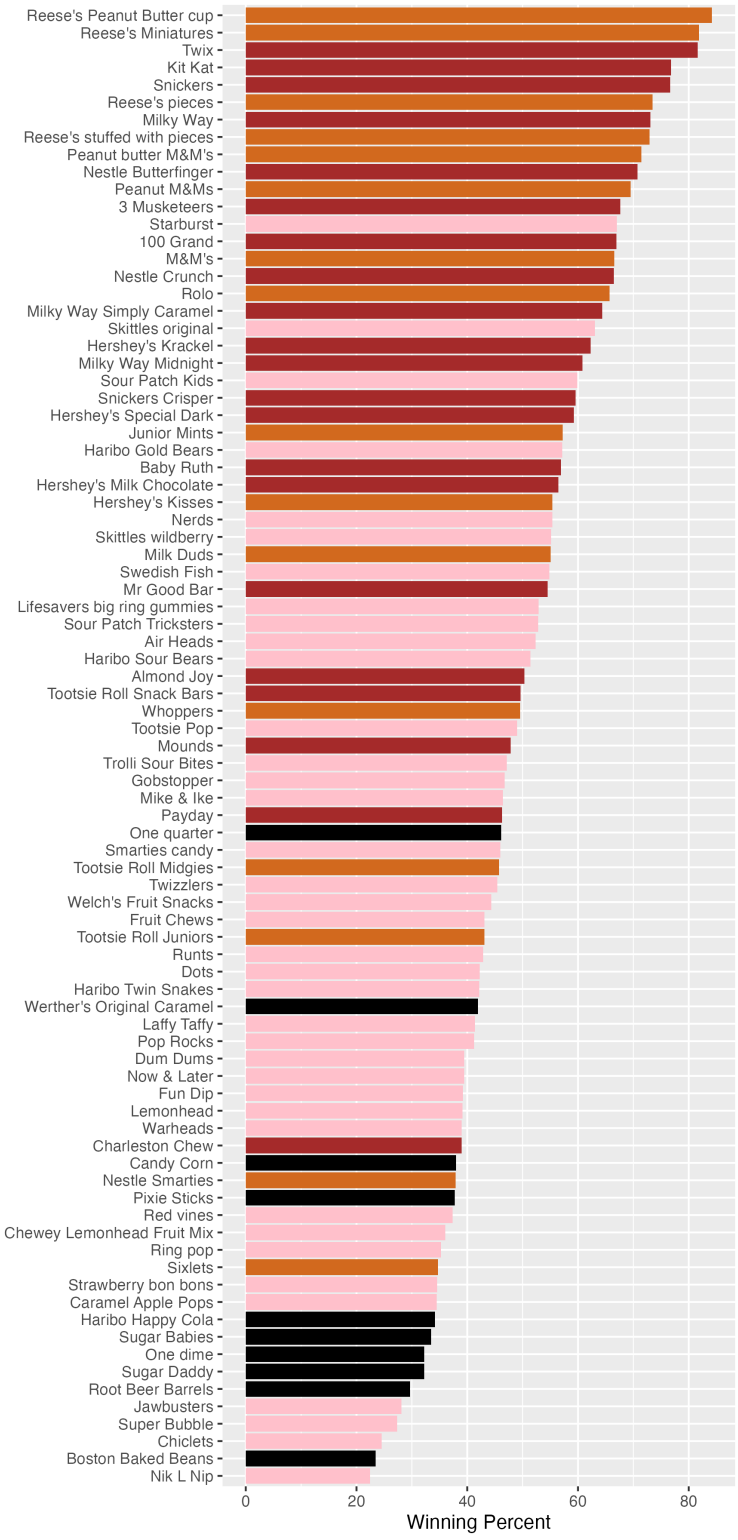
```
mycols <- rep("black", nrow(candy))

mycols[candy$chocolate == 1] <- "chocolate"
mycols[candy$bar == 1] <- "brown"
mycols[candy$fruity == 1] <- "pink"

mycols
```

```
 [1] "brown"     "brown"     "black"     "black"     "pink"      "brown"
```

```
 [7] "brown"     "black"     "black"     "pink"      "brown"     "pink"
[13] "pink"      "pink"      "pink"      "pink"      "pink"      "pink"
[19] "pink"      "black"     "pink"      "pink"      "chocolate" "brown"
[25] "brown"     "brown"     "pink"      "chocolate" "brown"     "pink"
[31] "pink"      "pink"      "chocolate" "chocolate" "pink"      "chocolate"
[37] "brown"     "brown"     "brown"     "brown"     "brown"     "pink"
[43] "brown"     "brown"     "pink"      "pink"      "brown"     "chocolate"
[49] "black"     "pink"      "pink"      "chocolate" "chocolate" "chocolate"
[55] "chocolate" "pink"      "chocolate" "black"     "pink"      "chocolate"
[61] "pink"      "pink"      "chocolate" "pink"      "brown"     "brown"
[67] "pink"      "pink"      "pink"      "pink"      "black"     "black"
[73] "pink"      "pink"      "pink"      "chocolate" "chocolate" "brown"
[79] "pink"      "brown"     "pink"      "pink"      "pink"      "black"
[85] "chocolate"
```

```
p <- ggplot(candy) +
  aes(x=winpercent,
      y=reorder(rownames(candy), winpercent)) +
  geom_col(fill=mycols) +
  ylab("") +
  xlab("Winning Percent")
```

```
ggsave("my_color_plot.png", heigh=12, width=6)
```


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
Winning Percent

Q17. What is the worst ranked chocolate candy?

sixlets

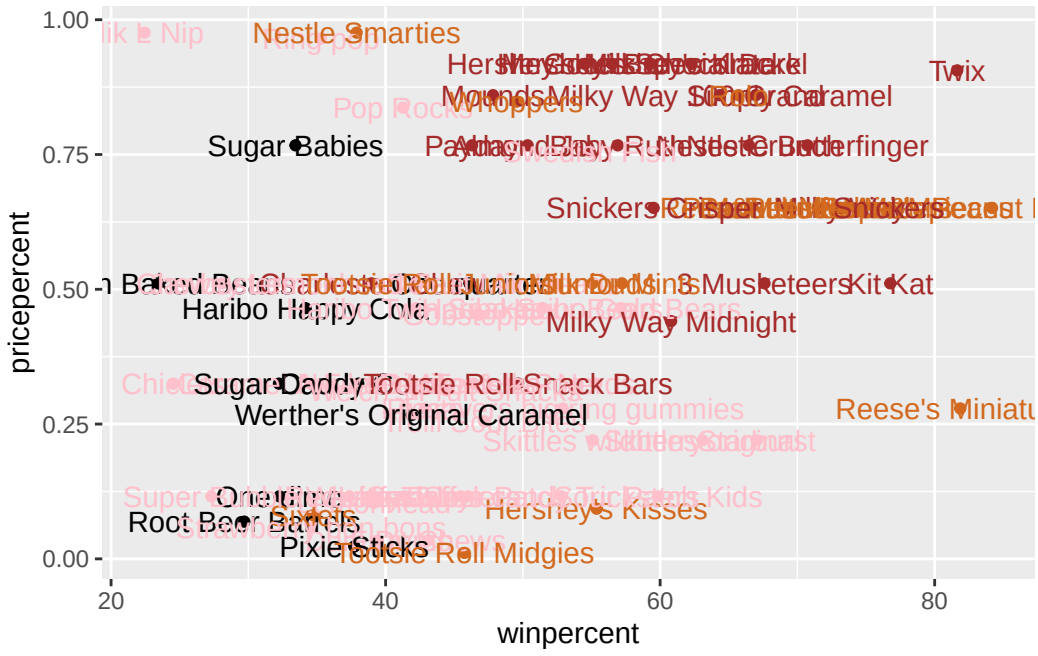
Q18. What is the best ranked fruity candy?

starburst

4. Taking a look at pricepercent

Plot of winpercent vs pricepercent

```
ggplot(candy) +
  aes(x=winpercent,
      y=pricepercent,
      label=rownames(candy)) +
  geom_point(col=mycols) +
  geom_text(col=mycols)
```




```
  theme_bw()
```

```
List of 136
 $ line                            :List of 6
  ..$ colour       : chr "black"
  ..$ linewidth    : num 0.5
  ..$ linetype     : num 1
  ..$ lineend      : chr "butt"
  ..$ arrow        : logi FALSE
  ..$ inherit.blank: logi TRUE
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ rect                            :List of 5
  ..$ fill         : chr "white"
  ..$ colour       : chr "black"
  ..$ linewidth    : num 0.5
  ..$ linetype     : num 1
  ..$ inherit.blank: logi TRUE
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ text                            :List of 11
  ..$ family       : chr ""
  ..$ face         : chr "plain"
  ..$ colour       : chr "black"
  ..$ size         : num 11
  ..$ hjust        : num 0.5
  ..$ vjust        : num 0.5
  ..$ angle        : num 0
  ..$ lineheight   : num 0.9
  ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
  .. ..- attr(*, "unit")= int 8
  ..$ debug        : logi FALSE
  ..$ inherit.blank: logi TRUE
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ title                           : NULL
 $ aspect.ratio                    : NULL
 $ axis.title                      : NULL
 $ axis.title.x                    :List of 11
  ..$ family       : NULL
  ..$ face         : NULL
  ..$ colour       : NULL
  ..$ size         : NULL
  ..$ hjust        : NULL
  ..$ vjust        : num 1
  ..$ angle        : NULL
  ..$ lineheight   : NULL
  ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
```

```
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.title.x.top               :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : NULL
 ..$ vjust        : num 0
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.title.x.bottom            : NULL
$ axis.title.y                   :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : NULL
 ..$ vjust        : num 1
 ..$ angle        : num 90
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.title.y.left              : NULL
$ axis.title.y.right             :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : NULL
 ..$ vjust        : num 1
 ..$ angle        : num -90
 ..$ lineheight   : NULL
```

```
 ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text                      :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : chr "grey30"
 ..$ size         : 'rel' num 0.8
 ..$ hjust        : NULL
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : NULL
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text.x                    :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : NULL
 ..$ vjust        : num 1
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text.x.top                :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : NULL
 ..$ vjust        : num 0
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
 .. ..- attr(*, "unit")= int 8
```

```
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text.x.bottom              : NULL
$ axis.text.y                     :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : num 1
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text.y.left                : NULL
$ axis.text.y.right               :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : num 0
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.text.theta                 : NULL
$ axis.text.r                     :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : num 0.5
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
```

```
 ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 2.2points
 .. ..- attr(*, "unit")= int 8
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ axis.ticks                     :List of 6
 ..$ colour       : chr "grey20"
 ..$ linewidth    : NULL
 ..$ linetype     : NULL
 ..$ lineend      : NULL
 ..$ arrow        : logi FALSE
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_line" "element"
$ axis.ticks.x                   : NULL
$ axis.ticks.x.top               : NULL
$ axis.ticks.x.bottom            : NULL
$ axis.ticks.y                   : NULL
$ axis.ticks.y.left              : NULL
$ axis.ticks.y.right             : NULL
$ axis.ticks.theta               : NULL
$ axis.ticks.r                   : NULL
$ axis.minor.ticks.x.top         : NULL
$ axis.minor.ticks.x.bottom      : NULL
$ axis.minor.ticks.y.left        : NULL
$ axis.minor.ticks.y.right       : NULL
$ axis.minor.ticks.theta         : NULL
$ axis.minor.ticks.r             : NULL
$ axis.ticks.length              : 'simpleUnit' num 2.75points
 ..- attr(*, "unit")= int 8
$ axis.ticks.length.x            : NULL
$ axis.ticks.length.x.top        : NULL
$ axis.ticks.length.x.bottom     : NULL
$ axis.ticks.length.y            : NULL
$ axis.ticks.length.y.left       : NULL
$ axis.ticks.length.y.right      : NULL
$ axis.ticks.length.theta        : NULL
$ axis.ticks.length.r            : NULL
$ axis.minor.ticks.length        : 'rel' num 0.75
$ axis.minor.ticks.length.x      : NULL
$ axis.minor.ticks.length.x.top  : NULL
$ axis.minor.ticks.length.x.bottom: NULL
$ axis.minor.ticks.length.y      : NULL
$ axis.minor.ticks.length.y.left : NULL
```

```
$ axis.minor.ticks.length.y.right : NULL
$ axis.minor.ticks.length.theta   : NULL
$ axis.minor.ticks.length.r       : NULL
$ axis.line                       : list()
 ..- attr(*, "class")= chr [1:2] "element_blank" "element"
$ axis.line.x                     : NULL
$ axis.line.x.top                 : NULL
$ axis.line.x.bottom              : NULL
$ axis.line.y                     : NULL
$ axis.line.y.left                : NULL
$ axis.line.y.right               : NULL
$ axis.line.theta                 : NULL
$ axis.line.r                     : NULL
$ legend.background               :List of 5
 ..$ fill         : NULL
 ..$ colour       : logi NA
 ..$ linewidth    : NULL
 ..$ linetype     : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_rect" "element"
$ legend.margin                   : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5poi
 ..- attr(*, "unit")= int 8
$ legend.spacing                  : 'simpleUnit' num 11points
 ..- attr(*, "unit")= int 8
$ legend.spacing.x                : NULL
$ legend.spacing.y                : NULL
$ legend.key                      : NULL
$ legend.key.size                 : 'simpleUnit' num 1.2lines
 ..- attr(*, "unit")= int 3
$ legend.key.height               : NULL
$ legend.key.width                : NULL
$ legend.key.spacing              : 'simpleUnit' num 5.5points
 ..- attr(*, "unit")= int 8
$ legend.key.spacing.x            : NULL
$ legend.key.spacing.y            : NULL
$ legend.frame                    : NULL
$ legend.ticks                    : NULL
$ legend.ticks.length             : 'rel' num 0.2
$ legend.axis.line                : NULL
$ legend.text                     :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
```

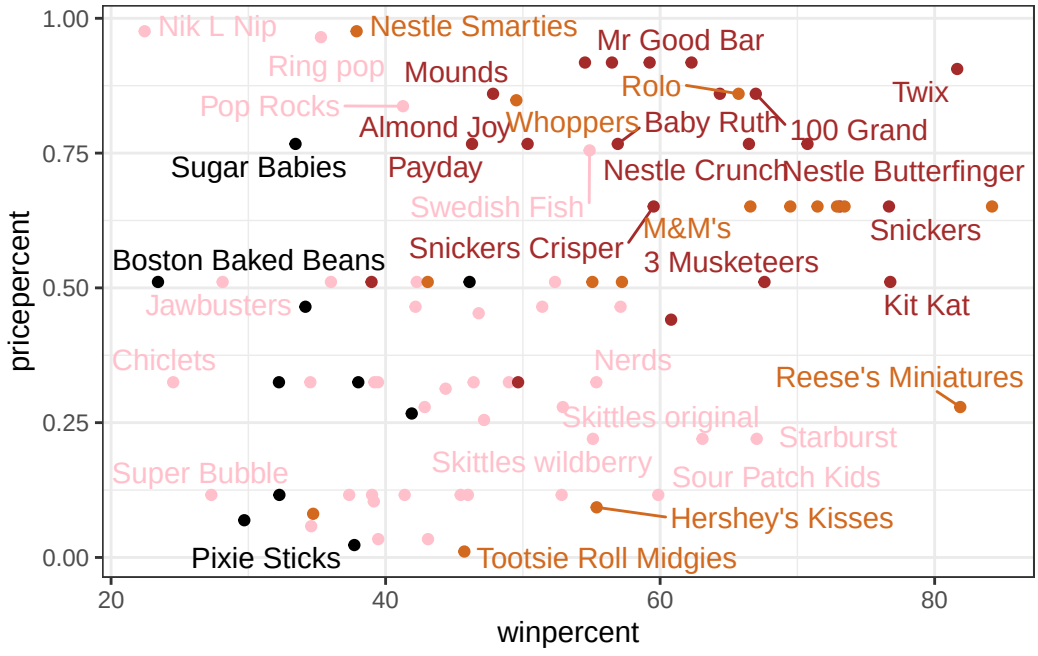
```
 ..$ size         : 'rel' num 0.8
 ..$ hjust        : NULL
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : NULL
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ legend.text.position            : NULL
$ legend.title                    :List of 11
 ..$ family       : NULL
 ..$ face         : NULL
 ..$ colour       : NULL
 ..$ size         : NULL
 ..$ hjust        : num 0
 ..$ vjust        : NULL
 ..$ angle        : NULL
 ..$ lineheight   : NULL
 ..$ margin       : NULL
 ..$ debug        : NULL
 ..$ inherit.blank: logi TRUE
 ..- attr(*, "class")= chr [1:2] "element_text" "element"
$ legend.title.position           : NULL
$ legend.position                 : chr "right"
$ legend.position.inside          : NULL
$ legend.direction                : NULL
$ legend.byrow                    : NULL
$ legend.justification            : chr "center"
$ legend.justification.top        : NULL
$ legend.justification.bottom     : NULL
$ legend.justification.left       : NULL
$ legend.justification.right      : NULL
$ legend.justification.inside     : NULL
$ legend.location                 : NULL
$ legend.box                      : NULL
$ legend.box.just                 : NULL
$ legend.box.margin               : 'margin' num [1:4] 0cm 0cm 0cm 0cm
 ..- attr(*, "unit")= int 1
$ legend.box.background           : list()
 ..- attr(*, "class")= chr [1:2] "element_blank" "element"
$ legend.box.spacing              : 'simpleUnit' num 11points
 ..- attr(*, "unit")= int 8
```

```
  [list output truncated]
 - attr(*, "class")= chr [1:2] "theme" "gg"
 - attr(*, "complete")= logi TRUE
 - attr(*, "validate")= logi TRUE
```

To avoid the common problem of label or text over-plotting we can use the **ggrepel** package like so:

```
library(ggrepel)

ggplot(candy) +
  aes(x=winpercent,
      y=pricepercent,
      label=rownames(candy)) +
  geom_point(col=mycols) +
  geom_text_repel(col=mycols) +
  theme_bw()
```

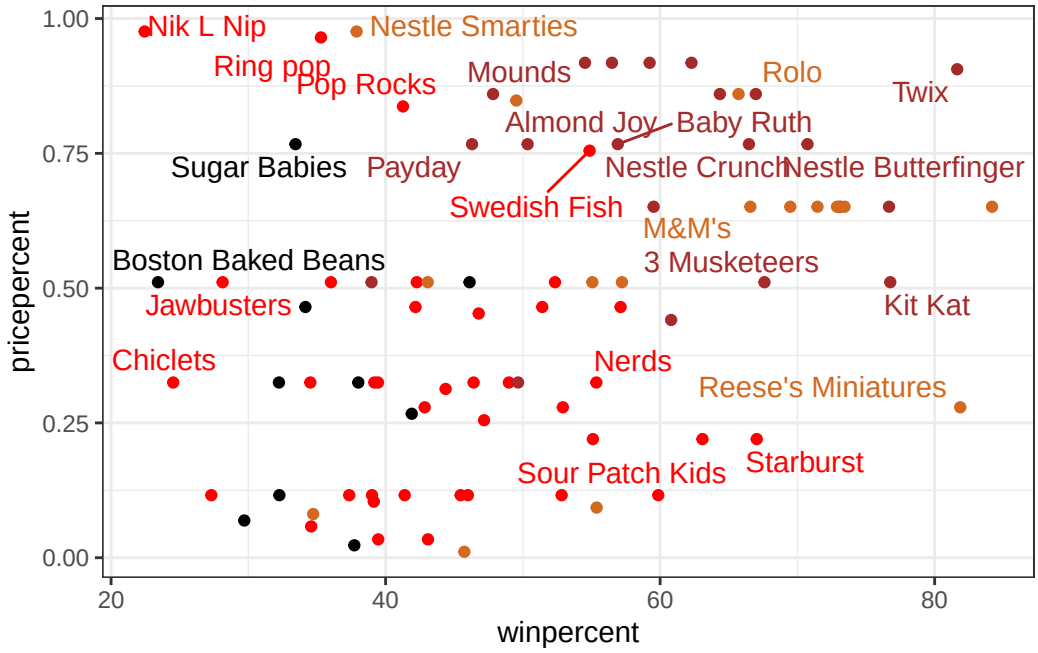
```
Warning: ggrepel: 50 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



We can control the amount of labels visible by setting different `max.overlaps` values:

```
# Change pink to be red for fruity candy
mycols[candy$fruity == 1] <- "red"

ggplot(candy) +
  aes(x=winpercent,
      y=pricepercent,
      label=rownames(candy)) +
  geom_point(col=mycols) +
  geom_text_repel(col=mycols, max.overlaps = 8) +
  theme_bw()
```

```
Warning: ggrepel: 61 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



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?

Reese's Miniatures

Q20. What are the top 5 most expensive candy types in the dataset and of these which is the least popular?

5. Exploring the correlation structure

The main function for correlation analysis in base R is called `cor()`

```
cij <- cor(candy)
head(cij)
```

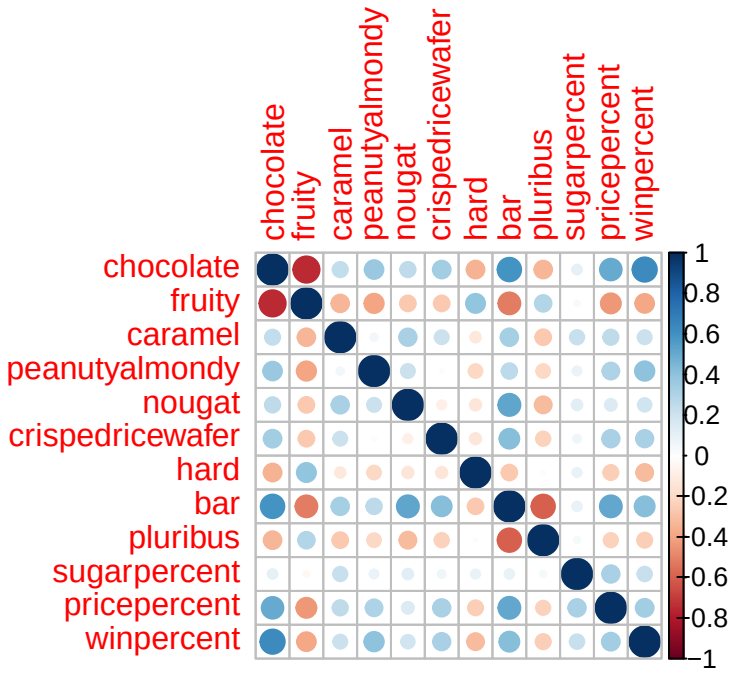
```
                  chocolate     fruity     caramel peanutyalmondy      nougat
chocolate         1.0000000 -0.7417211  0.24987535     0.37782357  0.25489183
fruity           -0.7417211  1.0000000 -0.33548538    -0.39928014 -0.26936712
caramel           0.2498753 -0.3354854  1.00000000     0.05935614  0.32849280
peanutyalmondy    0.3778236 -0.3992801  0.05935614     1.00000000  0.21311310
nougat            0.2548918 -0.2693671  0.32849280     0.21311310  1.00000000
crispedricewafer  0.3412098 -0.2693671  0.21311310    -0.01764631 -0.08974359
                  crispedricewafer       hard        bar   pluribus sugarpercent
chocolate              0.34120978 -0.3441769  0.5974211 -0.3396752   0.10416906
fruity                -0.26936712  0.3906775 -0.5150656  0.2997252  -0.03439296
caramel                0.21311310 -0.1223551  0.3339600 -0.2695850   0.22193335
peanutyalmondy        -0.01764631 -0.2055566  0.2604196 -0.2061093   0.08788927
nougat                -0.08974359 -0.1386750  0.5229764 -0.3103388   0.12308135
crispedricewafer       1.00000000 -0.1386750  0.4237509 -0.2246934   0.06994969
                  pricepercent winpercent
chocolate            0.5046754  0.6365167
fruity              -0.4309685 -0.3809381
caramel              0.2543271  0.2134163
peanutyalmondy       0.3091532  0.4061922
nougat               0.1531964  0.1993753
crispedricewafer     0.3282654  0.3246797
```

```
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.3.3
```

```
corrplot 0.95 loaded
```

```
corrplot(cij)
```



6. Principal Component Analysis (PCA)

We can use our old friend `prcomp()` finction with `scale=TRUE`.

```
pca <- prcomp(candy, scal=TRUE)
```

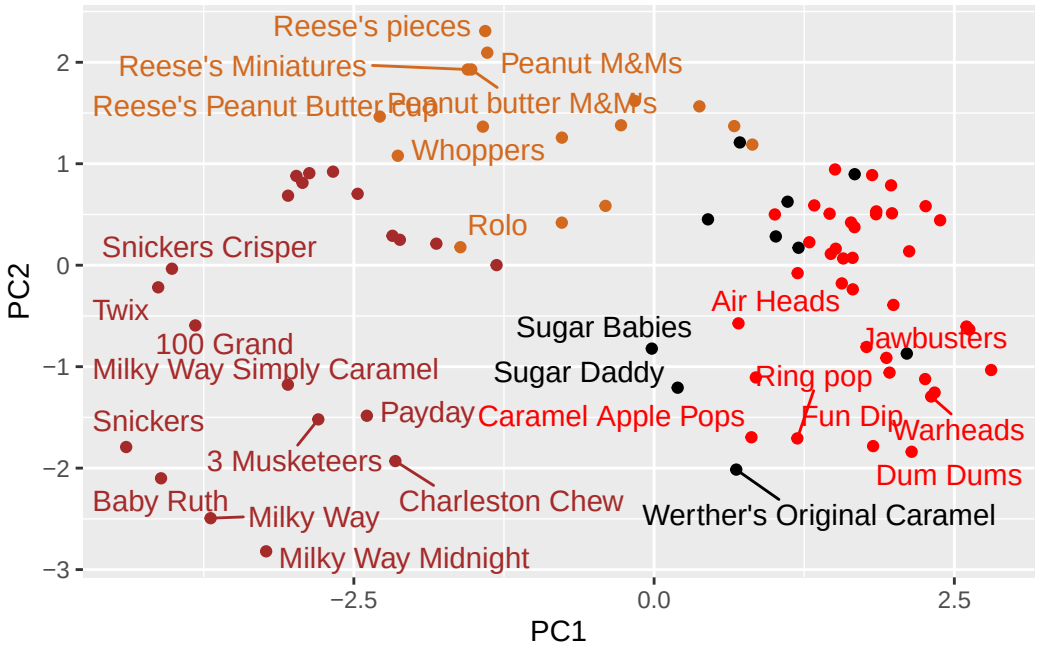
Let's make our main results figures, first our scroe plot (PC plot)

```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

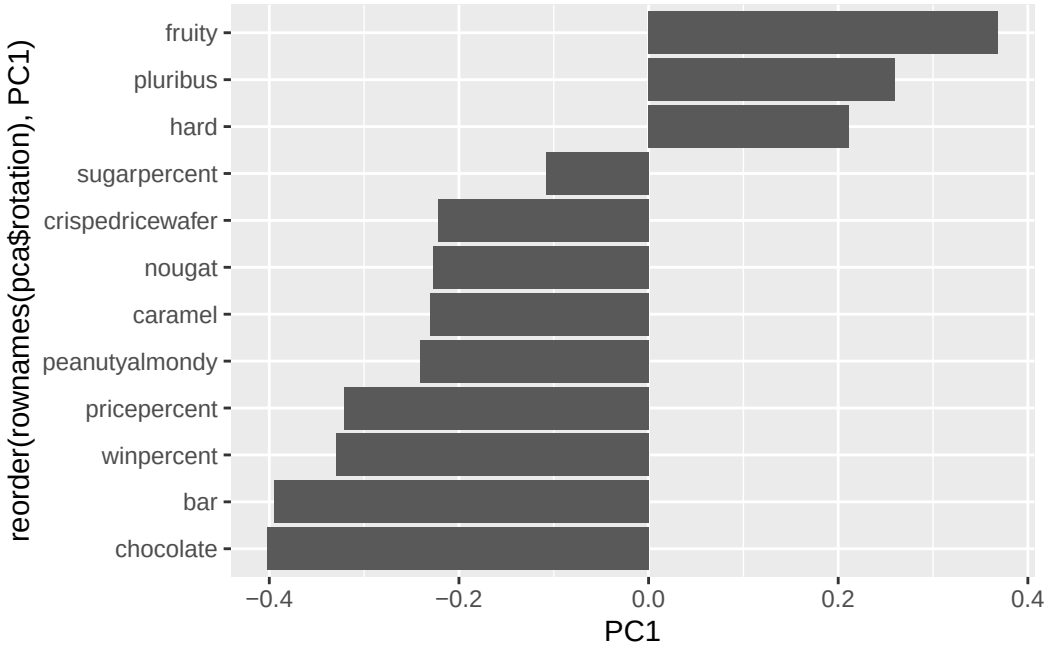
```
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(candy)) +
  geom_point(col=mycols) +
  geom_text_repel(col=mycols, max.overlaps = 8)
```

```
Warning: ggrepel: 57 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



Let's look at how the original variables contribute to our new PC's - this is often called the variable "loadings"

```
ggplot(pca$rotation) +
  aes(x=PC1,
      y=reorder(rownames(pca$rotation), PC1)) +
  geom_col()
```


reorder(rownames(pca$rotation), PC1)
fruity
pluribus
hard
sugarpercent
crispedricewafer
nougat
caramel
peanutyalmondy
pricepercent
winpercent
bar
chocolate
−0.4
−0.2
0.0
0.2
0.4
PC1