# Basic R for Business Wrap

🕯🎄🎅🎄🕯

1201 Data Science: Tim Raiswell

December 18, 2018

## Setting Up in R

## Getting Started

1. Open R Studio. In the console type: `install.packages("pacman")`.
2. Now go to `File > New File > R Markdown > Document > PDF`. Name the file and hit 'Okay'.
3. In your new R Markdown file delete everything after the `##R Markdown` heading.
4. Now change the heading to `##Loading Libraries`.
5. Underneath your heading create a new R code chunk by hitting CTRL + ALT + "I". Type the following and hit CTRL + Enter to run it in the chunk.
6. Download the `casino.csv` data file from the link below. Hit the download button to the top right of the dropbox web-page:

bit.ly/2LmvYE3

# Let's load some packages and data

### Packages

```r
library(tidyverse) #install.packages('tidyverse')
library(randomcoloR) #install.packages('randomcoloR') Note the capital 'R' at the end.
library(janitor) #install.packages('janitor')
library(psych) #install.packages('psych')
library(factoextra) #install.packages('factoextra')
```

### Casino Data

```r
casino_raw <- read.csv('casino.csv', stringsAsFactors = FALSE) # p, col_namest= TRUE) # put the
```

## Quick Data Check

```
head(casino_raw)
```

```
##          X Slots BlackJack Craps Bacarrat Bingo Poker Other Total.Spend
## 1 Player 1  1013      6190  4276      868     0     0     0       12348
## 2 Player 2    68        23    23       12     0    28    53         207
## 3 Player 3   148         0     0        0     0     0     0         148
## 4 Player 4    63        17    28        9     0    23    52         193
## 5 Player 5    92        44    18       10     0    26    60         250
## 6 Player 6   658         0     0        0   106     0     0         764
```

## Cleaning the Data

We perform several important transformations on the data ahead of some cluster analysis later. First, we run `distinct` because there are several duplicated player names. Second, we convert column `X` to row names. This is not something you will have to do frequently, but because the unsupervised learning model we use later can only be used on numeric data, we need some record of how it clusters our casino players. So we index the data by player name. Third, we clean the variable names to make them machine-friendly (lowercase, no spaces).

```r
casino_clean <- casino_raw %>%
  distinct() %>%
  column_to_rownames("X") %>%
  clean_names()
```

## Describing the Data

```
describe(casino_clean)[, 1:9]
```

```
##             vars    n    mean      sd median trimmed    mad min   max
## slots          1 5000  291.77  326.86  104.0  239.94 154.19   0  1861
## black_jack     2 5000  283.29  899.17   24.0   72.10  35.58   0  7294
## craps          3 5000  267.63  933.88   16.0   45.85  23.72   0  7251
## bacarrat       4 5000   82.07  247.14    7.0   20.65  10.38   0  2254
## bingo          5 5000   10.09   31.94    0.0    0.00   0.00   0   213
## poker          6 5000   54.59  105.86   11.0   29.62  16.31   0   914
## other          7 5000  132.97  217.51   34.5   86.74  51.15   0  1025
## total_spend    8 5000 1122.42 2226.22  336.5  600.80 292.81  13 15582
```

```
# the code in the square brackets is just to cut the
# column numbers down so it fits on the slide. No need for you to enter the same.
```

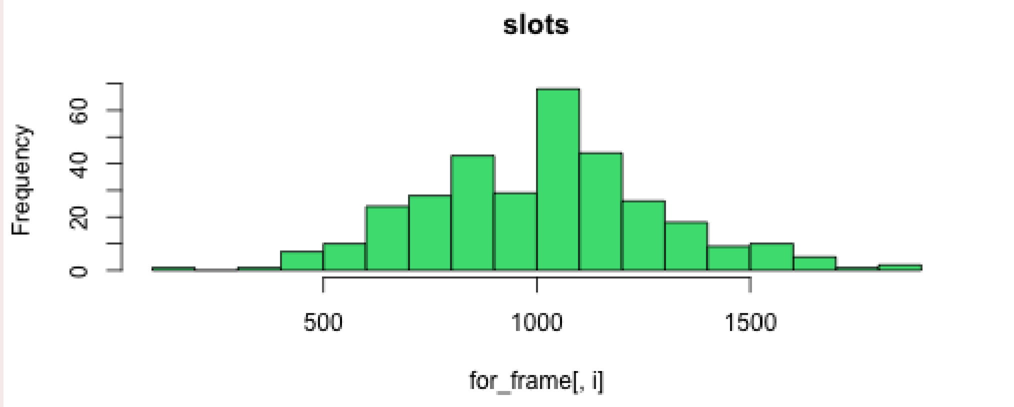
## For Loops and Functions

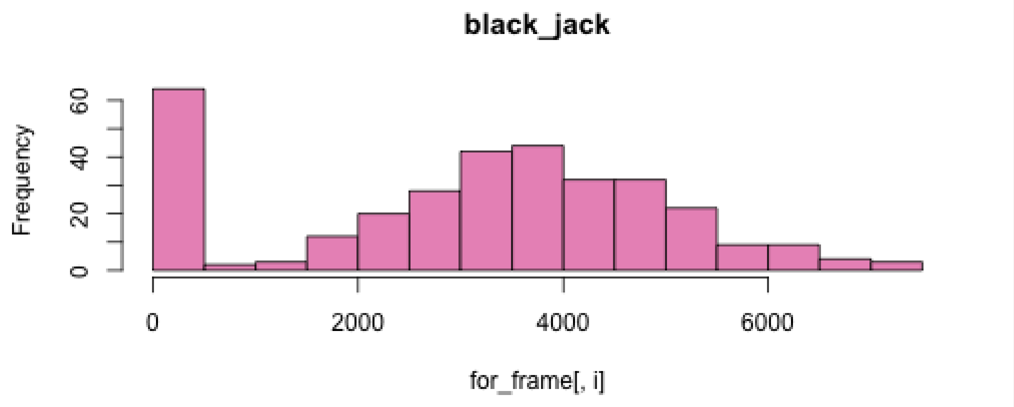
## For Loops

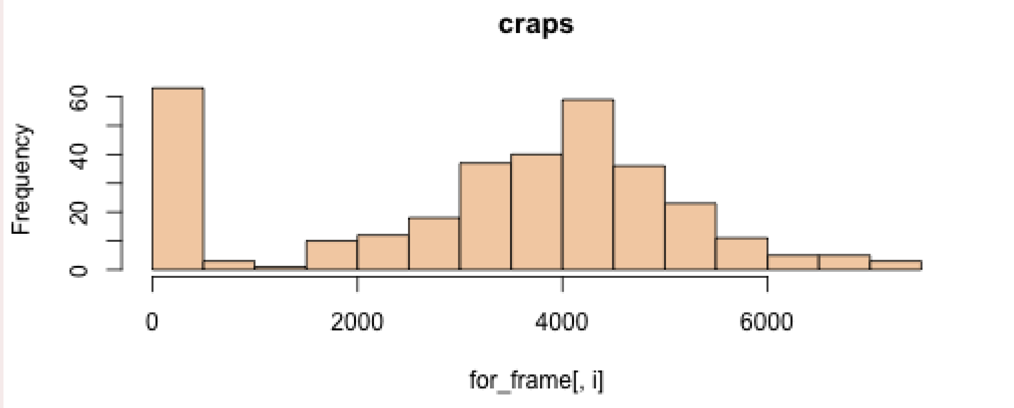
A for loop is a chunk of code that tells R to iterate `i`-times over a given list or vector object, performing some task or tasks for each iteration.

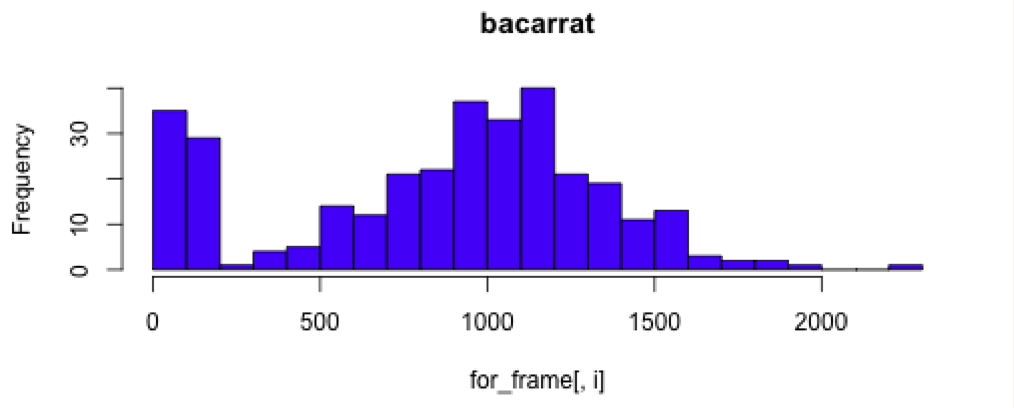
```r
for_frame <- data.frame(casino_clean[,2:5]) %>% # we subset our data to include only four numeri
    filter_all(all_vars(. > 1000)) # filter for only spend items above $1,000.

for (i in colnames(for_frame)) { # note the syntax...for (i in parentheses){ open brace...
  hist(for_frame[, i], # iteration occurs here...
  breaks = 20, # ...number of bars in each histogram...
  main = i, # ...and here too...
  col = randomColor()) # add random colors for fun just to prove R is iterating
} # closed brace
```

# For Loop Output









## For Loop Problem #1

Create the list `x_list` and then create a for loop to add 1 to each list value and print them.

```
x_list <- list(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
```

## For Loop Problem #1

Create the list `x_list` and then create a for loop to add 1 to each list value and print them.

```
x_list <- list(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
```

```
for (i in x_list){
  print(i-1)

}
```

```
## [1] 1
## [1] 3
## [1] 5
## [1] 7
## [1] 9
## [1] 11
## [1] 13
## [1] 15
## [1] 17
## [1] 19
```

## Remember That Lists Have Indices in R

```r
x_list[2] # single square bracket accesses list item #2
```

```
## [[1]]
## [1] 4
```

```r
x_list[[2]] # double square bracket
```

```
## [1] 4
```

## For Loop Problem #2

Using the same `x_list` object, now add 1 to each element and store the output in a new list, `y_list`.

- Hint 1: Before the for loop, create the empty `y_list` with this command: `<- list()`.
- Hint 2: A list has two layers, the layer number of name and then the item within that section of the list, so iterate over `i in 1:length(x_list)`
- Hint 3: Access the right values to modify by iterating over `xlist[[i]]`, not just `x_list` or `x_list[i]`.
- Hint 4: I made this stupid difficult on purpose to show how some of the seemingly simpler programming elements need to be mastered.

## Solution

```r
y_list <- list()

for (i in 1:length(x_list)){ # iterate once for ever value in the list index
  y_list[i] <- x_list[[i]]+1 # take the list value (not the index numer) and add 1
  # drop these new values into the i-th index in empty list y_list
}

y_list[1:3]
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 5
##
## [[3]]
## [1] 7
```

## Your First Function

A function is a chunk of code that combines multiple operators and functions into a repeatable task list.

## Your First Function

A function is a chunk of code that combines multiple operators and functions into a repeatable task list.

```r
# Normalize the dataset between values 0 and 1
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
```

# Your First Function

A function is a chunk of code that combines multiple operators and functions into a repeatable task list.

```r
# Normalize the dataset between values 0 and 1
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
```

```r
non_norm <- c(2, 22, 31, 6, 34, 67, 12, 12.5)
normalize(non_norm)
```

```
## [1] 0.00000000 0.30769231 0.44615385 0.06153846 0.49230769 1.00000000
## [7] 0.15384615 0.16153846
```

## D is for Dimensionality Reduction

## Dimensionality

Often the goal with broad data sets is to reduce the number of features down to a group that provide the most information gain for a machine-learning model. Examples include principle component analysis, vector auto regression, k nearest neighbor, and k-means.

We will take a closer look at k-means clustering in our meeting tonight:

> k-means clustering aims to partition `n` observations into `k` clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. ~ Wikipedia.

# K Means Clustering

We start by normalizing the data by total spend to avoid arbitrary units of variables. R has a function that performs a similar function called `scale`. Check it out by typing `?scale` into your console.

```
km_data <- casino_clean %>%
  mutate(slots_norm = slots/total_spend,
         black_jack_norm = black_jack/total_spend,
         craps_norm = craps/total_spend,
         bacarrat_norm = bacarrat / total_spend,
         bingo_norm = bingo/total_spend,
         poker_norm = poker / total_spend,
         other_norm = other / total_spend
         ) %>%
  select(contains("norm"), -total_spend) %>%
  na.omit
```

## K-Means 2

K-means clustering organizes unlabeled data into similar groups called clusters. It is a kind of unsupervised learning because there is no dependent (target) variable. K-means is a type of centroid clustering. For more information review this MIT presentation:
http://www.mit.edu/~9.54/fall14/slides/Class13.pdf

## What types of questions can cluster analysis answer?

1. Do our best/worst customers have anything in common with one another?
2. Which products should we recommend to our customers based on their existing preferences?
3. Are there any implicit patterns in my data to help me get started?
4. Which variables in my model are related?

# K-Means 3

We need three things to cluster data:

1. A proximity measure. In K-means, this is Euclidean distance. This is the "ordinary" straight-line distance between two points.
2. A criterion function. In K-means, this is the mean of the cluster.
3. An algorithm to optimize the criterion. In K-means, `k` random data points are selected (hence the use of a `seed` function for reproducibility) to be the centroids. Each new data point is allocated to the centroid whose mean has the least squared Euclidean distance. (i.e. the closest 😀).

## K-Means 3

The analyst must allocate k in the analysis. We'll start with five clusters.

```r
# Enhanced k-means clustering
km_analysis <- kmeans(km_data, centers = 5, nstart = 25)
fviz_cluster(km_analysis, data = km_data) +
  labs(title = "Casino Player Clusters")
```

# K-Means 4

We can see from the cluster chart on the previous page that our k=5 selection has provided five relatively isolated clusters of casino players! What does that mean? Let's take a look at what constitutes each cluster:

```
km_analysis$centers# this gives us the info on each centroid
```

```
##   slots_norm black_jack_norm craps_norm bacarrat_norm   bingo_norm
## 1 0.34520332       0.1506149  0.1018350    0.04998684 0.0009588283
## 2 0.00000000       0.0000000  0.0000000    0.00000000 0.0000000000
## 3 0.00000000       0.0000000  0.0000000    0.00000000 0.0000000000
## 4 0.93431631       0.0000000  0.0000000    0.00000000 0.0656760712
## 5 0.04751628       0.4382551  0.3693229    0.14490553 0.0000000000
##   poker_norm other_norm
## 1  0.1005338  0.2508698
## 2  1.0000000  0.0000000
## 3  0.0000000  1.0000000
## 4  0.0000000  0.0000000
## 5  0.0000000  0.0000000
```

## K-Means 5

The clusters are numbered 1 to 5 to the left of the table. The variable means for each cluster are provided. So, cluster 3 contains casino players who favor only "other" games. That's a very focused group of players. Contrast cluster 3 with cluster 1. Players in cluster 1 seem to play a little of everything. Maybe we could refer to these as our casino 'omnivores'.

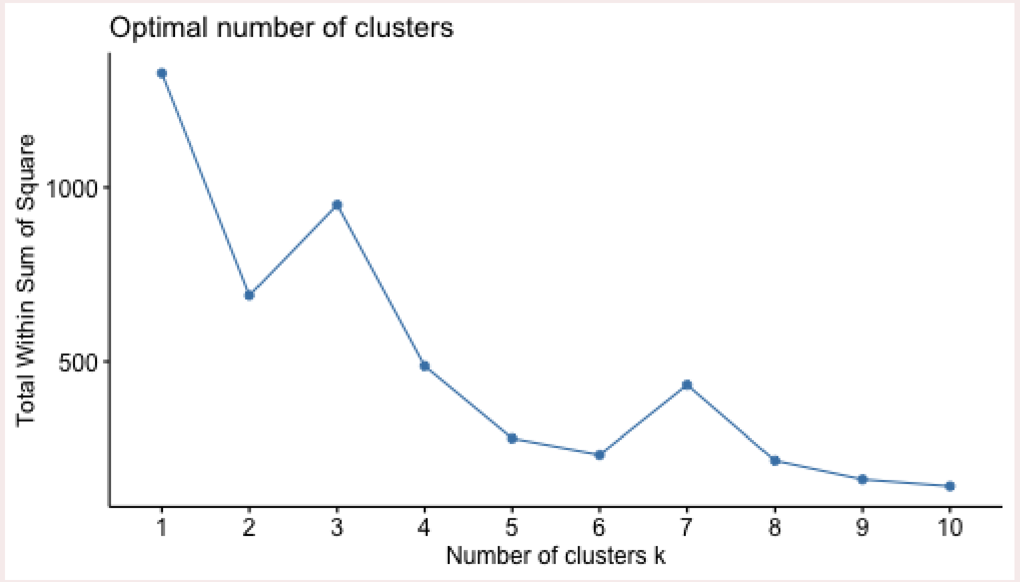
**But is this the optimal number of clusters?** Remember when we ran the algorithm, *we chose* `k=5` arbitrarily. The answer to this question has two elements: an objective, mathematical element; and a subjective business element.

The **math answer** is that the optimal number of clusters will minimize in-cluster sum of squares. The code on the next slide calculates that number for each number of clusters up to a maximum of ten. More than ten clusters would seem redundant given that we only have seven variables, and that inter-variable combination wouldn't intuitively provide us with any more information than, say, seven clusters would.

The **non-math answer** is "Well, what are you going to do with the cluster information". Let's say that as casino owners we want to treat different groups of players differently. Perhaps we want to determine whether we should replace the bingo game with higher-profit poker games? Or perhaps we want to know which groups to provide with free drinks? How many clusters do the answers to those questions require? As a casino floor manager we might find it particularly onerous (and expensive) to deal with more than five different types of customer.

## K-Means 6

```r
set.seed(1968)
fviz_nbclust(km_data, kmeans, method = "wss")
```



## K-Means Conclusion

Based on the chart on the previous slide, if we want to minimize the sum of squares across clusters, we might pick 6 or 8 clusters. But 5 clusters - our original number - is close to 6 in terms of the sum of squares performance metric. And 5 appears to be a manageable number both in terms of allocating a reasonable number of customer "types", and in yielding useful information. For example, we might not want to remove the bingo space because bingo players also play a lot of slots games. And slots games are profitable. They have minimal overhead costs and pay out infrequently.

Using a simple, unsupervised machine-learning approach and some common sense, we are able to optimize profit in our casino.