Lecture 11: Exploratory Data Analysis - Part 1

STAT 385 - James Balamuta

July 12, 2016

On the Agenda

1. Administrative Issues
   - HW4 Assigned Tonight
   - Graded Exams available for pick up tomorrow during Office Hour
2. Exploratory Data Analysis
   - Quantitative
   - Intro to Visual

Starting an Analysis

> *"When one begins an analysis, the facts of the analysis will stick to oneself."*
> *— James Balamuta*

What does it mean that the "facts" are **sticking** to them?

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a philosophy for data analysis that employs a variety of techniques that are primarily visual but sometimes quantitative pioneered by John Tukey in his 1977 book Exploratory Data Analysis

The goals are to:

- understand the structure of the data;
- detect mistakes in importing data or within the dataset;
- find outliers and anomalies; and
- test underlying assumptions;

Variable Types in Statistics

Unlike in *Base R*, Statistics views data stored in variables in two forms:

- Quantitative
  - A *number* that describes an outcome
    - **Discrete**: Integers e.g. 1 brother, 2 Starbucks Drinks
    - **Continuous**: Real number e.g. **86.25** on a test, $\pi = 3.141593\ldots$
- Categorical
  - A *string* that describes a trait
    - e.g. “Male” or “Female”, “Student” or “Instructor”, “Ninjas” or “Pirates”

Sample data

To investigate this, we're going to simulate some data that might be commonly associated with an experiment

```
# Make some data
n = 20

# Set seed for reproducibility
set.seed(1133)
d = data.frame(id = paste0("s",sample(1:n, n)),
               sex = sample(c("male","female"),
                            n, replace = T),
               food = sample(c("cake","pie"),
                             n, replace = T),
               trt_a = runif(n),
               trt_b = rnorm(n)
               )
```

Verify the Data

The first step to this process is to verify the `data`.
To do so, use:

- `head()` and `tail()`
  - to make sure the data has been imported correctly.
- `nrow()` and `ncol()` OR `dim()`
  - to understand the amount of observations and variables.
- `class`
  - to verify import data type of each variable.
- `is.na`
  - to obtain whether missing values exist.

Verify the Data - Head

```r
head(d)         # Defaults to showing the first 6
```

```
##    id    sex food      trt_a      trt_b
## 1 s19 female  pie 0.07404679  1.8732555
## 2  s6 female cake 0.57559848 -0.4506255
## 3 s16   male  pie 0.15270363 -1.0601917
## 4 s18 female  pie 0.97374584 -0.4109764
## 5  s5 female cake 0.95795011 -0.6313457
## 6  s1   male  pie 0.65279380 -0.6462618
```

```r
head(d, n = 2) # Shows the first 2
```

```
##    id    sex food      trt_a      trt_b
## 1 s19 female  pie 0.07404679  1.8732555
## 2  s6 female cake 0.57559848 -0.4506255
```

Verify the Data - Tail

```r
tail(d)          # Defaults to showing the last 6
```

```
##      id    sex food      trt_a      trt_b
## 15 s17 female  pie 0.32042847 -1.4592700
## 16 s10 female  pie 0.41355622 -0.3774795
## 17 s14 female cake 0.69593115  0.1023129
## 18 s15 female  pie 0.05747749 -0.2826929
## 19  s8   male  pie 0.31232103 -0.4166077
## 20 s20 female cake 0.90455963  0.1255623
```

```r
tail(d, n = 2) # Shows the last 2
```

```
##      id    sex food     trt_a      trt_b
## 19  s8   male  pie 0.3123210 -0.4166077
## 20 s20 female cake 0.9045596  0.1255623
```

Verify the Data - Observations and Variables

```r
nrow(d)  # Find the number of observations
```

```
## [1] 20
```

```r
ncol(d)  # Find the number of variables
```

```
## [1] 5
```

```r
dim(d)   # Both observations and variables (n x p)
```

```
## [1] 20  5
```

Verify the Data - Check Data Types

```
sapply(d, FUN=class)   # Obtain each columns data type
```

```
##        id       sex      food     trt_a     trt_b
##  "factor"  "factor"  "factor" "numeric" "numeric"
```

Verify the Data - Missing Values

```r
# Count number of missing values
sum_na = function(x){
  sum(is.na(x))
}
```

```r
sapply(d, FUN=sum_na)   # Missing values per column
```

```
##     id    sex   food trt_a trt_b
##      0      0      0     0     0
```

Types of EDA

There are two types of EDA:

- Quantitative
- Visual

Both with *ups* and *downs*.

Univariate **Quantitative** Analysis

Depending on the *data type* there are different ways of obtaining univariate **quantitative** information

- **numeric**
  - 5 Summary
- **categorical**
  - frequency
  - contigency table

Univariate **Quantitative** Analysis - Numeric

The 5 Summary Statistics are defined as follows:

- **Minimum**
  - `min()`
- **1st Quartile** or **25% Quantile**
  - `quantile(x, probs = 0.25)`
- **2nd Quartile** or **50% Quantile**
  - `median()`
- **3rd Quartile** or **75% Quantile:**
  - `quantile(x, probs = 0.75)`
- **Maximum:**
  - `max()`
- **(*Optional*) Mean:**
  - `mean()`

Univariate **Quantitative** Analysis - Numeric

Quick implementation

```
stat5summary = function(x, na.rm = T){
  if(class(x) != "numeric")
    stop("`x` must be numeric data")

  # Calculate quantiles
  q = quantile(x, probs = c(0.25, 0.5, 0.75),
               na.rm = na.rm)

  # Return
  c("min" = min(x, na.rm = na.rm),
    "q1" = q[[1]], "median" = q[[2]], "q3" = q[[3]],
    "max" = max(x, na.rm = na.rm))
}
```

What might we want to add here?

Univariate **Quantitative** Analysis - Numeric

Let's try out our function!

```
sapply(d[,4:5], FUN = stat5summary)
```

```
##             trt_a      trt_b
## min    0.05747749 -1.4749439
## q1     0.37380872 -0.9103675
## median 0.59020143 -0.4336166
## q3     0.75354446 -0.2113880
## max    0.97374584  1.8732555
```

Univariate **Quantitative** Analysis - Numeric

Psst... the `summary()` function does this by default on `numeric` data!

```
sapply(d[,4:5], FUN = summary)
```

```
##            trt_a   trt_b
## Min.    0.05748 -1.4750
## 1st Qu. 0.37380 -0.9104
## Median  0.59020 -0.4336
## Mean    0.55020 -0.4259
## 3rd Qu. 0.75350 -0.2114
## Max.    0.97370  1.8730
```

Univariate **Quantitative** Analysis - Categorical

Categorical data normally is associated with:

- Frequency Counts
- Table Format x vs. y
- Percentages

Univariate **Quantitative** Analysis - Categorical

```
sapply(d[,1:3], FUN = summary)
```

```
## $id
##  s1 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19  s2 s20  s3
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  s5  s6  s7  s8  s9
##   1   1   1   1   1
##
## $sex
## female   male
##     12      8
##
## $food
## cake  pie
##    9   11
```

Univariate **Quantitative** Analysis - Categorical Tabulate

Overall counts between two variables

```
(o = table(d[,2], d[,3]))
```

```
##
##           cake pie
##   female     6   6
##   male       3   5
```

Univariate **Quantitative** Analysis - Categorical Proportions

Element / Total number of observations

```r
prop.table(o)  # Requires table() object
```

```
##
##          cake  pie
##   female 0.30 0.30
##   male   0.15 0.25
```

Univariate **Quantitative** Analysis - Categorical Headache

Make sure to avoid unique comparisons. . .

```r
head(table(d[,1], d[,2]))
```

```
##
##        female male
##   s1        0    1
##   s10       1    0
##   s11       0    1
##   s12       0    1
##   s13       1    0
##   s14       1    0
```

Rules of Thumb

There are a couple *rules of thumb* that are slightly helpful with EDA and statistical modeling.

1. If the number of distinct numbers is less than 20, treat them as *categorical* variables.
2. Try to floor and cap *numerical* values to avoid large extrema.
   - Floor and Cap means to set a boundary point for low and high values.
   - Never tell a robust statistician this...

Exercises

1. Determine the summary information for the `PlantGrowth` dataset.
   - What variables exist, what kind of variables are there?
2. Obtain the `msos` package from cran and look at the `spam` dataset.
   - How often was spam detected?
3. Download the `faraway` package from CRAN and explore the `pima` dataset.
   - Any pattern with missing values?

Univariate **Visual** Analysis

"The greatest value of a picture is when it forces us to notice what we never expected to see."
— John Tukey in Exploratory Data Analysis (1977)

A sample data generation

```
set.seed(2016) # Set Seed for reproducibility
n = 1e4        # Number of observations

(n*2) %>%      # Generate some data
  rnorm %>%
  matrix(ncol = 2) -> a

runif(n, 0, 2 * pi) %>%
  {0.5 * cbind(sin(.), cos(.))} -> b

o = rbind(a,b)   # Combine generate data

x = as.data.frame(o[sample(nrow(o)), ])

colnames(x) = c("x","y")
```

Do you know what is happening?

Numerically we have...

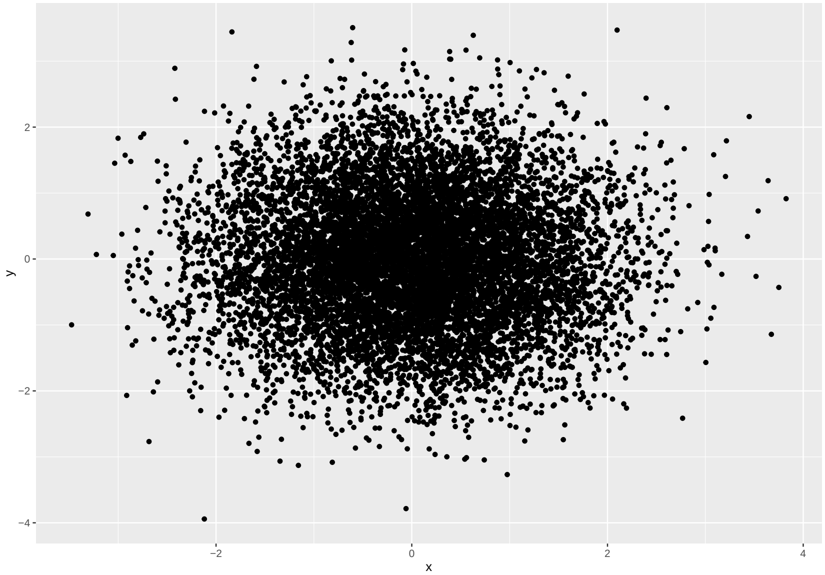
```r
summary(x)    # data.frame implements summary.
```

```
##        x                   y
##  Min.   :-3.476111   Min.   :-3.94248
##  1st Qu.:-0.436574   1st Qu.:-0.43595
##  Median : 0.005087   Median : 0.00629
##  Mean   :-0.000790   Mean   : 0.00137
##  3rd Qu.: 0.430879   3rd Qu.: 0.43412
##  Max.   : 3.825197   Max.   : 3.50801
```

Insight: Data looks to be bounded between -4 and 4.

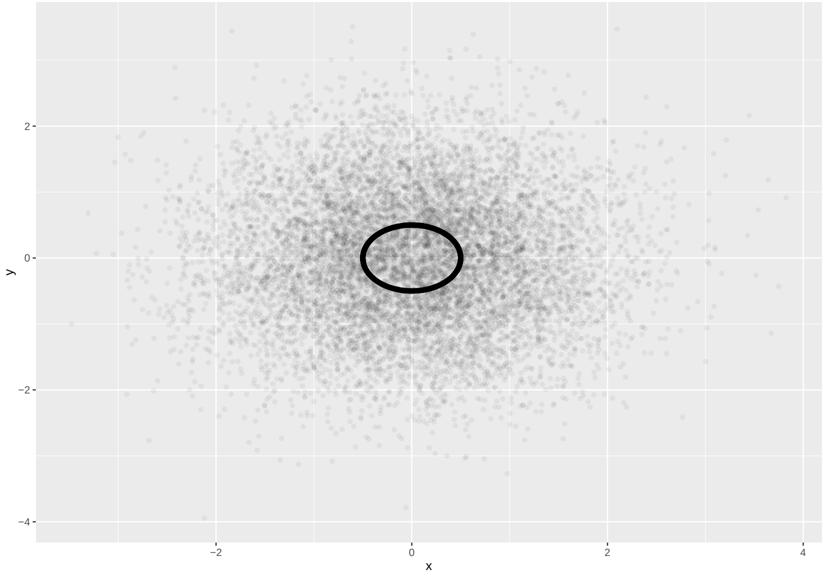
Graphically we have ...

```
ggplot(x) + geom_point(aes(x,y))
```



Redux of Graphically we have . . .

```
ggplot(x) + geom_point(aes(x,y), alpha = 0.05)
```



A note...

- Notice in the previous slides, there was no call to `plot()`.
- Instead, `ggplot()` was used to create the graphic through the use of layering via the + symbol.
- To do the same with base *R*, we would of used:

```
plot(x, col = rgb(0, 0, 0, 0.05)) # Transparent color
```