Primary information sources:
- lecture slides
- text book
- tutorial content.

STA130 - Class #2: – 10:00 AM Class

Nathan Taback
2018-01-15

Today's Class

- Histograms and density functions
- Statistical data
- Tidy data
- Data wrangling
- Transforming data

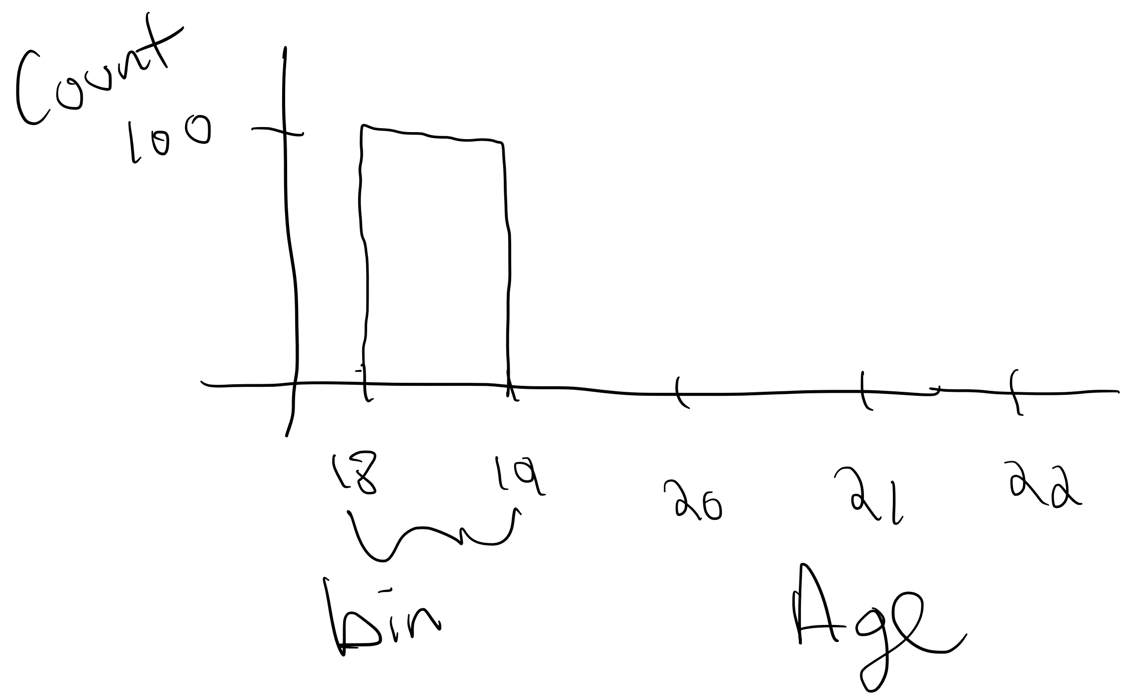
R library
dplyr.

Histograms and Density Functions

Histograms and Density Functions

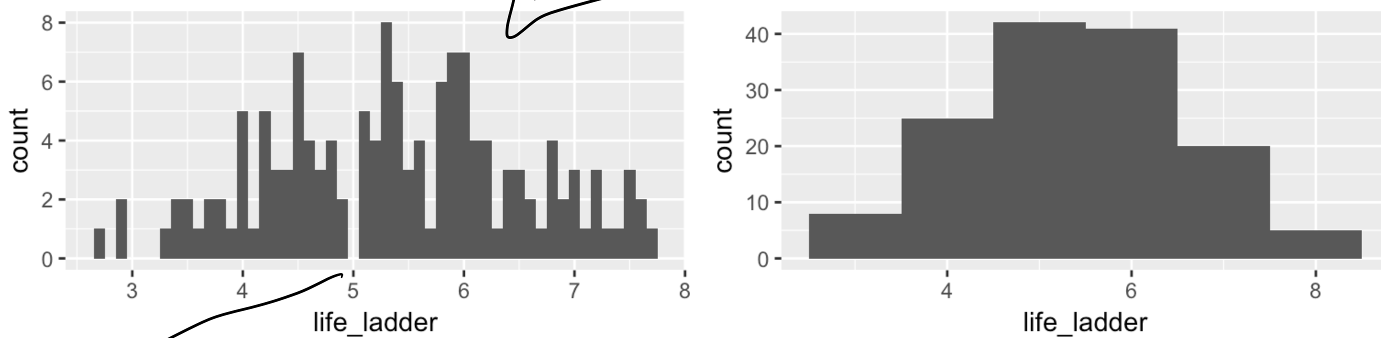
- The histogram of a variable is a graphical method to vizualize the distribution of a single variable.
- To construct a basic histogram:

1. Divide the data into intervals (called bins). x- axis.
2. Count the number of observations that are contained in the bin.
3. Plot rectangles with height equal to the count from (2) and width equal to the width of the bin.



Histograms and Density Functions

- Different bin width will yield different histograms



from happiness data from last Class. Which histogram has Smaller bin width? histogram on left.

Count = 0

Mathematical Definition of Histogram

- The bins of the histogram are the intervals:

$$[x_0 + mh, x_0 + (m+1)h).$$

$x \in [1, 2)$

$1 \le x < 2$

x_0 is the origin, $m = \dots, -1, 0, 1, \dots$ indexes the bins, and $h = (x_0 + (m+1)h) - (x_0 + mh)$ is the bin width.

Right endpoint

left endpoint.

Example - Mathematical Definition of Histogram

```
dat <- data_frame(x = c(1,2,2.5,3,7))
dat$x
```

x1 x2 x3 x4 x5 — defines a data set in R with 1 variable named x.

Max value.

```
[1] 1.0 2.0 2.5 3.0 7.0
```

Let $x_0 = 0.5, h = 0.25, m = 1, \ldots, 29$

```
seq(0.5,7.5,by = 0.25)
```

```
 [1] 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75
[15] 4.00 4.25 4.50 4.75 5.00 5.25 5.50 5.75 6.00 6.25 6.50 6.75 7.00 7.25
[29] 7.50
```

0.25

The bins are: $[0.50, 0.75), [0.75, 1.00), [1.00, 1.25), \ldots, [7.25, 7.50)$.

$0.75 - 0.50 = 0.25$

One observation in this bin

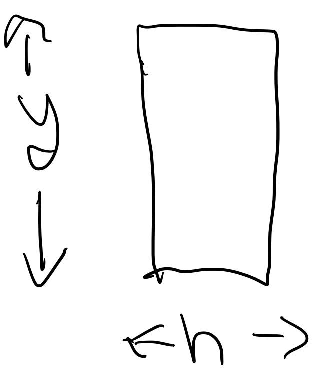
Example - Mathematical Definition of Histogram

- The bins can be used to construct rectangles with width $h = 0.25$ and height y.
- y will be called density.
- The area of these rectangles is hy. ✓
- We would like the area of these rectangles, hy, to be the same as the proportion of data in the bin. This will make the sum of all areas equal 1.
- Let n be the number of observations. Then,

$$(*) \quad hy = \left(\frac{\#\{X_i \text{ in bin}\}}{n} \right)$$

← proportion of data in a bin

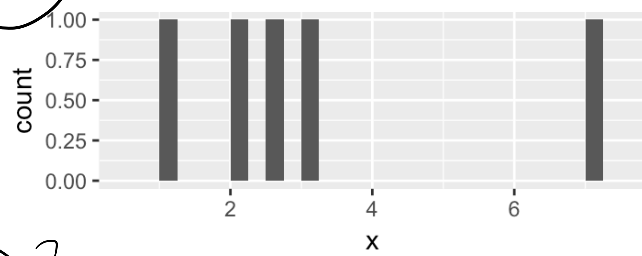
- In this example, $n = 5$, and $X_1 = 1, X_2 = 2, X_3 = 2.5, X_4 = 3, X_5 = 7$.



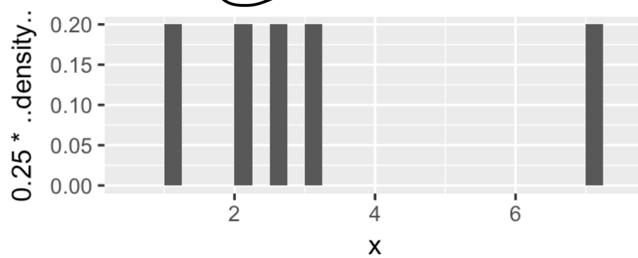
Area = $h \cdot y$

Example - Mathematical Definition of Histogram

(A)



(C)



C: Relative frequency

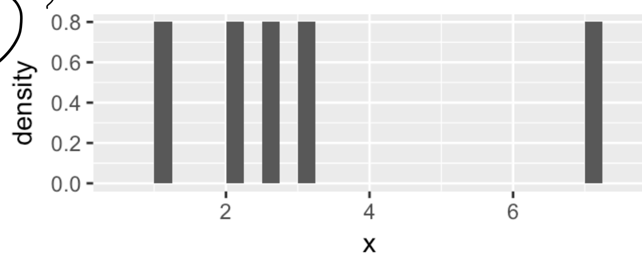
$= \frac{\# x_i \text{ in bin}}{n}$

$= 1/5 = 0.20$

Area of rect

0.25 x 0.20

(B)



- Three histograms with different values on y-axis.

A: Count

from (*)

B: density: $\text{density} = \frac{\# \text{obs. in a bin}}{n \cdot h} = \frac{1}{5 \cdot (0.25)} = 0.8$

↳ Areas of rectangles add to 1.

Mathematical Definition of Histogram

$$\hat{f}(x) = \frac{1}{hn} \#\{X_i \text{ in same bin as } x\}$$

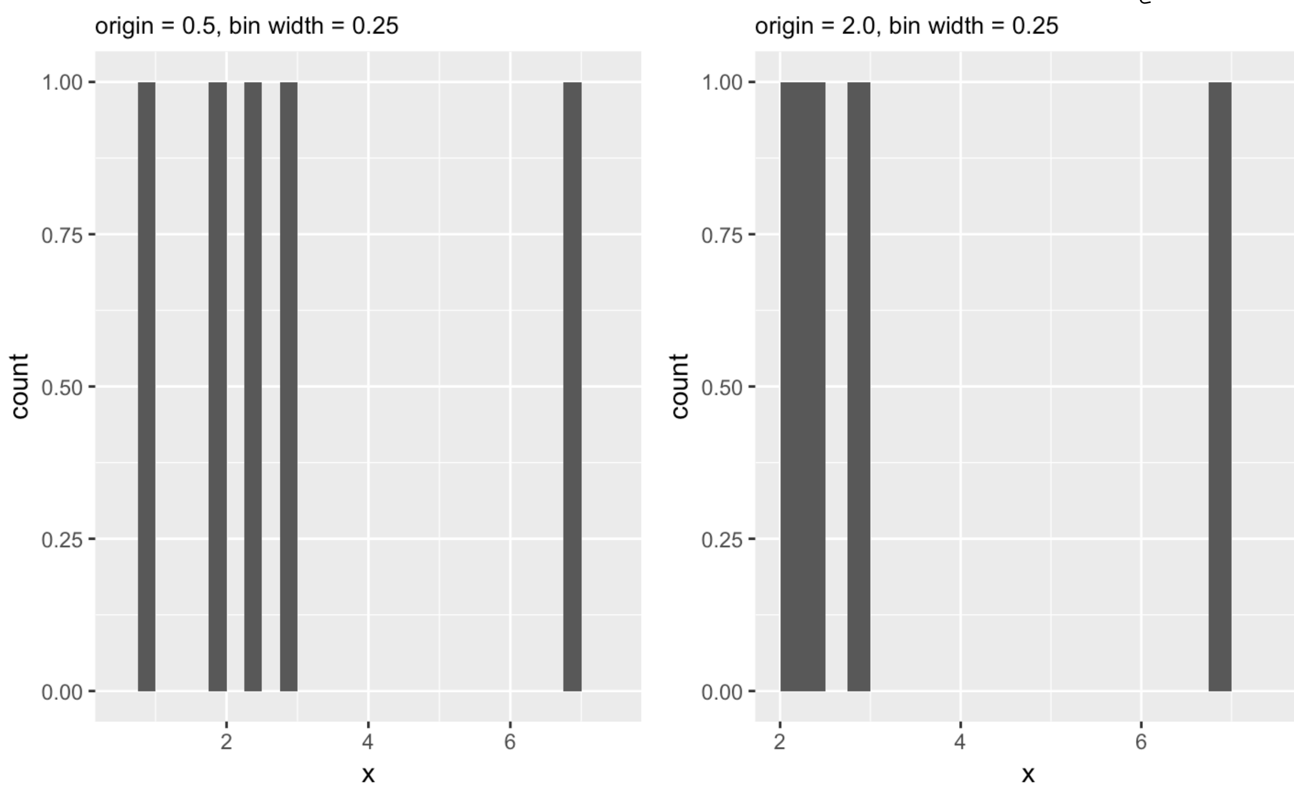
is called the **histogram estimator**.

$\hat{f}(x)$ is an estimate of the density at a point x.

To construct the histogram we have to choose an origin x_0 and bin width h.

Choosing Origin and Bin Width in R

Same bin width but different origin



← this histogram doesn't capture all the data ∵ the origin is too large relative to smallest observation. (i.e., x = 1)

1, 2, 2.5, 3, 7 : Data

Statistical data

What is statistical data?

- Statistical data is obtained by observing (random) variables.
- A random variable can be given a precise mathematical definition that we will cover later in the course.
- In this class we will discuss examples.

Observing a few variables on STA130 students

- What is your height?
- How many years have been at UofT?
- What is your sex (male or female)?

Collecting this data will generate three variables: `height`, `years`, and `sex`.

Enter variables on STA130 students

```
height <- c()
years <- c()
sex <- c()
```

Put the variables into an R data frame.

NB: `data_frame` is the `tidyverse` version of base R `data.frame`.

```
sta130_dat <- data_frame(height, years, sex)
```

We could have entred this in a spreadsheet program like MS Excel, saved it as a CSV file, then imported the file into R.

Tidy data

Tidy data

There are three interrelated rules which make a dataset tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Not tidy

① Since each variable does not have it's own column.

② Each value does not have it's own cell.

Cell

Cell

height	Years and sex
1.78	1, M
1.63	1, F
1.75	1, M
⋮	⋮

Cell

f(x, y) = x / y. to get rate Cases / population

Tidy data

Which data set is tidy?

Cases of TB — population size of Afg. in 1999

```
## # A tibble: 6 x 4
##       country  year  cases population
##         <chr> <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3      Brazil  1999  37737  172006362
## 4      Brazil  2000  80488  174504898
## 5       China  1999 212258 1272915272
## 6       China  2000 213766 1280428583
```

(A) is tidy ∵ it follow the three rules

```
## # A tibble: 6 x 3
##       country  year              rate
## *       <chr> <int>             <chr>
## 1 Afghanistan  1999      745/19987071
## 2 Afghanistan  2000     2666/20595360
## 3      Brazil  1999    37737/172006362
## 4      Brazil  2000    80488/174504898
## 5       China  1999 212258/1272915272
## 6       China  2000 213766/1280428583
```

(B) is not tidy ∵ each value does not have it's own cell.

→ Compare rates between Afg, Brazil, China then compare groups of rows.

Tidy data

"For a given dataset, it is usually easy to figure out what are observations and what are variables, but it is surprisingly difficult to precisely define variables and observations in general." (Wickham, 2014)

A general rule of thumb:

- It is easier to describe functional relationships between variables (e.g., z is a linear combination of x and y, density is the ratio of weight to volume) than between rows.
- It is easier to make comparisons between groups of observations (e.g., average of group a vs. average of group b) than between groups of columns.

(Wickham, 2014)

Data wrangling

Data wrangling

- The `ggplot` library implements a **grammer of graphics**.
- Similarily the `dplyr` library presents a grammer for data wrangling.

The Economic Guide to Picking a Major

FiveThirtyEight

Politics Sports Science & Health **Economics** Culture

SEP. 12, 2014 AT 7:37 AM

The Economic Guide To Picking A College Major

By Ben Casselman

Filed under Higher Education

Get the data on GitHub

Students walk across the campus of UCLA in Los Angeles. KEVORK DJANSEZIAN / GETTY IMAGES

"…A college degree is no guarantee of economic success. But through their choice of major, they can take at least some steps toward boosting their odds."

The Economic Guide to Picking a Major

- The data used in the article is from the American Community Survey 2010-2012 Public Use Microdata Series.
- We can use the `fivethirtyeight` library in R.

Data behind the article

```
library(fivethirtyeight) # load the library
glimpse(college_recent_grads)
```

```
## Observations: 173
## Variables: 21
## $ rank                        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,...
## $ major_code                  <int> 2419, 2416, 2415, 2417, 2405, 2418...
## $ major                       <chr> "Petroleum Engineering", "Mining A...
## $ major_category              <chr> "Engineering", "Engineering", "Eng...
## $ total                       <int> 2339, 756, 856, 1258, 32260, 2573,...
## $ sample_size                 <int> 36, 7, 3, 16, 289, 17, 51, 10, 102...
## $ men                         <int> 2057, 679, 725, 1123, 21239, 2200,...
## $ women                       <int> 282, 77, 131, 135, 11021, 373, 960...
## $ sharewomen                  <dbl> 0.1205643, 0.1018519, 0.1530374, 0...
## $ employed                    <int> 1976, 640, 648, 758, 25694, 1857, ...
## $ employed_fulltime           <int> 1849, 556, 558, 1069, 23170, 2038,...
## $ employed_parttime           <int> 270, 170, 133, 150, 5180, 264, 296...
## $ employed_fulltime_yearround <int> 1207, 388, 340, 692, 16697, 1449, ...
## $ unemployed                  <int> 37, 85, 16, 40, 1672, 400, 308, 33...
## $ unemployment_rate           <dbl> 0.018380527, 0.117241379, 0.024096...
## $ p25th                       <dbl> 95000, 55000, 50000, 43000, 50000,...
## $ median                      <dbl> 110000, 75000, 73000, 70000, 65000...
```

# of rows
# of Columns

Select variables/columns using `select()`

To retrieve a data frame with only major, number of male and female graduates we use the `select()` function in the `dplyr` library.

```
select(college_recent_grads,major, men,women)
```

data frame

Variables that I wish to select.

```
## # A tibble: 173 x 3
##                                     major   men women
##                                     <chr> <int> <int>
##  1                  Petroleum Engineering  2057   282
##  2         Mining And Mineral Engineering   679    77
##  3              Metallurgical Engineering   725   131
##  4 Naval Architecture And Marine Engineering  1123   135
##  5                   Chemical Engineering 21239 11021
##  6                    Nuclear Engineering  2200   373
##  7                      Actuarial Science   832   960
##  8             Astronomy And Astrophysics  2110  1667
##  9                 Mechanical Engineering 12953  2105
## 10                 Electrical Engineering  8407  6548
## # ... with 163 more rows
```

Select observations/rows using `filter()`

If we want to retrieve only those observations (rows) that pertain to engineering majors then we need to specify that the value of the `major` variable is Electrical Engineering.

```
EE <- filter(college_recent_grads, major == "Electrical Engineering")
glimpse(EE)
```

```
## Observations: 1
## Variables: 21
## $ rank                         <int> 10
## $ major_code                   <int> 2408
## $ major                        <chr> "Electrical Engineering"
## $ major_category               <chr> "Engineering"
## $ total                        <int> 81527
## $ sample_size                  <int> 631
## $ men                          <int> 8407
## $ women                        <int> 6548
## $ sharewomen                   <dbl> 0.4378469
## $ employed                     <int> 61928
## $ employed_fulltime            <int> 55450
## $ employed_parttime            <int> 12695
## $ employed_fulltime_yearround <int> 41413
```

Combine `select()` and `filter()`

- We can drill down to get certain pieces of information using `filter()` and `select()` together.
- The `median` variable is median salary.

```
select(filter(college_recent_grads, median >= 60000), major, men, women)
```

(1) Which students, and (2) variables are in this data frame?

Respond at **PollEv.com/nathantaback**

Text **NATHANTABACK** to **37607** once to join, then **A, B, C, D, or E**

(1) All students in the original data set; (2) all variables in the data set.	A
(1) All students in the original data set in a major where the median salary is at most than 60,000; (2) all variables in the data set.	B
(1) All students in the original data set in a major where the median salary is at least than 60,000; (2) all variables in the data set.	C
(1) All students in the original data set in a major where the median salary is at least than 60,000; (2) three variables: major, men, women	D
(1) All students in the original data set in a major where the median salary is at least than 60,000; (2) all variables in the data set.	E

The pipe operator `%>%`

In the code:

```
select(filter(college_recent_grads, median >= 60000), major,men,women)
```

filter is nested inside select.

The pipe operator allows is an alternative to nesting and yields easier to read code. The same expression can be written with the pipe operator

```
college_recent_grads %>%
  filter(median >= 60000) %>%
  select(major, men, women)
```

Create new variables from existing variables using `mutate()`

What percentage of graduates from each major where the median earnings is at least $60,000 are men ?

how many variables in this data frame? 5 variables

```
college_recent_grads %>%
  filter(median >= 60000) %>%
  select(major, men, women) %>%
  mutate(total = men + women, pct_male = round((men / total)*100, 2))
```

(1) major, (2) men, (3) women, (4) total, (5) pct_male

Compare to nested code:

```
mutate(select(filter(college_recent_grads,median >= 60000),
              major, men, women),
       total = men + women, pct_male = round((men / total)*100, 2))
```

total and pct_male are new variables.

Create new variables from existing variables using `mutate()`

major	men	women	total	pct_male
Petroleum Engineering	2057	282	2339	87.94
Mining And Mineral Engineering	679	77	756	89.81
Metallurgical Engineering	725	131	856	84.70
Naval Architecture And Marine Engineering	1123	135	1258	89.27
Chemical Engineering	21239	11021	32260	65.84
Nuclear Engineering	2200	373	2573	85.50
Actuarial Science	832	960	1792	46.43
Astronomy And Astrophysics	2110	1667	3777	55.86
Mechanical Engineering	12953	2105	15058	86.02

in Console type ?ifelse.

Create new variables from existing variables using `mutate()`

ifelse(Test, Value1, Value2)

- Suppose that we would like to create a categorical variable to identify majors with 45% and 55% women (ie., approximately equal numbers of males and females).
- We can use `ifelse()` in a `mutate()` statement.

Not valid R code.
pseudo-code

```
college_recent_grads %>%
  select(major, men, women) %>%
  mutate(total = men + women, pct_female = round((women / total)*100, 2),
         male.bias = ifelse(pct_female >= 45 & pct_female <= 55, "No","Yes")) %>%
  select(major,male.bias)
```

If pct_fem ≥ 45 and
pct_female ≤ 55
then
male.bias = "No"
otherwise
male.bias = "Yes"

```
## # A tibble: 173 x 2
##                                    major male.bias
##                                    <chr>     <chr>
##  1                 Petroleum Engineering       Yes
##  2        Mining And Mineral Engineering       Yes
##  3             Metallurgical Engineering       Yes
##  4 Naval Architecture And Marine Engineering       Yes
##  5                  Chemical Engineering       Yes
```

Rename variables using `rename()`

- It's considered bad practice in R to use periods in variable names.
- We can use `rename()` to change the name of `sex.equal` to `sex_equal`.

```
my_college_dat <- college_recent_grads %>%
  select(major, men, women, median) %>%
  mutate(total = men + women, pct_female = round((women / total)*100, 2),
         sex.equal = ifelse(pct_female >= 45 & pct_female <= 55, "No","Yes")) %>%
  select(major,sex.equal, median)

my_college_dat <- my_college_dat %>%
  rename(sex_equal = sex.equal, salary_median = median)
glimpse(my_college_dat)
```

old variable name.

new variable name

```
## Observations: 173
## Variables: 3
## $ major         <chr> "Petroleum Engineering", "Mining And Mineral Eng...
## $ sex_equal     <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", ...
## $ salary_median <dbl> 110000, 75000, 73000, 70000, 65000, 65000, 62000...
```

Sort a data frame using **`arrange()`**

```
my_college_dat %>%
  arrange(salary_median) %>%
  select(major, salary_median) %>%
  arrange(desc(salary_median))
```

descending order.

```
## # A tibble: 173 x 2
##                                       major salary_median
##                                       <chr>         <dbl>
##  1                    Petroleum Engineering        110000
##  2           Mining And Mineral Engineering         75000
##  3                Metallurgical Engineering         73000
##  4 Naval Architecture And Marine Engineering        70000
##  5                     Chemical Engineering         65000
##  6                      Nuclear Engineering         65000
##  7                        Actuarial Science         62000
##  8             Astronomy And Astrophysics           62000
##  9                   Mechanical Engineering         60000
## 10                   Electrical Engineering         60000
## # ... with 163 more rows
```

Summarize a data frame using `summarize()`

The average number of female grads and the total number of majors in the data set.

```
college_recent_grads %>%
  select(major, men, women) %>%
  summarise(femgrad_mean = mean(women), N = n())
```

Number of observations in data frame.

```
## # A tibble: 1 x 2
##   femgrad_mean     N
##          <dbl> <int>
## 1     22530.36   173
```

mean number of female students in data.

Summarize groups in a data frame using `summarize()` and `group_by()`

The median salary in majors with 45%-55% female students.

```
my_college_dat %>%
  group_by(sex_equal) %>%
  summarise(median(salary_median))
```

Seperates data into groups

```
## # A tibble: 2 x 2
##   sex_equal `median(salary_median)`
##       <chr>                   <dbl>
## 1        No                   37400
## 2       Yes                   36000
```

Combining Multiple Tables

Sentiment of Trump's Tweets

- Donald Trump likes to tweet a lot.
- Some tweets have an angry sentiment or contain insults, and some are not.
- Trump supposedly used to send tweets from a Samsung Galaxy when he is insulting people, places, and things, from other devices such as an iPhone when he is not.
- Trump's last tweet from Android were March 25, 2017

Trump's Tweets

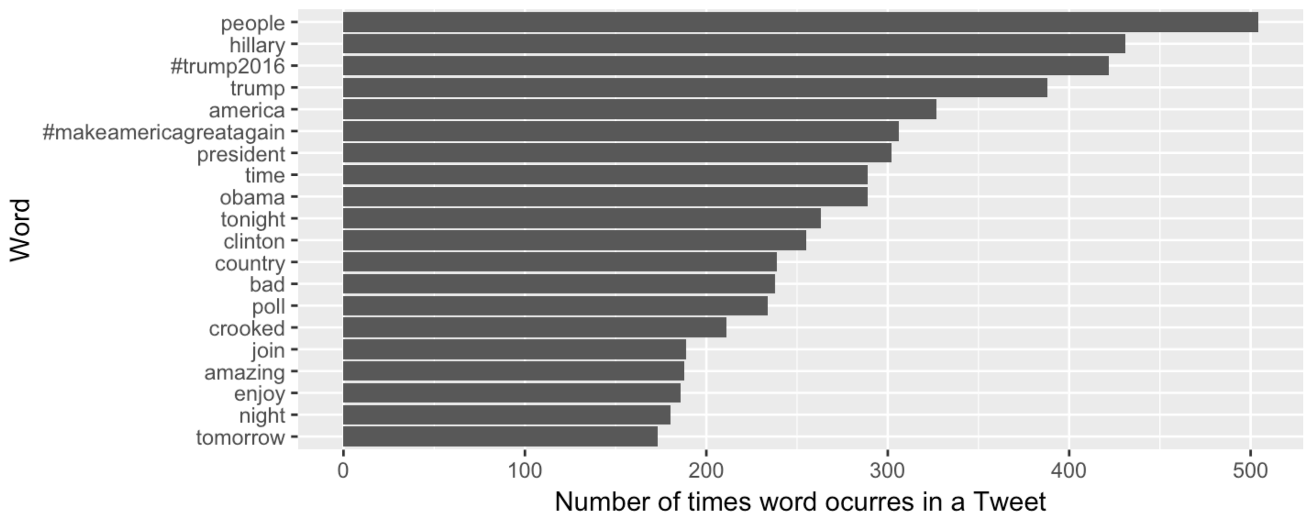
A data frame with Trump's Tweets.

```
trumptweets <- read_csv("trumptweets.csv") #import from csv file
glimpse(trumptweets)
```

```
## Observations: 53,333
## Variables: 4
## $ source     <chr> "Android", "Android", "Android", "Android", "Androi...
## $ created_at <dttm> 2013-02-06 01:53:40, 2013-02-06 01:53:40, 2013-02-...
## $ id_str     <dbl> 2.989727e+17, 2.989727e+17, 2.989727e+17, 2.989727e...
## $ word       <chr> "@sherrieshepherd", "nice", "comments", "view", "te...
```

Trump's tweets

```
trumptweets %>%
  count(word) %>%
  mutate(word = reorder(word,n)) %>%
  top_n(20) %>%
  ggplot(aes(word, n)) + geom_col() + coord_flip()  +
  labs(x = "Word",y = "Number of times word ocurres in a Tweet")
```



Sentiment Lexicon

- Several lexicons (dictionaries) have been developed that categorize words according to sentiment (feeling or emotion).
- The **`tidytext`** library contains several lexicons.

```
library(tidytext)
sentiments
```

```
## # A tibble: 27,314 x 4
##           word sentiment lexicon score
##          <chr>     <chr>   <chr> <int>
##  1      abacus     trust     nrc    NA
##  2     abandon      fear     nrc    NA
##  3     abandon  negative     nrc    NA
##  4     abandon   sadness     nrc    NA
##  5   abandoned     anger     nrc    NA
##  6   abandoned      fear     nrc    NA
##  7   abandoned  negative     nrc    NA
##  8   abandoned   sadness     nrc    NA
##  9 abandonment     anger     nrc    NA
## 10 abandonment      fear     nrc    NA
## # ... with 27,304 more rows
```

NRC Lexicon

- The nrc lexicon categorizes words in a binary fashion (“yes”/“no”) into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.
- The `getsentiments()` function provides a way to get specific sentiment lexicons without the columns that are not used in that lexicon.

NRC Lexicon

```
get_sentiments("nrc")
```

```
## # A tibble: 13,901 x 2
##           word sentiment
##          <chr>     <chr>
##  1      abacus     trust
##  2     abandon      fear
##  3     abandon  negative
##  4     abandon   sadness
##  5   abandoned     anger
##  6   abandoned      fear
##  7   abandoned  negative
##  8   abandoned   sadness
##  9 abandonment     anger
## 10 abandonment      fear
## # ... with 13,891 more rows
```

Sentiment of Words used in Tweets

- To examine the sentiment of the words Trump used in tweets we need to join the data frame containing the NRC lexicon and the data frame of Trump's words used in tweets.

- `inner_join(x,y)`: return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.

```
trumptweets %>% inner_join(get_sentiments("nrc"))
```

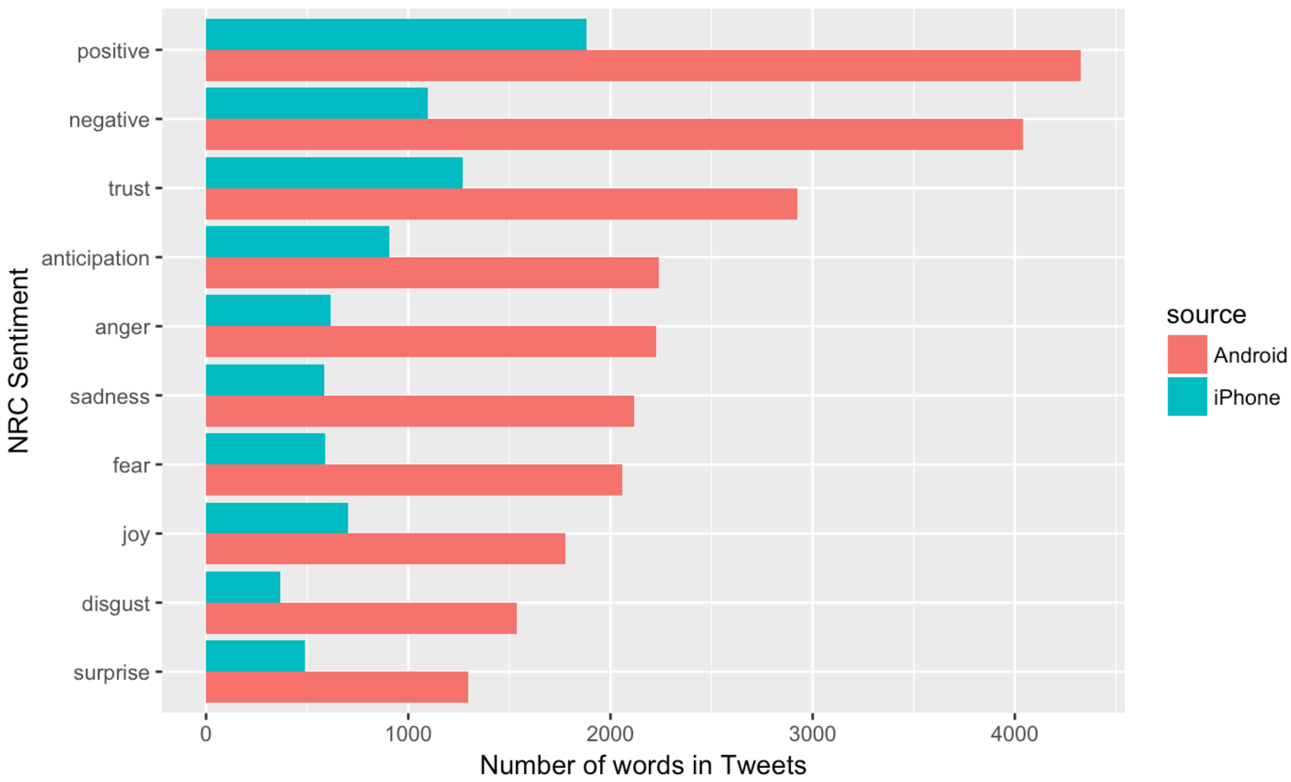
```
## # A tibble: 33,043 x 5
##     source          created_at       id_str      word sentiment
##      <chr>              <dttm>        <dbl>     <chr>     <chr>
##  1 Android 2013-02-06 01:53:40 2.989727e+17  terrific   sadness
##  2 Android 2013-02-18 23:36:36 3.036492e+17       sky  positive
##  3 Android 2013-02-18 23:36:36 3.036492e+17    rocket     anger
##  4 Android 2013-02-18 23:36:36 3.036492e+17   payback     anger
##  5 Android 2013-02-18 23:36:36 3.036492e+17   payback  negative
##  6 Android 2013-02-19 00:25:48 3.036616e+17 surprised  surprise
##  7 Android 2013-02-19 12:36:19 3.038455e+17      buss       joy
##  8 Android 2013-02-19 12:36:19 3.038455e+17      buss  positive
##  9 Android 2013-02-19 12:36:19 3.038455e+17    friend       joy
```

Sentiment of Words used in Tweets

```
trumptweets %>%
  inner_join(get_sentiments("nrc")) %>%
  group_by(sentiment,source) %>%
  summarise(n=n()) %>%
  mutate(pct= round(n/sum(n)*100,2)) %>%
  arrange(desc(pct))
```

```
## # A tibble: 20 x 4
## # Groups:   sentiment [10]
##       sentiment  source     n   pct
##           <chr>   <chr> <int> <dbl>
##  1      disgust Android  1537 80.68
##  2     negative Android  4040 78.68
##  3      sadness Android  2117 78.32
##  4        anger Android  2228 78.31
##  5         fear Android  2057 77.80
##  6     surprise Android  1297 72.70
##  7          joy Android  1777 71.65
##  8 anticipation Android  2240 71.25
##  9     positive Android  4328 69.72
## 10        trust Android  2924 69.70
## 11        trust  iPhone  1271 30.30
```

Sentiment of Words used in Tweets



Join two tables together

- In the `dplyr` library there are several other ways to join tables: `left_join()`, `right_join()`, `full_join()`, `semi_join()`, `anti_join()`.
- See `dplyr` documentation.

Transforming data

Statistical Transformations

- In statistical anaysis it's often necessary to transform data.
- Transforming data takes each value of a variable x_i and transforms it into $f(x_i)$:

$$x_i \mapsto f(x_i).$$

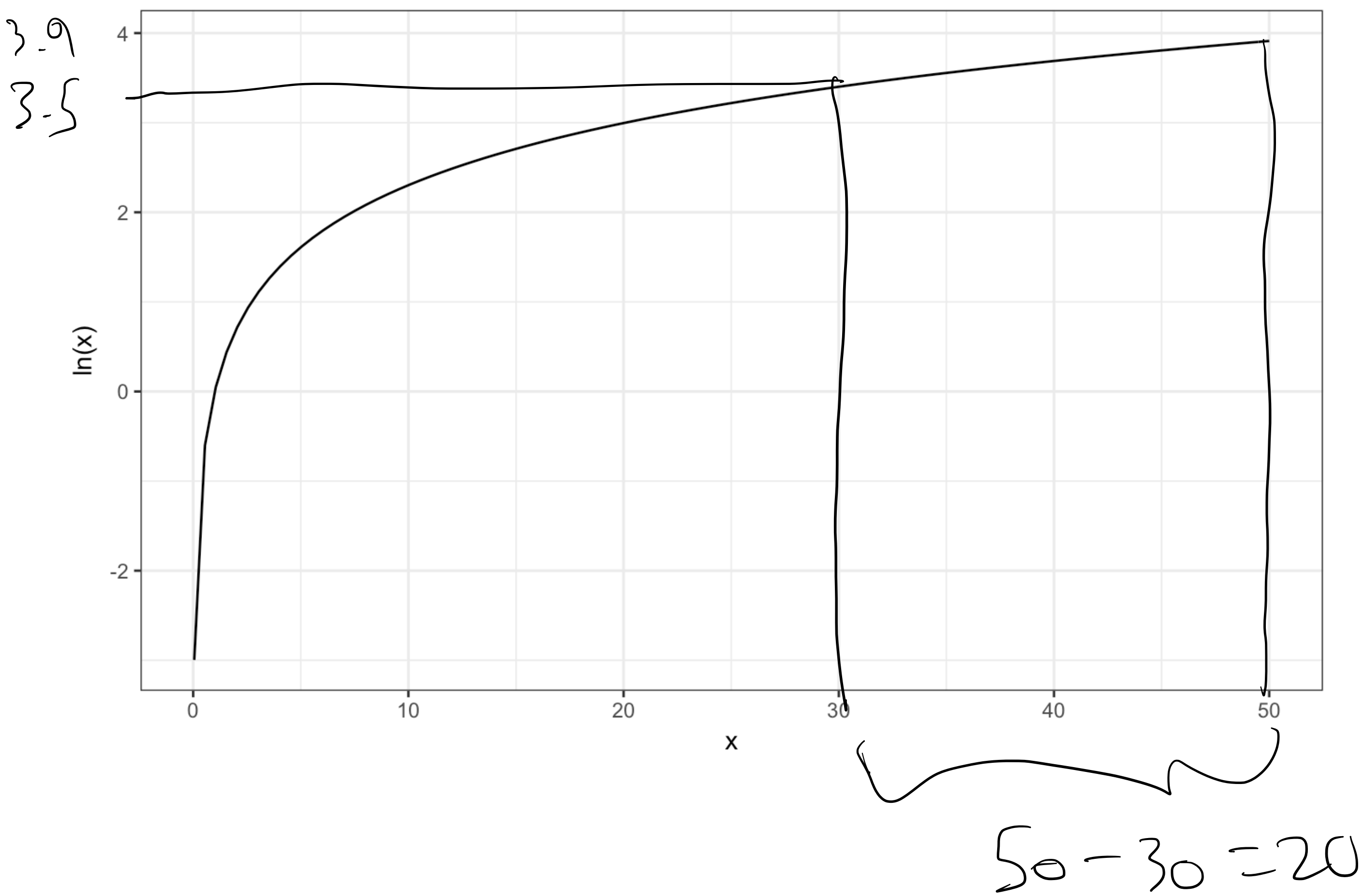
- Common transformations include: $f(x) = \ln(x)$, and $f(x) = x^p, \; p \in \mathbb{R}.$ For example, if $p = 1/2$ then f is the square-root transformation.

$x^{1/2}$ $\quad$ $x^{1/3}$

Logarithmic transformation

- Logarithmic transformation refers to the natural logarithm:

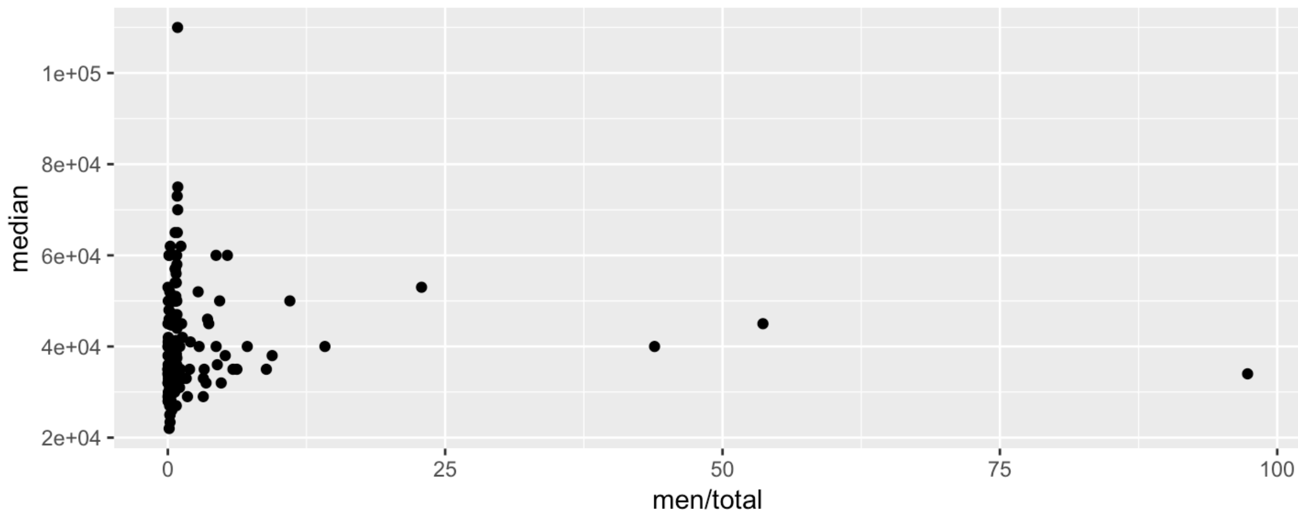
$$y = \log_e(x) \iff \exp(y) = e^y = x$$



Transforming Variables in R

The relationship between Salary (`median`) and percentage of male graduates.

```
college_recent_grads %>%
  ggplot(aes(x = men / total, y = median)) + geom_point()
```



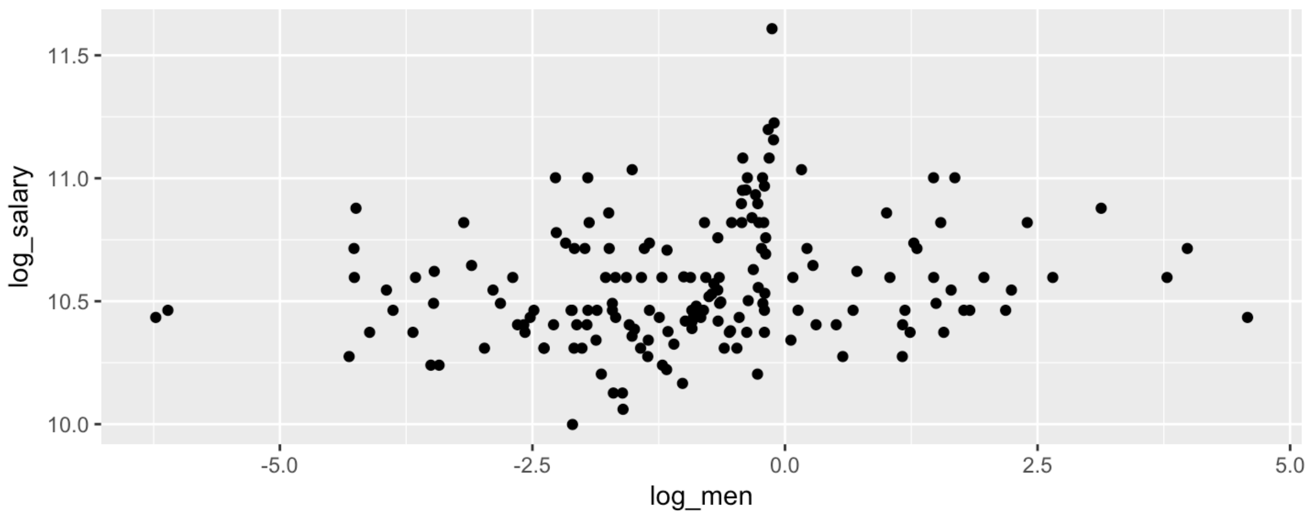
Transforming Variables in R

office hours
SS6027C.

The same plot but on the log-log scale.

piped into ggplot

```
college_recent_grads %>%
  mutate(log_men = log(men / total), log_salary = log(median)) %>%
  ggplot(aes(x = log_men, y = log_salary)) + geom_point()
```



ggplot(data,)