STA130H1F

Class #3

Prof. Nathalie Moon

2018-24-09

Welcome back to STA130 😁

Today's class

- Statistical data
- Tidy data
- Data wrangling
- ~~Transforming data~~
- Boxplots

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 eye colour?

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

Enter variables on STA130 students

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

c(150, 175, 146, ...)

vectors

Put the variables into an R data frame.

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

1 2 3

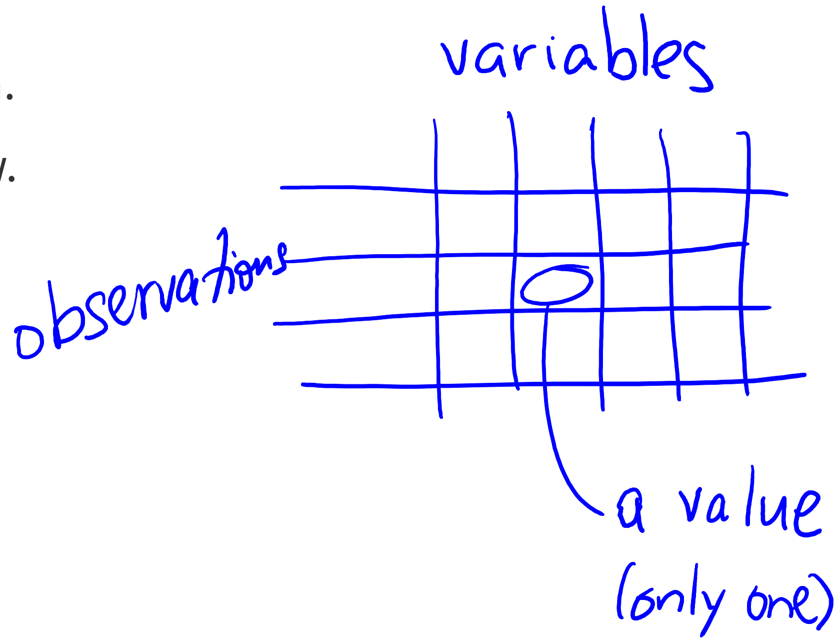
```
sta130_dat <- data_frame(height, years, eye_colour)
```

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

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.



Tidy data: Eye colour example

Suppose that a first year class of 250 students has the following distribution of eye colour.

Colour	N
Brown	105
Green	55
Blue	75
Other	15

Tidy data: Eye colour example

Suppose that a first year class of 250 students has the following distribution of eye colour.

Colour	N
Brown	105
Green	55
Blue	75
Other	15

We can create a tidy data set with a categorical variable `eye_col`.

250 rows

brown	green	blue	other
yes	no	no	no

one option

st#	eye_colour
1	brown
2	blue
3	blue
	green

second option

Tidy data: Eye colour example

Suppose that a first year class of 250 students has the following distribution of eye colour.

Colour	N
Brown	105
Green	55
Blue	75
Other	15

rep("Brown", 105)

↳ vector with "Brown" repeated 105 times

We can create a tidy data set with a categorical variable eye_col.

```
library(tidyverse)
blue_eye <- rep("Brown", 105)
hazel_eye <- rep("Green", 55)
green_eye <- rep("Blue", 75)
other_eye <- rep("Other", 15)
eye_col = c(blue_eye, hazel_eye,
            green_eye, other_eye)
eye_data <- data_frame(stnum = 1:250, eye_col)
glimpse(eye_data)
```

R code for tidy data.

Tidy data: Eye colour

glimpse(eye_data)

```
## Observations: 250
## Variables: 2
## $ stnum   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
## $ eye_col <chr> "Brown", "Brown", "Brown", "Brown", "Brown", "Brown", ...
```

Tidy data

var1 var2
cases/population

Which data set is tidy?

```
## # 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
```

each cell contains one piece of information.

```
## # 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
```

2 pieces of information

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 **grammar of graphics**.
- Similarily the `dplyr` library presents a **grammar for data wrangling**.

both included in tidyverse

The Economic Guide to Picking a Major

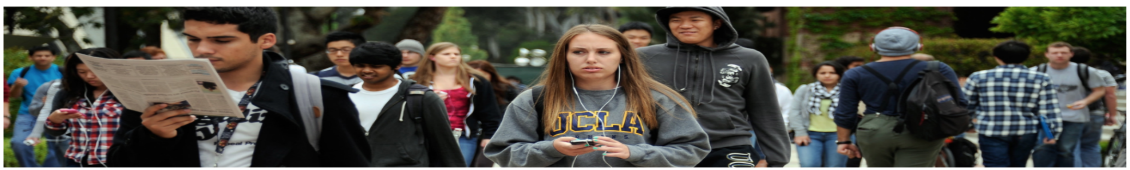
FiveThirtyEight

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.

library(fivethirtyeight)

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, 166...
## $ 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...
## $ p75th                       <dbl> 125000, 90000, 105000, 80000, 7500...
```

# of rows

#of columns

Select variables/columns using `select()` → used to view only certain columns in a dataset

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

$$\underbrace{\texttt{select(college\_recent\_grads,}}_{\text{dataframe}}\ \underbrace{\texttt{major, men, women)}}_{\text{columns we want to keep}}$$

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

```
## # 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                       2110  1667
##  8 Astronomy And Astrophysics               832   960
##  9 Mechanical Engineering                 80320 10907
## 10 Electrical Engineering                 65511 16016
## # ... with 163 more rows
```

Select observations/rows using `filter()` → used to view certain rows only

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.

data

```
# == is a test for equality and is different than =.
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> 65511
## $ women                         <int> 16016
## $ sharewomen                    <dbl> 0.1964503
## $ employed                      <int> 61928
```

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)
```

only majors with median salary of at least 60k

these three variables

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

Respond at **PollEv.com/loop** Text **LOOP** to **(780) 800-5606** once to join, then **A, B, C, D, or E**

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

Total Results: 0

The pipe operator `%>%`

pipe

In the code:

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

previous slide

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

original data

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

first keep only rows for majors with med. income ≥ 60k

with piping.

↳ next we keep only these 3 variables

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 ?

data

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

→ only keep majors w income ≥60k

→ only keep these 3 variables

Compare to nested code:

↳ 5 variables: major, men, women, total, pct_men

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

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

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

from prev. slide

new vars.

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

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

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

logical: T/F

The format of an `ifelse()` statement in R is: `ifelse(test, yes, no)`

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

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

The format of an `ifelse()` statement in R is: `ifelse(test, yes, no)`

female OR female

```
people <- c("Don", "Lei", "Francois", "Fanny")
ifelse(people == "Lei" | people == "Fanny", "Female", "Male")
```

test

c(FALSE, TRUE, FALSE, TRUE)

c("Male", "Female", "Male", "Female")

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

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

The format of an `ifelse()` statement in R is: `ifelse(test, yes, no)`

```
people <- c("Don", "Lei", "Francois", "Fanny")
ifelse(people == "Lei" | people == "Fanny", "Female", "Male")
```

```
## [1] "Male"   "Female" "Male"   "Female"
```

copy into R

AND.
&

test

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

```
## # A tibble: 173 x 2
##    major                                    sex.equal
##    <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
##  6 Nuclear Engineering                      Yes
##  7 Actuarial Science                        Yes
##  8 Astronomy And Astrophysics               Yes
##  9 Mechanical Engineering                   Yes
## 10 Electrical Engineering                   Yes
## # ... with 163 more rows
```

No
No
wrong.
No
No
Yes
No.

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`.

same as before

```
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, yes="Yes", no="No")) %>%
  select(major,sex.equal, median)
```

correct

new

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

new name old name

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

Sort a data frame using `arrange()`

↳ used to sort the rows in dataframe

```
my_college_dat %>%
  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, na.rm = T), N = n())
```

exclude NAs from calculation.

mean # of women grads in each major

total # of majors

```
## # A tibble: 1 x 2
##   femgrad_mean     N
##          <dbl> <int>
## 1       22647.   173
```

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

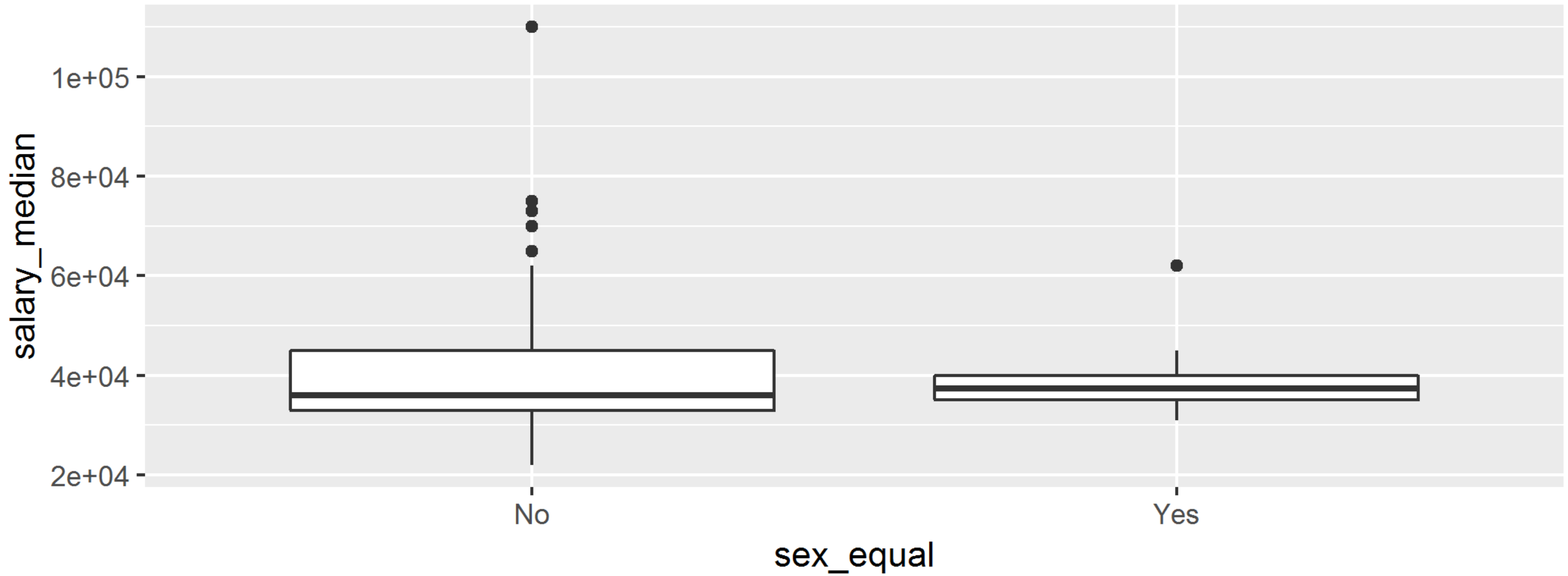
The median salary in majors with 45%-55% female students.

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

```
## # A tibble: 3 x 2
##   sex_equal `median(salary_median)`
##   <chr>                       <dbl>
## 1 No                          36000
## 2 Yes                         37400
## 3 <NA>                        53000
```

Boxplots to compare distribution of salary in majors with balanced vs unbalanced sex distributions

```
my_college_dat %>% filter(is.na(sex_equal) == FALSE) %>%
  ggplot(aes(x = sex_equal, y = salary_median)) + geom_boxplot()
```



What is a boxplot?

What is a boxplot?

A boxplot summarizes the distribution of a quantitative (numerical) variable using five statistics, while also plotting unusual observations (outliers)

The elements of a boxplot are:

What is a boxplot?

A boxplot summarizes the distribution of a quantitative (numerical) variable using five statistics, while also plotting unusual observations (outliers)

The elements of a boxplot are:

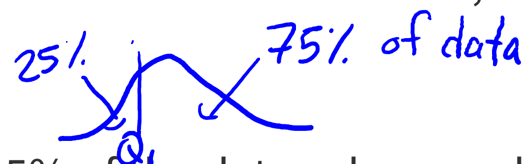
- Line in the middle of the box:
    - **median**: middle data value (50% of data values above, 50% below)

What is a boxplot?

A boxplot summarizes the distribution of a quantitative (numerical) variable using five statistics, while also plotting unusual observations (outliers)

The elements of a boxplot are:

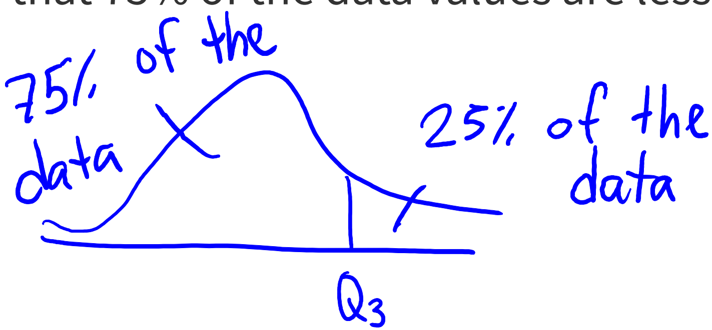
- Line in the middle of the box:
    - **median**: middle data value (50% of data values above, 50% below)
- Edges of the box:



Q1

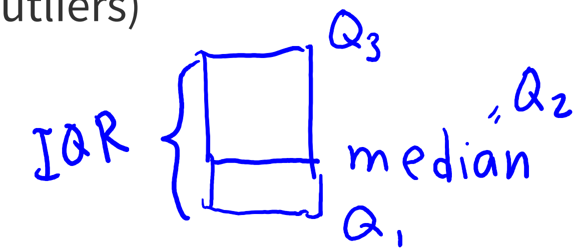
    - **1st quartile**: value such that 25% of the data values are less it (Q_1)
    - **3rd quartile** - value such that 75% of the data values are less it (Q_3)

Q3



What is a boxplot?

A boxplot summarizes the distribution of a quantitative (numerical) variable using five statistics, while also plotting unusual observations (outliers)



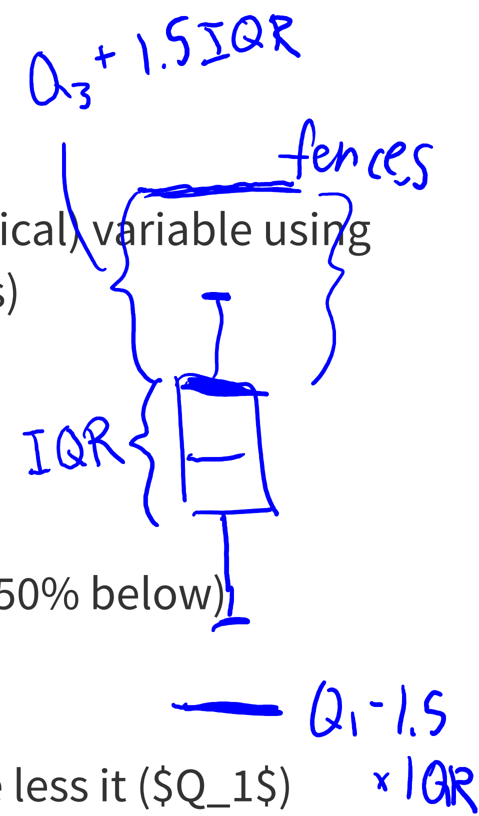
The elements of a boxplot are:

- Line in the middle of the box:
  - **median**: middle data value (50% of data values above, 50% below)
- Edges of the box:
  - **1st quartile**: value such that 25% of the data values are less it (Q_1)
  - **3rd quartile** - value such that 75% of the data values are less it (Q_3)
- Length of the box is the Interquartile Range (IQR) = 3rd quartile - 1st quartile
  - gives an indication of how spread out the data are $Q_3 - Q_1$

$Q_3 + 1.5 IQR$

What is a boxplot?

A boxplot summarizes the distribution of a quantitative (numerical) variable using five statistics, while also plotting unusual observations (outliers)



The elements of a boxplot are:

- Line in the middle of the box:
  - **median**: middle data value (50% of data values above, 50% below)
- Edges of the box:
  - **1st quartile**: value such that 25% of the data values are less it (Q_1)
  - **3rd quartile** - value such that 75% of the data values are less it (Q_3)
- Length of the box is the Interquartile Range (IQR) = 3rd quartile - 1st quartile
  - gives an indication of how spread out the data are
- Whiskers on the box extend to the most extreme value that is outside of the box but within $1.5 \times IQR$

whiskers can't go beyond the fences

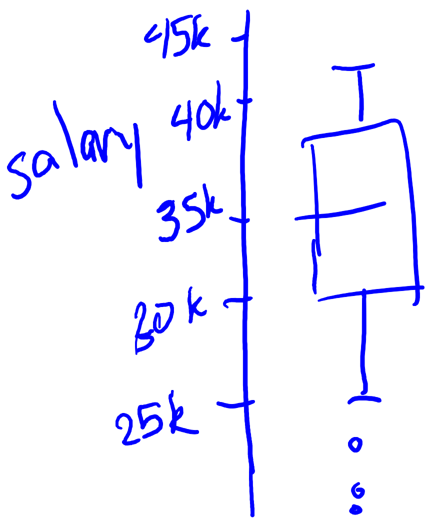
Outliers – unusual values

An *outlier* is a value which is beyond the whiskers, that is either

- less than $Q_1 - 1.5 \times IQR$ OR
- more than $Q_3 + 1.5 \times IQR$

The whiskers of the boxplot capture data which is outside the box, but not beyond $1.5 \times IQR$ on either side

↳ outliers are represented on the plot with dots



Example

```
x
```

$n = 10$

```
##  [1] 0.14 0.15 0.15 0.44 0.54 0.76 0.96 1.18 1.23 2.89
```

Example

```
x
```

```
##  [1] 0.14 0.15 0.15 0.44 0.54 0.76 0.96 1.18 1.23 2.89
```

```
quantile(x, 0.25)
```

```
##    25%
## 0.2225
```

Q_1

```
quantile(x, 0.50)
```

```
##  50%
## 0.65
```

Q_2 = median

```
quantile(x, 0.75)
```

Q_3

```
##   75%
## 1.125
```

```
quantile(x, 0.75) -
  quantile(x, 0.25)
```

```
##    75%
## 0.9025
```

$IQR = Q_3 - Q_1$

Example

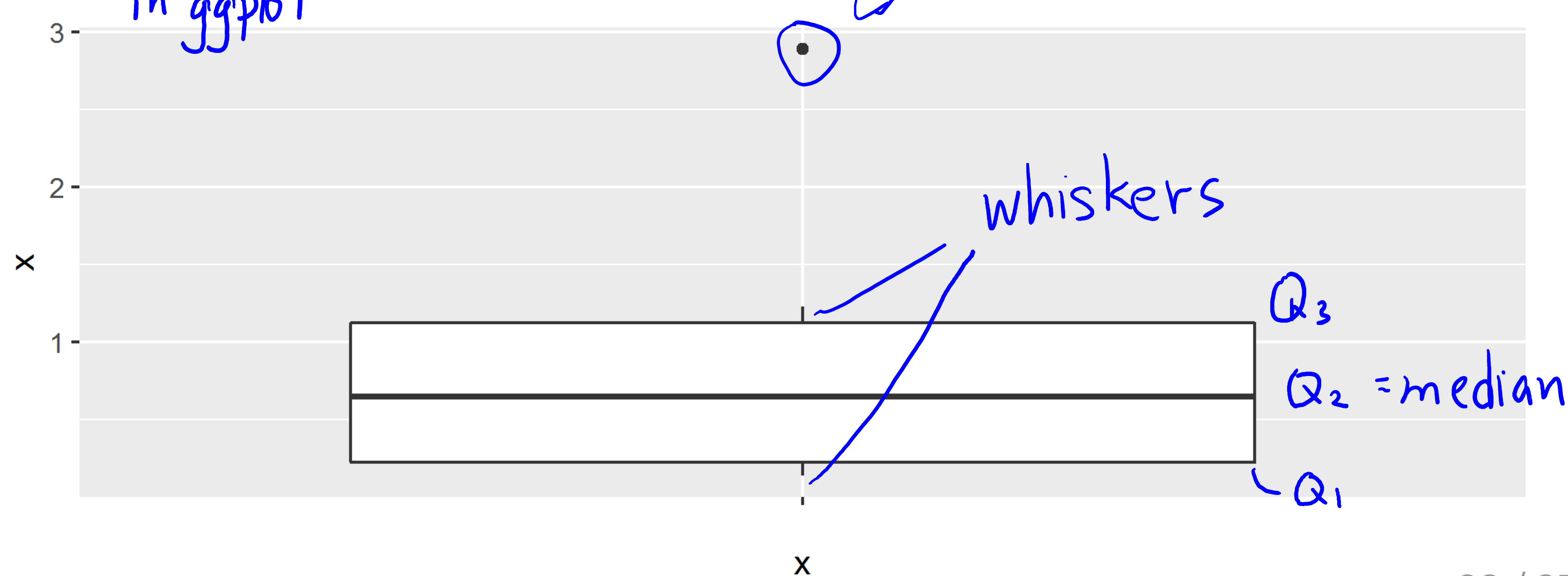
```
x
```

```
##  [1] 0.14 0.15 0.15 0.44 0.54 0.76 0.96 1.18 1.23 2.89
```

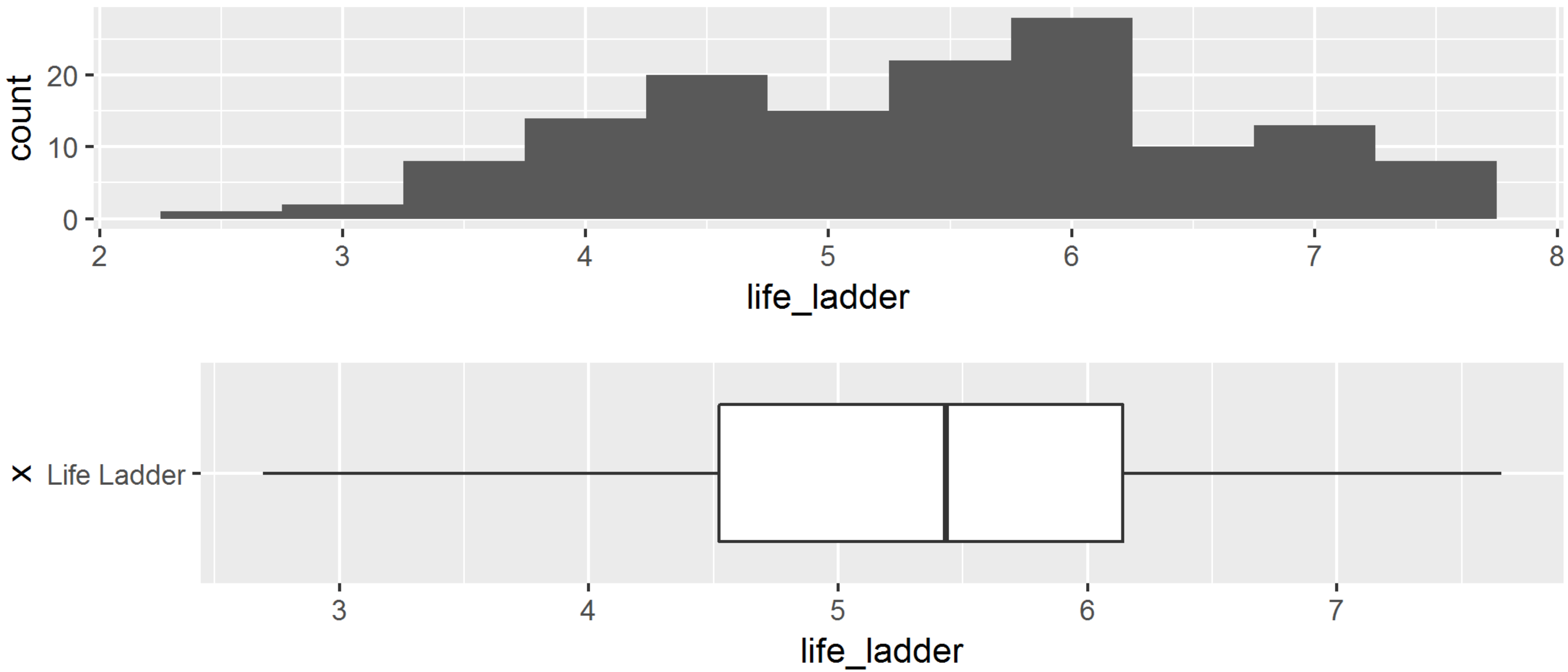
```
data_frame(x) %>%
  ggplot(aes(x = "", y = x)) +
  geom_boxplot()
```

in ggplot

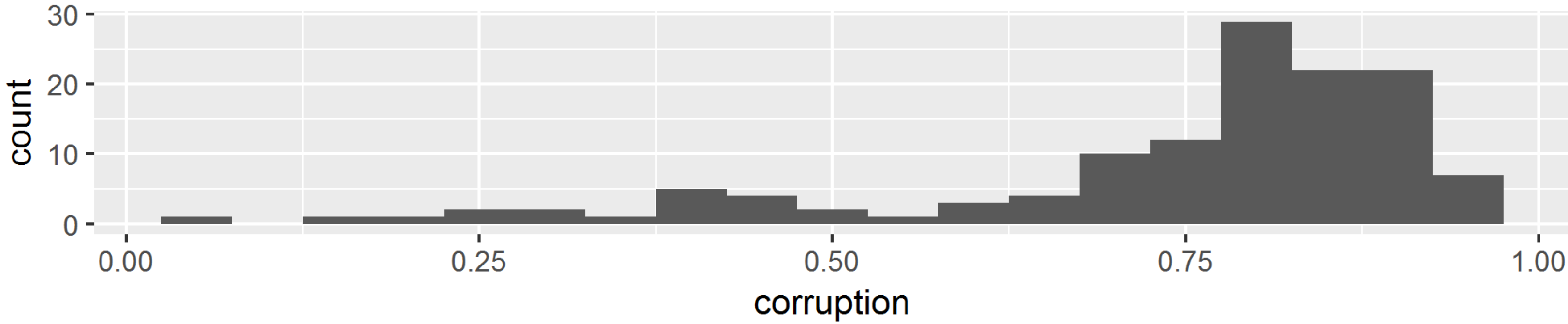
x-axis

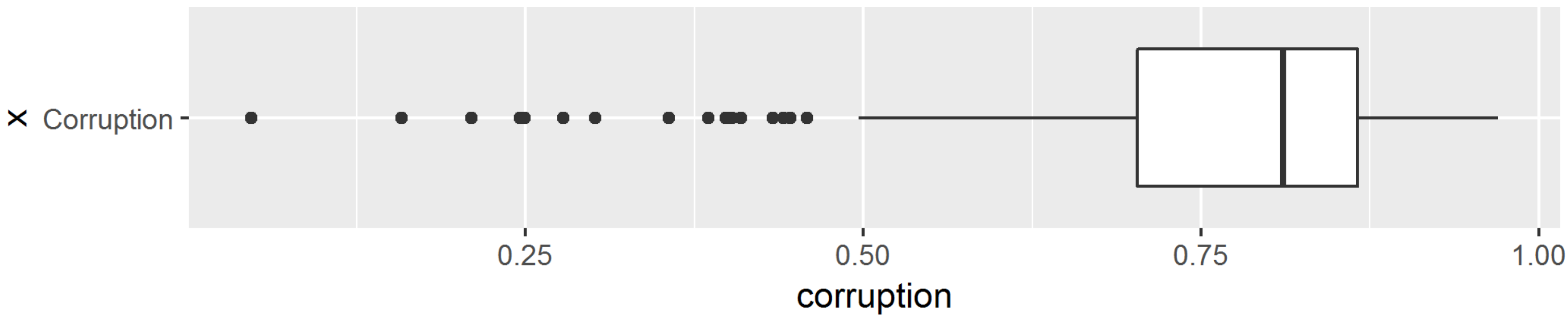


Example: Histogram vs boxplot

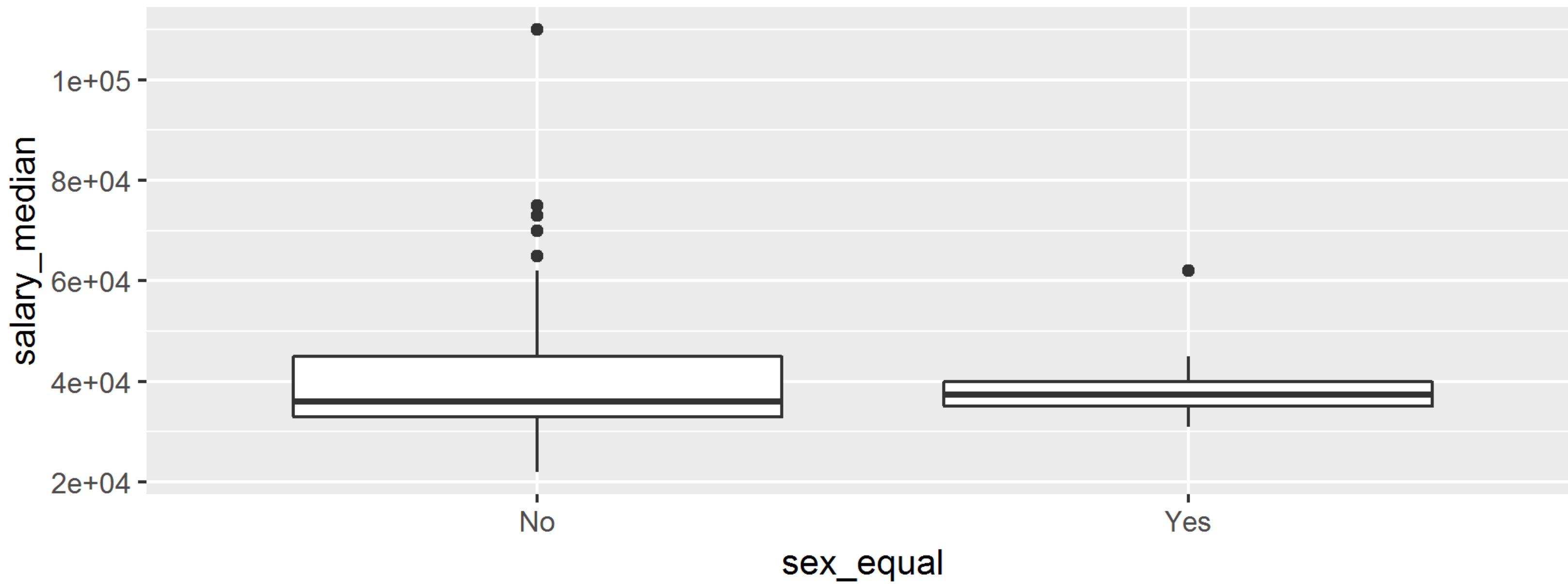


Example: Histogram vs boxplot

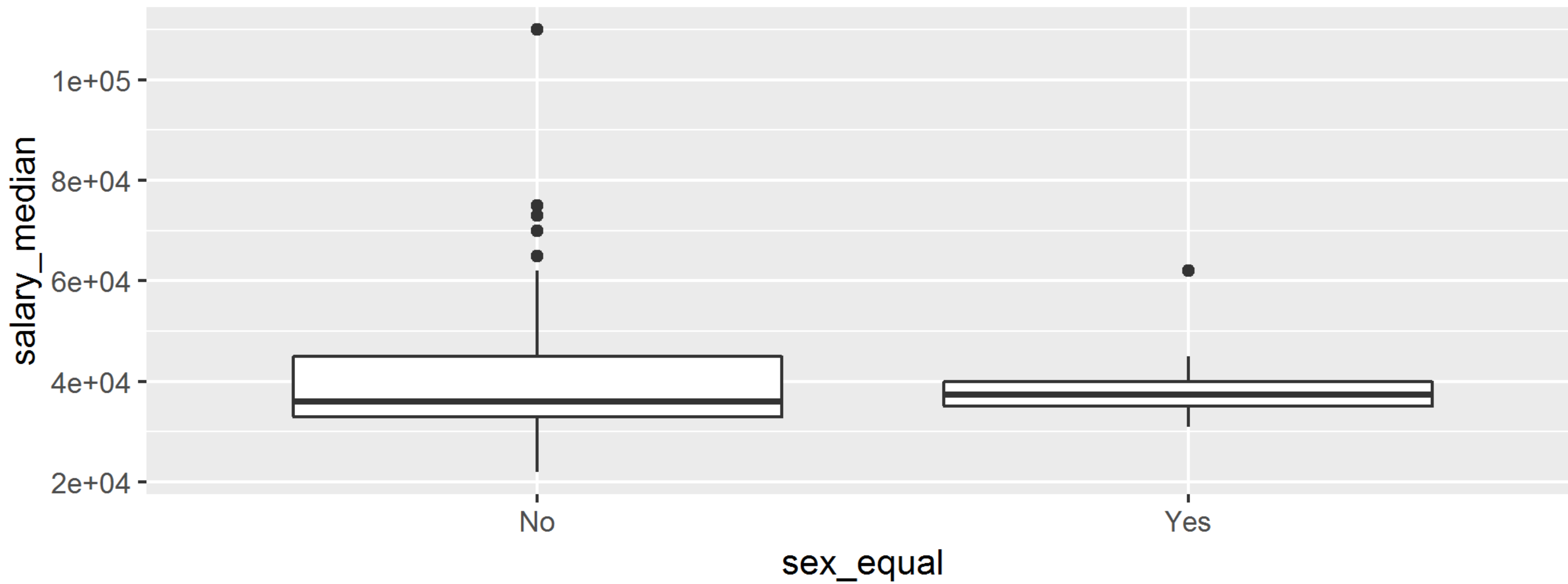




Back to the salary example...



Back to the salary example...



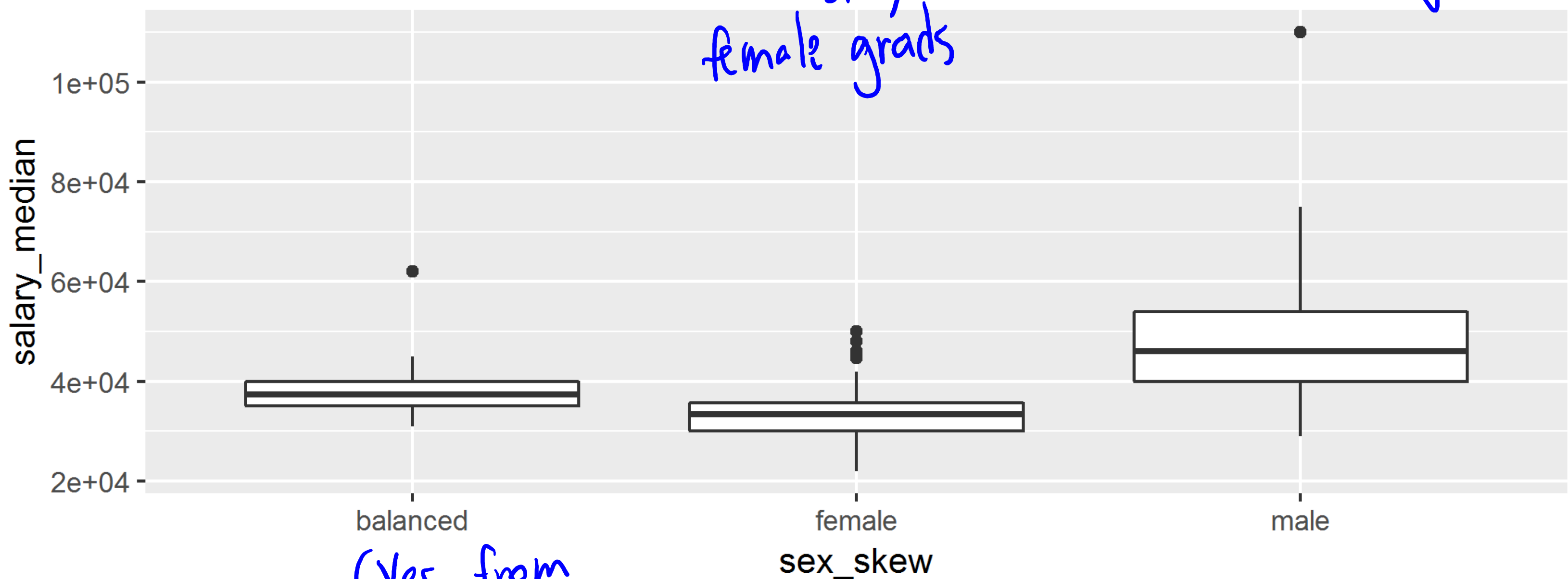
- Similar medians? ↳ Yes
- Range / variability?
  ↳ look at the height of the boxes
  ↳ whiskers and outliers

} variability is different across the two levels of sex_equals

Boxplots to compare distribution of salary in majors with mostly men vs mostly women vs balanced

mostly male grads

mostly female grads



(Yes from prev slide)