Ideas for Pre-institute

Thoughts

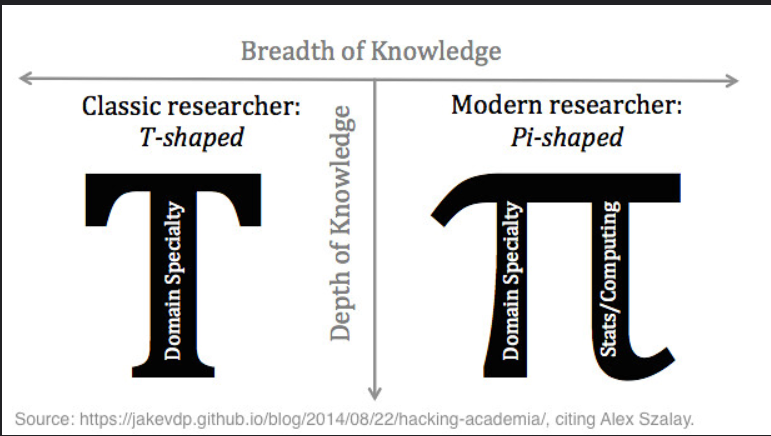
“Next to Knowing is knowing where to find out”

You can say they you are Not good at computers, and still use them. … the same goes with Math

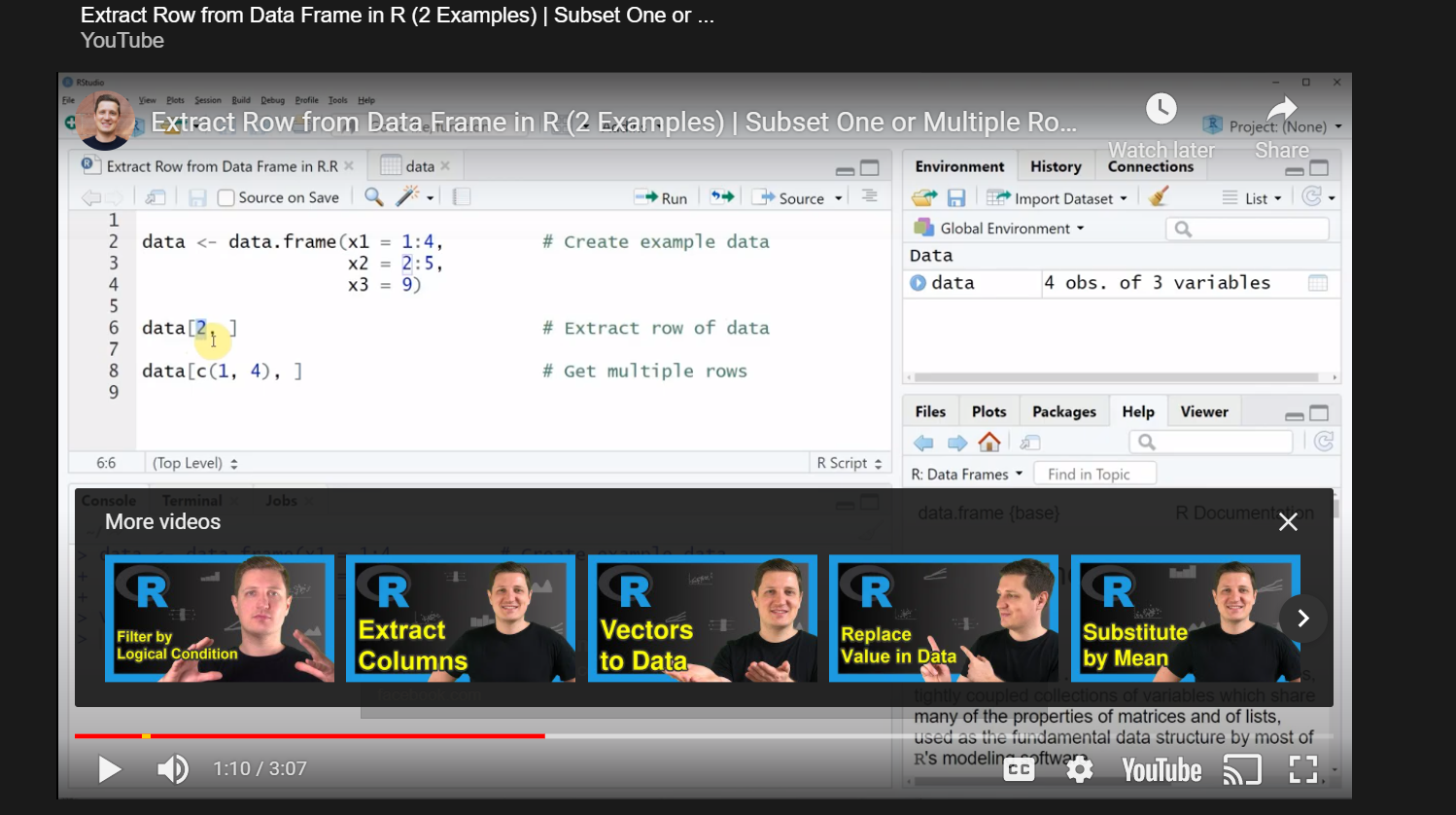
Code of conduct university of Washington neuroinformatics.uw.edu eScience Institute

<https://escience.washington.edu/code-of-conduct>

“pi-shaped” faculty -



1. Using rows and columns from data <https://www.google.com/search?q=find+rows+in+R&rlz=1C1GCEU_enPR903PR903&sxsrf=ALeKk01jsGjAbmvLoCBf9UviUm3qauMi5g%3A1621443618908&ei=IkSlYMj3NouOwbkPjtqT-AY&oq=find+rows+in+R&gs_lcp=Cgdnd3Mtd2l6EAM6BwgAEEcQsAM6BQgAEJECOgIIADoHCAAQhwIQFFDSGViYHmDGH2gBcAJ4AIAB4gGIAckGkgEFMC41LjGYAQCgAQGqAQdnd3Mtd2l6yAEIwAEB&sclient=gws-wiz&ved=0ahUKEwjIkLaXnNbwAhULRzABHQ7tBG8Q4dUDCA4&uact=5#kpvalbx=_KUSlYOWqAb-OwbkPyfCgkAw21>



1. Basic R: <http://stanford.edu/~jgrimmer/RDataManagement.pdf>
2. Using Mutate and Pipe operator lines of codes in R

More info about MUTATE <https://dplyr.tidyverse.org/reference/mutate.html>

polls <- polls\_us\_election\_2016 %>% filter(enddate >= "2016-10-31" & state == "U.S.")

polls <- polls %>% mutate(d\_hat = rawpoll\_clinton/100 - rawpoll\_trump/100)

1. Actual elections popular vote proportions: 0.482 (Clinton) – 0.461 (Trump)
2. Pipe usage in graphs

# Add variable called `error` to the object `polls` that contains the difference between d\_hat and the actual difference on election day. Then make a plot of the error stratified by pollster.

polls <- polls %>% mutate(error= polls$d\_hat - 0.021)

polls %>% ggplot(aes(x=pollster, y=error))+geom\_point()+theme(axis.text.x = element\_text(angle=90,hjust=1))

1. Generate Data: a Sampling code

# The vector `p` contains 100 proportions of Democrats ranging from 0 to 1 using the `seq` function

p <- seq(0, 1, length = 100)

# The vector `sample\_sizes` contains the three sample sizes

sample\_sizes <- c(25, 100, 1000)

# Write a for-loop that calculates the standard error `se` for every value of `p` for each of the three samples sizes `N` in the vector `sample\_sizes`. Plot the three graphs, using the `ylim` argument to standardize the y-axis across all three plots.

for (ii in sample\_sizes)

{

    se=sqrt(p\*(1-p)/ii)

    plot(p,se,ylim=c(0,0.1))

2) Confidence Interval of 95% MonteCarlo simulation

**Key points**

* We can run a Monte Carlo simulation to confirm that a 95% confidence interval contains the true value of p 95% of the time.
* A plot of confidence intervals from this simulation demonstrates that most intervals include p, but roughly 5% of intervals miss the true value of p.

**Code: Monte Carlo simulation**

Note that to compute the exact 95% confidence interval, we would use qnorm(.975)\*SE\_hat instead of 2\*SE\_hat. P = 0.45, N = 100

B <- 10000  
inside <- replicate(B, {

X <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))

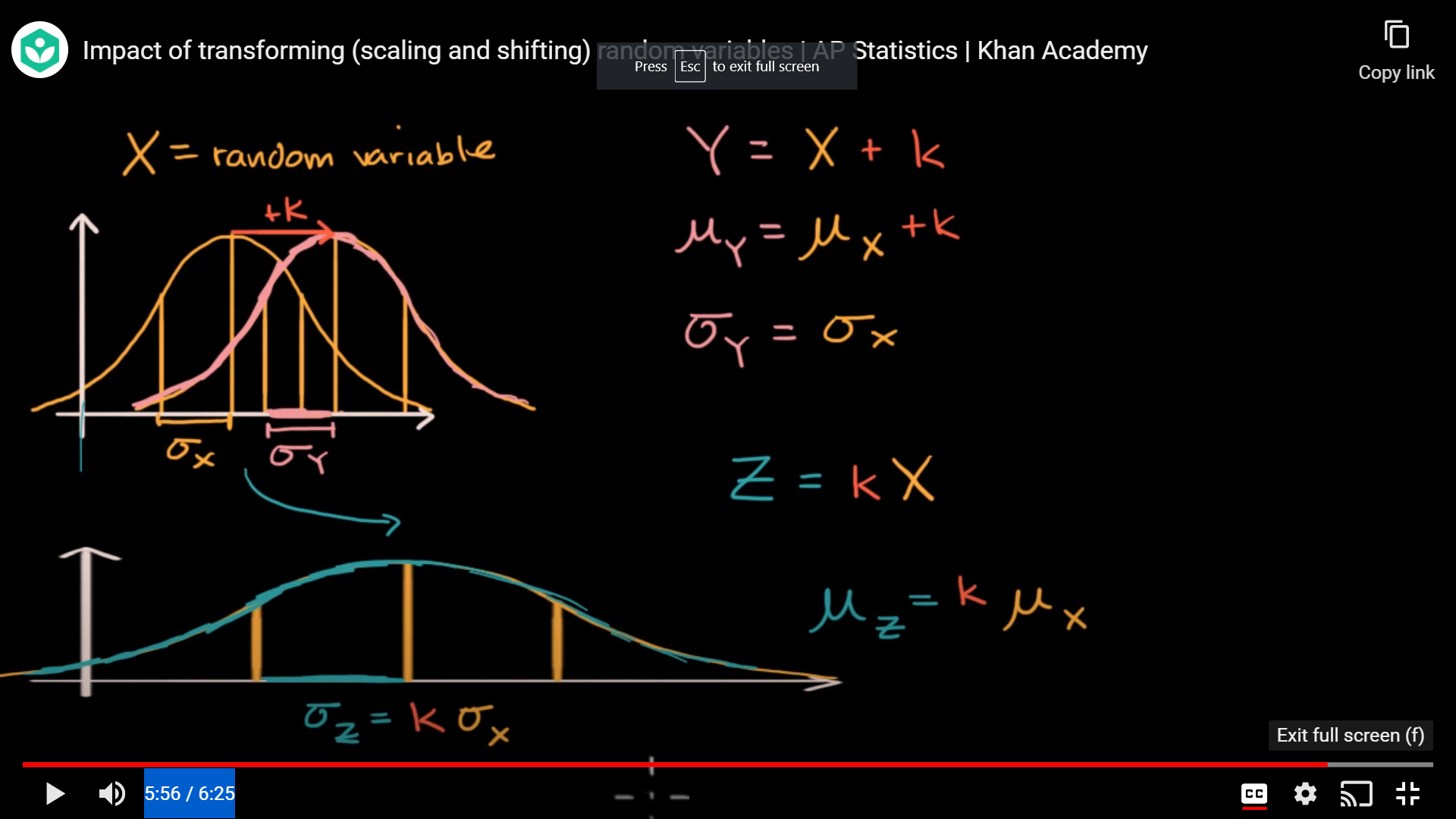
X\_hat <- mean(X)  
 SE\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)

between(p, X\_hat - 2\*SE\_hat, X\_hat + 2\*SE\_hat) # TRUE if p in confidence interval

})

mean(inside)

RANDOM VARIABLES: Reference Khan Academy



Study the code:

library(tidyverse)

> library(tidyverse)

-- **Attaching packages** -------------

v ggplot2 3.3.3 v purrr 0.3.4

v tibble 3.1.1 v dplyr 1.0.5

v tidyr 1.1.3 v stringr 1.4.0

v readr 1.4.0 v forcats 0.5.1

install.packages('dslabs')

d <- 0.039 #the spread of the Obama elections- difference or spread

Ns <- c(1298, 533, 1342, 897, 774, 254, 812, 324, 1291, 1056, 2172, 516)

p <- (d+1)/2 # since d = 2p – 1

# calculate confidence intervals of the spread for each of the 12

Polls calculating the mean of the spread (2p – 1), low and high boundaries of the confidence interval of the spread using: mean X\_hat, standard error SE\_hat, and confidence interval

confidence\_intervals <- sapply(Ns, function(N){

X <- sample(c(0,1), size=N, replace=TRUE, prob = c(1-p, p))

X\_hat <- mean(X)

SE\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)

2\*c(X\_hat, X\_hat - 2\*SE\_hat, X\_hat + 2\*SE\_hat) - 1

})

# generate a data frame storing results

polls <- data.frame(poll = 1:ncol(confidence\_intervals),

t(confidence\_intervals), sample\_size = Ns)

names(polls) <- c("poll", "estimate", "low", "high", "sample\_size")

polls

#confidence\_intervals 3 x 12 built using sapply to the function(N), #transposed by t()

#Output

> head(polls)

poll estimate low

1 1 0.02003082 -0.0354707836

2 2 0.09193246 0.0056696966

3 3 0.04470939 -0.0098311269

4 4 0.06577480 -0.0008586431

5 5 -0.01033592 -0.0822205927

6 6 0.17322835 0.0496343999

high sample\_size

1 0.07553242 1298

2 0.17819522 533

3 0.09924990 1342

4 0.13240825 897

5 0.06154876 774