

# Introductory Statistics with R

Clay Ford, UVa Library StatLab

Spring 2020

```
# To submit a line of code, place your cursor in the line and hit Ctrl + Enter
# (Win/Linux) or Cmd + Enter (Mac)

# load packages -----

library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0 --
## v ggplot2 3.2.1      v purrr   0.3.3
## v tibble  2.1.3      v dplyr  0.8.3
## v tidyr   1.0.2      v stringr 1.4.0
## v readr   1.3.1      v forcats 0.4.0

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(binom)

# Suppress the numerous start up messages these two packages produce
suppressPackageStartupMessages(library(Hmisc))
suppressPackageStartupMessages(library(mosaic))

# Getting data into R -----

# Results from the US Census American Community Survey, 2012.
# From the openintro package

# Import the acs12.csv file; the result is a data frame
acs12 <- read.csv("data/acs12.csv")

# str() shows us the structure of the data frame
str(acs12)

## 'data.frame':   2000 obs. of  13 variables:
## $ income      : int  60000 0 NA 0 0 1700 NA NA NA 45000 ...
## $ employment  : Factor w/ 3 levels "employed","not in labor force",...: 2 2 NA 2 2 1 NA NA NA 1 ...
## $ hrs_work     : int  40 NA NA NA NA 40 NA NA NA 84 ...
## $ race         : Factor w/ 4 levels "asian","black",...: 4 4 4 4 4 3 4 3 1 4 ...
## $ age         : int  68 88 12 17 77 35 11 7 6 27 ...
## $ gender      : Factor w/ 2 levels "female","male": 1 2 1 2 1 1 2 2 2 2 ...
```

```
## $ citizen      : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 ...
## $ time_to_work: int   NA NA NA NA NA 15 NA NA NA 40 ...
## $ lang         : Factor w/ 2 levels "english","other": 1 1 1 2 2 2 1 1 2 1 ...
## $ married      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 1 1 1 2 ...
## $ edu          : Factor w/ 3 levels "college","grad",...: 1 3 3 3 3 3 3 3 3 3 ...
## $ disability   : Factor w/ 2 levels "no","yes": 1 2 1 1 2 2 1 2 1 1 ...
## $ birth_qrtr   : Factor w/ 4 levels "apr thru jun",...: 3 2 4 4 3 3 4 3 2 4 ...
```

```
# NOTE: Factors are integers with character labels. Good to think of them as
# categorical variables. By default the read.csv() function automatically
# converts all columns containing text into factors.
#
# Can also click on the name in the Environment window to view the data.
#
# The summary function will provide summaries for all columns in a data frame.
# NA means "Not available", or missing. May not be useful for data with many
# columns.
summary(acs12)
```

```
##      income      employment      hrs_work      race
## Min.      :    0      employed      :843      Min.      : 1.00      asian:  87
## 1st Qu.:    0      not in labor force:656      1st Qu.:32.00      black: 206
## Median : 3000      unemployed      :106      Median :40.00      other: 152
## Mean   : 23600      NA's          :395      Mean    :37.98      white:1555
## 3rd Qu.: 33700
## Max.    :450000
## NA's    :377
##      age      gender      citizen      time_to_work      lang
## Min.      : 0.00      female: 969      no : 118      Min.      : 1      english:1527
## 1st Qu.:19.75      male  :1031      yes:1882      1st Qu.: 10      other  : 368
## Median :40.00
## Mean     :40.22
## 3rd Qu.:59.00
## Max.     :94.00
## NA's     :1217
##      married      edu      disability      birth_qrtr
## no :1167      college      : 359      no :1676      apr thru jun:479
## yes: 833      grad        : 144      yes: 324      jan thru mar:485
##      hs or lower:1439
##      NA's          : 58
##      oct thru dec:532
##
##
##
```

```
# What am I typically looking for?
# - Unusual values (too low, too high)
# - missing data (NA's)
# - big differences between mean and median (skewed data?)
# - consistent Factor level names ("male" vs "Male")
# - excess zeroes, or some other value
#
# Another option to get all summaries is the describe function from the Hmisc
# package
Hmisc::describe(acs12)
```

```
## acs12
```

```

##
## 13 Variables      2000 Observations
## -----
## income
##      n missing distinct      Info      Mean      Gmd      .05      .10
##    1623      377      266    0.909    23600    35276         0         0
##      .25      .50      .75      .90      .95
##        0     3000     33700    64000    92000
##
## lowest :      0      50      100      110      130, highest: 345000 360000 398000 399000 450000
## -----
## employment
##      n missing distinct
##    1605      395         3
##
## Value              employed not in labor force              unemployed
## Frequency              843              656              106
## Proportion              0.525              0.409              0.066
## -----
## hrs_work
##      n missing distinct      Info      Mean      Gmd      .05      .10
##    959    1041         55    0.919    37.98    13.8         10         20
##      .25      .50      .75      .90      .95
##        32       40       40       50       60
##
## lowest :  1  2  4  5  6, highest: 72 75 80 84 99
## -----
## race
##      n missing distinct
##    2000         0         4
##
## Value      asian black other white
## Frequency      87    206    152  1555
## Proportion 0.044 0.103 0.076 0.778
## -----
## age
##      n missing distinct      Info      Mean      Gmd      .05      .10
##    2000         0       95         1    40.22    27.24     4.00     8.00
##      .25      .50      .75      .90      .95
##    19.75    40.00    59.00    71.00    79.00
##
## lowest :  0  1  2  3  4, highest: 90 91 92 93 94
## -----
## gender
##      n missing distinct
##    2000         0         2
##
## Value      female      male
## Frequency      969    1031
## Proportion 0.484 0.516
## -----
## citizen
##      n missing distinct
##    2000         0         2

```

```

##
## Value          no   yes
## Frequency      118 1882
## Proportion 0.059 0.941
## -----
## time_to_work
##      n missing distinct      Info      Mean      Gmd      .05      .10
##      783   1217      44    0.989      26    21.49      5      5
##      .25    .50    .75    .90    .95
##      10     20     30     50     60
##
## lowest :    1    2    3    4    5, highest: 145 148 152 157 163
## -----
## lang
##      n missing distinct
##      1895     105      2
##
## Value      english  other
## Frequency      1527    368
## Proportion    0.806   0.194
## -----
## married
##      n missing distinct
##      2000      0      2
##
## Value          no   yes
## Frequency      1167   833
## Proportion 0.584 0.416
## -----
## edu
##      n missing distinct
##      1942     58      3
##
## Value          college      grad hs or lower
## Frequency          359        144        1439
## Proportion          0.185      0.074      0.741
## -----
## disability
##      n missing distinct
##      2000      0      2
##
## Value          no   yes
## Frequency      1676   324
## Proportion 0.838 0.162
## -----
## birth_qtrtr
##      n missing distinct
##      2000      0      4
##
## Value      apr thru jun jan thru mar jul thru sep oct thru dec
## Frequency          479          485          504          532
## Proportion          0.240          0.242          0.252          0.266
## -----

```

```
# We can also use summary on individual columns. Use the "$" to access columns
# in a data frame.
```

```
summary(acs12$income)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
##         0         0   3000   23600   33700  450000    377
```

```
summary(acs12$employment)
```

```
##      employed not in labor force      unemployed      NA's
##         843          656          106          395
```

```
# Notice that RStudio allows autocompletion of column names after you type the
# "$".
```

```
# YOUR TURN #0 -----
```

```
# Try summary on the time_to_work and edu columns, or any other columns of
# interest.
```

```
# Counts and Proportions -----
```

```
# Counts - the most fundamental statistic
```

```
# The table() function generates counts of unique values for a given vector,
# such as a column in a data frame. By default it excludes missing data.
```

```
table(acs12$employment)
```

```
##
##      employed not in labor force      unemployed
##         843          656          106
```

```
# To see if there are missing data, set exclude = NULL:
```

```
table(acs12$employment, exclude = NULL)
```

```
##
##      employed not in labor force      unemployed      <NA>
##         843          656          106          395
```

```
# Can also exclude other values, for example
```

```
table(acs12$employment, exclude = "not in labor force")
```

```
##
##      employed unemployed      <NA>
##         843          106          395
```

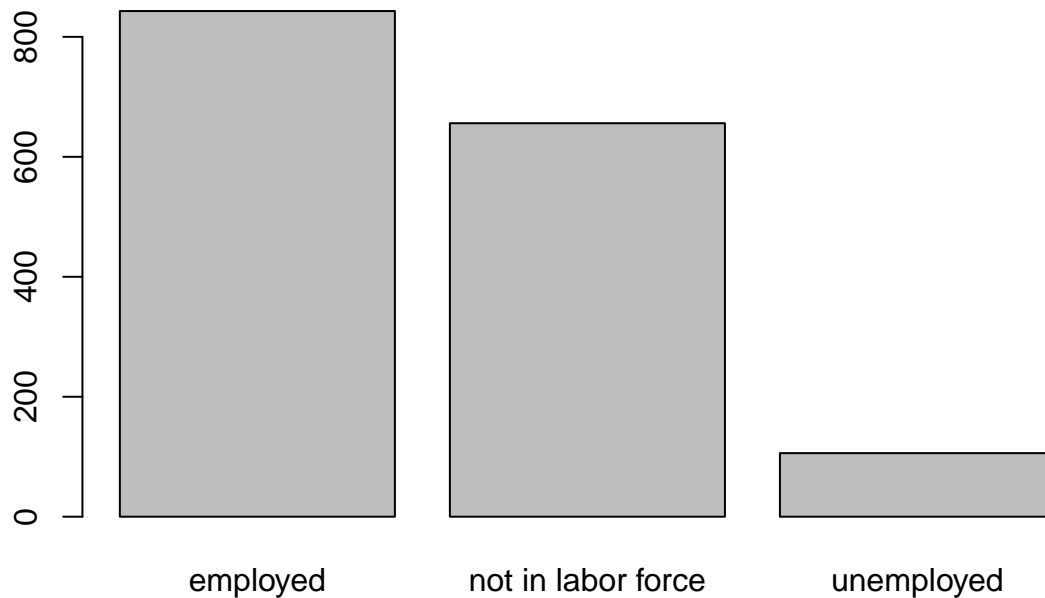
```
table(acs12$employment, exclude = c("not in labor force", NA))
```

```
##
##      employed unemployed
##         843          106
```

```
# summary() returns NAs by default (for Factors)
```

```
summary(acs12$employment)
```

```
##          employed not in labor force      unemployed      NA's
##          843          656          106          395
# quick way to visualize counts, as long as the column is a Factor:
plot(acs12$employment)
```

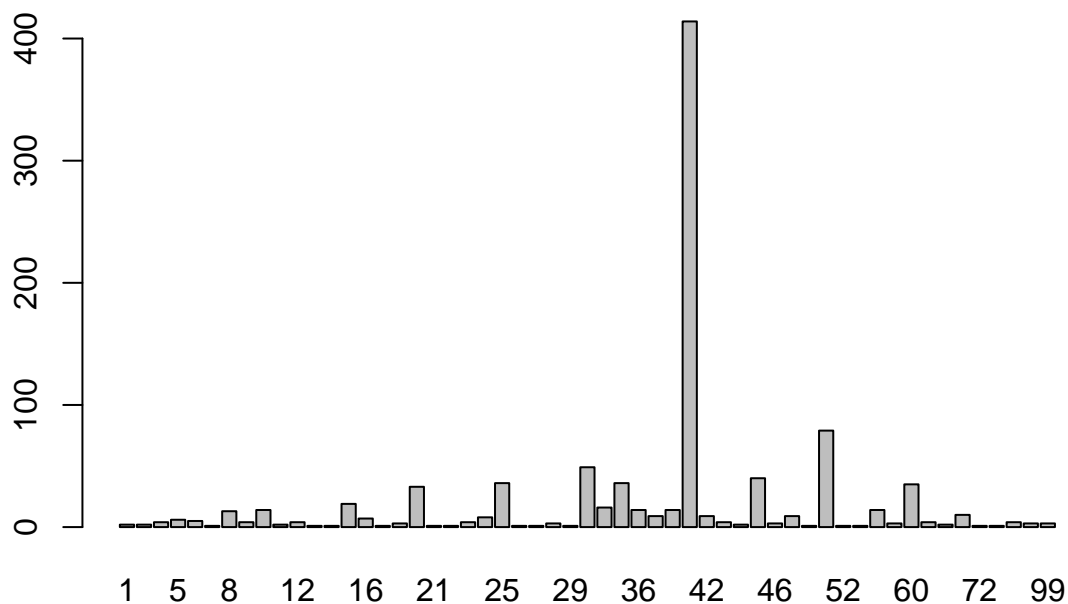


```
# Or we can use barplot() with table(); table() will count anything that can be
# converted to a factor (integers, numeric, character).
```

```
# counts of subjects by number of hours they work each week
table(acs12$hrs_work)
```

```
##
##  1  2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 20 21 22
##  2  2  4  6  5  1 13  4 14  2  4  1  1 19  7  1  3 33  1  1
## 23 24 25 26 27 28 29 30 32 35 36 37 38 40 42 43 44 45 46 48
##  4  8 36  1  1  3  1 49 16 36 14  9 14 414  9  4  2 40  3  9
## 49 50 52 53 55 56 60 65 68 70 72 75 80 84 99
##  1 79  1  1 14  3 35  4  2 10  1  1  4  3  3
```

```
barplot(table(acs12$hrs_work))
```



```
# We can also count things satisfying a condition.
```

```
#
```

```
# == EQUAL
```

```
# != NOT EQUALS
```

```
# > GREATER THAN
```

```
# < LESS THAN
```

```
# >= GREATER THAN OR EQUAL
```

```
# <= LESS THAN OR EQUAL
```

```
# How many people are age 60?
```

```
sum(acs12$age == 60)
```

```
## [1] 28
```

```
# How many people are over age 60
```

```
sum(acs12$age > 60)
```

```
## [1] 466
```

```
# How many people commute more than 60 minutes to work?
```

```
sum(acs12$time_to_work > 60)
```

```
## [1] NA
```

```
# In certain cases, R does not skip missing values unless told to do so.
```

```
# Set na.rm = TRUE
```

```
sum(acs12$time_to_work > 60, na.rm = TRUE)
```

```
## [1] 36
```

```

# How many people do NOT have employment = "employed"
sum(acs12$employment != "employed", na.rm=TRUE)

## [1] 762

# Proportions
#
# What proportion of people are over age 60? The mean of 0s and 1s is the
# proportion of 1s.
mean(acs12$age > 60)

## [1] 0.233

# What proportion of people commute over 60 minutes to work?
mean(acs12$time_to_work > 60, na.rm = TRUE)

## [1] 0.04597701

# Use prop.table to get proportions from tables

# NOTE: these are proportions of non-missing!
prop.table(table(acs12$employment))

##
##      employed not in labor force      unemployed
##      0.52523364      0.40872274      0.06604361

# Notice the difference if we include the missings
prop.table(table(acs12$employment, exclude = NULL))

##
##      employed not in labor force      unemployed      <NA>
##      0.4215      0.3280      0.0530      0.1975

# or using summary
prop.table(summary(acs12$employment))

##      employed not in labor force      unemployed      NA's
##      0.4215      0.3280      0.0530      0.1975

# Quick note about pipes. If the dplyr package is loaded, we can use pipes
# instead of nesting functions. For example:
table(acs12$employment) %>% prop.table()

##
##      employed not in labor force      unemployed
##      0.52523364      0.40872274      0.06604361

# Use Ctrl + Shift + M or Cmd + Shift + M to enter %>%
#
# Pipes take the output of one function and feed it to the first argument of the
# next function.

table(acs12$employment) %>% prop.table() %>% round(2)

##
##      employed not in labor force      unemployed
##      0.53      0.41      0.07

```



```

# Confidence intervals of proportions
#
# These are just estimates. How certain are we? Another sample would yield
# slightly different results.
#
# Confidence intervals help us quantify the uncertainty. They provide an
# estimated lower and upper bound of our estimate.
#
# Example: how certain are we about 0.53 employed?
#
# The prop.test() function returns a 95% confidence interval for an estimated
# proportion.
#
# 95% Confidence Interval theory: sample the data, calculate a confidence
# interval, repeat many times. About 95% of confidence intervals will contain
# the "true" value you're trying to estimate. See Appendix for a demo.
#
# The prop.test() function requires number of "successes" (ie, number employed)
# and total number of "trials" (ie, number of respondents).

# Number of Employed: 843
sum(acs12$employment == "employed", na.rm = TRUE)

```

```
## [1] 843
```

```

# Total number who responded (ie, total not missing): 1605
sum(table(acs12$employment))

```

```
## [1] 1605
```

```

# x = number of "successes", n = number of "trials"
prop.test(x = 843, n = 1605)

```

```

##
## 1-sample proportions test with continuity correction
##
## data: 843 out of 1605
## X-squared = 3.9875, df = 1, p-value = 0.04584
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
## 0.5004608 0.5498845
## sample estimates:
## p
## 0.5252336

```

```

# 95 percent confidence interval: (0.50, 0.55)
#

```

```

# can save into an object
p.out <- prop.test(x = 843, n = 1605)

```

```

# access the confidence interval
p.out$conf.int

```

```

## [1] 0.5004608 0.5498845
## attr(,"conf.level")
## [1] 0.95

```

```

# The mosaic version of prop.test allows us to do the following:
mosaic::prop.test(acs12$employment == "employed")

##
## 1-sample proportions test with continuity correction
##
## data:  acs12$employment == "employed" [with success = TRUE]
## X-squared = 3.9875, df = 1, p-value = 0.04584
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.5004608 0.5498845
## sample estimates:
##          p
## 0.5252336

# Note: placing the package name and two colons before a function tells R to use
# the function in that package.
#
# The prop.test function in the stats package that comes with R cannot do this.
# stats::prop.test(acs12$employment == "employed")
#
# The binom.confint() function from the binom package allows us to get
# confidence intervals for all three employment levels. method = "prop.test"
# returns traditional confidence intervals
binom.confint(x = table(acs12$employment),
              n = sum(table(acs12$employment)),
              method = "prop.test")

##      method  x    n      mean      lower      upper
## 1 prop.test 843 1605 0.5252336 0.50046077 0.54988448
## 2 prop.test 656 1605 0.40872274 0.38461017 0.43327685
## 3 prop.test 106 1605 0.06604361 0.05461547 0.07959697

# The binom.confint() function provides a 11 different methods for calculating
# CIs. # Why use other methods? When proportion estimate is near boundary (ie, 0
# or 1). Notice these are equal out to 3 decimal places.
binom.confint(x = 843, n = 1605)

##      method  x    n      mean      lower      upper
## 1 agresti-coull 843 1605 0.5252336 0.5007722 0.5495745
## 2 asymptotic 843 1605 0.5252336 0.5008035 0.5496638
## 3 bayes 843 1605 0.5252179 0.5007974 0.5496209
## 4 cloglog 843 1605 0.5252336 0.5005007 0.5493391
## 5 exact 843 1605 0.5252336 0.5004647 0.5499102
## 6 logit 843 1605 0.5252336 0.5007625 0.5495842
## 7 probit 843 1605 0.5252336 0.5007713 0.5496012
## 8 profile 843 1605 0.5252336 0.5007778 0.5496090
## 9 lrt 843 1605 0.5252336 0.5007827 0.5496030
## 10 prop.test 843 1605 0.5252336 0.5004608 0.5498845
## 11 wilson 843 1605 0.5252336 0.5007723 0.5495745

# Sometimes we derive proportions from numeric data. Example: proportion of
# people working more than 40 hours
mean(acs12$hrs_work > 40, na.rm = TRUE)

## [1] 0.2387904

```

```

# How certain are we about that proportion?
# The mosaic version of prop.test allows us to do the following:
mosaic::prop.test(acs12$hrs_work > 40)

##
## 1-sample proportions test with continuity correction
##
## data:  acs12$hrs_work > 40 [with success = TRUE]
## X-squared = 260.69, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.2123835 0.2673217
## sample estimates:
##           p
## 0.2387904

# 95 percent confidence interval: (0.21, 0.27)

# YOUR TURN #1 -----

# What proportion of ACS respondents are married? As a population estimate, how
# certain is it? Calculate a confidence interval

# means and medians -----

# The mean and median are two measures of center.

# calling summary() on a numeric variable returns the mean and median
summary(acs12$hrs_work)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
##      1.00  32.00   40.00   37.98  40.00   99.00    1041

# The median is the middle of the sorted data
# The mean is the "balance point" of the data
# Symmetric data have similar means and medians.

# The mean and median functions; if any data is missing the result is NA
mean(acs12$hrs_work)

## [1] NA

median(acs12$hrs_work)

## [1] NA

# specify na.rm = TRUE to ignore missing data
mean(acs12$hrs_work, na.rm = TRUE)

```

```
## [1] 37.97706
```

```
median(acs12$hrs_work, na.rm = TRUE)
```

```
## [1] 40
```

```
# How much data is missing? The is.na() function can help us answer this.
```

```
sum(is.na(acs12$hrs_work)) # count
```

```
## [1] 1041
```

```
mean(is.na(acs12$hrs_work)) # proportion
```

```
## [1] 0.5205
```

```
# How much data is NOT missing? Precede is.na() with !
```

```
sum(!is.na(acs12$hrs_work))
```

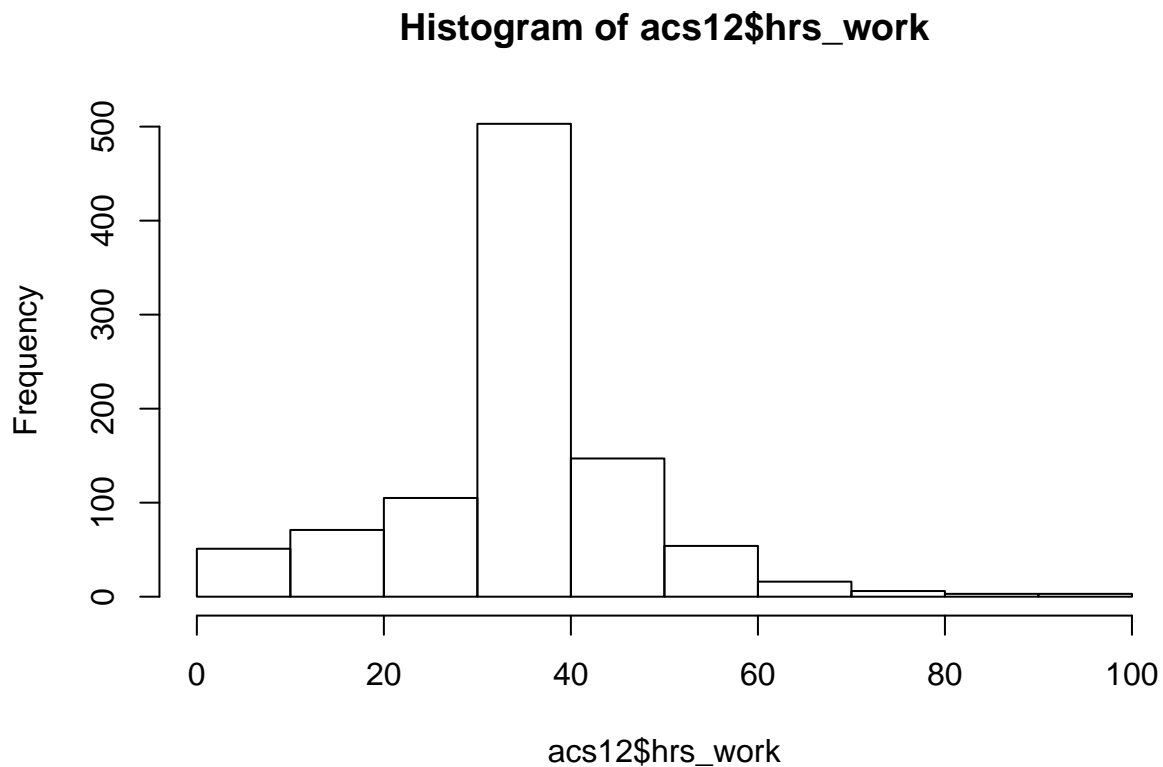
```
## [1] 959
```

```
mean(!is.na(acs12$hrs_work))
```

```
## [1] 0.4795
```

```
# histograms are good to visualize the distribution of numeric variables
```

```
hist(acs12$hrs_work)
```



```
# Numeric measures of spread include the standard deviation...
```

```
sd(acs12$hrs_work, na.rm = TRUE)
```

```
## [1] 13.49768
```

```

# ...the Interquartile Range (difference between 75th and 25th quartiles)
IQR(acs12$hrs_work, na.rm = TRUE)

## [1] 8

# The IQR may be preferred with highly skewed data, but can always report both.

# Recall the estimated mean of hrs_work
mean(acs12$hrs_work, na.rm = TRUE)

## [1] 37.97706

# The mean of 37.977 is just an estimate. How certain are we about the mean?
# Another sample would yield slightly different results. The t.test function
# returns a 95% confidence interval for a mean
t.test(acs12$hrs_work)

##
## One Sample t-test
##
## data:  acs12$hrs_work
## t = 87.131, df = 958, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  37.12170 38.83242
## sample estimates:
## mean of x
##  37.97706

# CI: (37.1, 38.8)

# We can save into an object and access the confidence interval directly
t.out <- t.test(acs12$hrs_work)
t.out$conf.int

## [1] 37.12170 38.83242
## attr(,"conf.level")
## [1] 0.95

# The smean.cl.normal() function from the Hmisc package returns just a
# confidence interval:
Hmisc::smean.cl.normal(acs12$hrs_work)

##      Mean      Lower      Upper
## 37.97706 37.12170 38.83242

# The smedian.hilow() function from the Hmisc package returns a confidence
# interval for the median:
Hmisc::smmedian.hilow(acs12$hrs_work)

## Median Lower Upper
##  40.00   8.00  65.15

# The smean.cl.boot() function the the Hmisc package returns a bootstrapped
# confidence interval of the mean. This may be useful for small samples.
Hmisc::smean.cl.boot(acs12$hrs_work, B = 1500)

##      Mean      Lower      Upper
## 37.97706 37.12291 38.79638

```

```

# Bootstrapping means resampling your data with replacement, calculating a
# statistic of choice (such as a mean) and then repeating many times. The result
# is many means which we then can use to get an estimate of uncertainty of our
# original estimated mean. Because it's based on a resampling, your bootstrapped
# CI will probably be different from mine, but only slightly.
#
# Bootstrapping is effective for estimating uncertainty when the usual
# assumptions for calculating standard errors are suspect, or when a standard
# error formula is complex or not available.
#
# See Appendix below for more information on basic bootstrapping.

# YOUR TURN #2 -----

# (1) What is the mean income of ACS respondents? As a population estimate, how
# certain is it? Calculate a confidence interval.

# (2) How many respondents reported an income of 0?

# (3) What proportion of people reported earning more than $100,000? As a
# population estimate, how certain is it? Calculate a confidence interval.

# Comparing two proportions -----

# We often want to compare two proportions.

# Example: Is there a difference between the percent of people with disabilities
# between citizens and non-citizens?

# cross tabulation of citizenship and disability using table()
table(acs12$citizen, acs12$disability)

##
##      no  yes
## no   105   13
## yes 1571  311

# can also use xtabs(); the table includes row and column labels
# read the "~" as "tabulate the data by"
xtabs(~ citizen + disability, data = acs12)

##      disability
## citizen  no  yes
## no      105  13
## yes    1571 311

# prop.table() calculates proportions; margin = 1 means across rows; margin = 2
# means down the columns
xtabs(~ citizen + disability, data = acs12) %>%

```

```

prop.table(margin = 1) %>%
round(3)

##          disability
## citizen    no    yes
##      no 0.890 0.110
##      yes 0.835 0.165

# About 0.11 of non-citizens have a disability. About 0.17 of citizens have a
# disability. That's a difference of about -0.06. How certain are we about that
# difference? Would another random sample result in a difference in the opposite
# direction?
#
# Can use prop.test() to answer this. The first argument x takes number of
# "successes" for each group (citizens and non-citizens); the second argument n
# takes number of people surveyed in each group (citizens and non-citizens)
prop.test(x = c(13, 311), n = c(105 + 13, 1571 + 311))

##
## 2-sample test for equality of proportions with continuity correction
##
## data: c(13, 311) out of c(105 + 13, 1571 + 311)
## X-squared = 2.0923, df = 1, p-value = 0.148
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.118515145 0.008354659
## sample estimates:
##      prop 1      prop 2
## 0.1101695 0.1652497

# The p-value says there's about a 15% chance of getting a difference this big,
# or bigger, if there really is no difference between the proportions.
#
# Traditionally, we judge a difference "significant", or not due to chance
# alone, if the p-value is small, say below 0.05.
#
# In this case, we might conclude:
#
# - "with the current sample size the data were unable to overcome the
#   supposition of no difference in the proportions"
#
# The 95 percent confidence interval is on the difference of proportions. Since
# it overlaps 0 (barely) we're uncertain about the direction of the difference.
#
#
# To avoid hard-coding numbers, we can use the mosaic version which allows us to
# use formula notation to specify we want to compare the proportion of people
# with disabilities by citizen. Notice we need to specify success = "yes", which
# means we want to compare proportion of people who answered "yes" to
# disability.
mosaic::prop.test(disability ~ citizen, data = acs12, success = "yes")

##
## 2-sample test for equality of proportions with continuity correction
##

```

```
## data:  tally(disability ~ citizen)
## X-squared = 2.0923, df = 1, p-value = 0.148
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.118515145  0.008354659
## sample estimates:
##      prop 1      prop 2
## 0.1101695 0.1652497
```

```
# A slightly more complicated example...
#
# cross tabulation of citizenship and education
xtabs(~ citizen + edu, data = acs12)
```

```
##          edu
## citizen college grad hs or lower
##    no         15      8          94
##    yes        344    136         1345
```

```
# proportion of education level by citizenship
xtabs(~ citizen + edu, data = acs12) %>%
  prop.table(margin = 1) %>%
  round(3)
```

```
##          edu
## citizen college grad hs or lower
##    no    0.128 0.068      0.803
##    yes   0.188 0.075      0.737
```

```
# About 0.13 of non-citizens are college graduates. About 0.19 of citizens are
# college graduates. That's a difference of -0.06. How certain are we about that
# difference? Would another random sample result in a difference in the opposite
# direction?
#
```

```
# We can use prop.test() to answer this. The first argument x takes number of
# "successes"; the second argument n takes number of trials
```

```
# Number of success: 15, 344
xtabs(~ citizen + edu, data = acs12)
```

```
##          edu
## citizen college grad hs or lower
##    no         15      8          94
##    yes        344    136         1345
```

```
# Number of trials: 117, 1825
xtabs(~ citizen + edu, data = acs12) %>%
  addmargins(margin = 2)
```

```
##          edu
## citizen college grad hs or lower Sum
##    no         15      8          94    117
##    yes        344    136         1345  1825
```

```
# Use values with prop.test()
prop.test(x = c(15, 344), n = c(117, 1825))
```

```
##
```



```
## 2-sample test for equality of proportions with continuity correction
##
## data:  c(15, 344) out of c(117, 1825)
## X-squared = 2.2671, df = 1, p-value = 0.1321
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.128015161  0.007439116
## sample estimates:
##      prop 1      prop 2
## 0.1282051 0.1884932

# The p-value says there's about a 13% chance of getting a difference this big,
# or bigger, if there really is no difference between the proportions.
#
# The confidence level of the difference in proportions barely overlaps 0.
#
# To avoid entering numbers we can also save table with margins and use with
# prop.test
tab <- xtabs(~ citizen + edu, data = acs12) %>%
  addmargins(margin = 2)
tab

##           edu
## citizen college grad hs or lower Sum
##    no      15      8          94 117
##    yes     344    136         1345 1825

tab[, "college"]

##    no yes
##    15 344

tab[, "Sum"]

##    no yes
##   117 1825

prop.test(x = tab[, "college"], n = tab[, "Sum"])

##
## 2-sample test for equality of proportions with continuity correction
##
## data:  tab[, "college"] out of tab[, "Sum"]
## X-squared = 2.2671, df = 1, p-value = 0.1321
## alternative hypothesis: two.sided
## 95 percent confidence interval:
## -0.128015161  0.007439116
## sample estimates:
##      prop 1      prop 2
## 0.1282051 0.1884932

# YOUR TURN #3 -----

# Compare the proportion of people with disabilities between male and female
# (gender). What is the confidence interval of the difference of proportions?
```

```

# Comparing two means -----

# We often want to compare means between two groups.

# Example: does mean income differ between gender?

# The aggregate() function allows us to calculate a specified statistic for
# groups.

# Example: Take income, group by gender, and calculate mean for each group
aggregate(income ~ gender, data = acs12, mean)

##   gender   income
## 1 female 14335.99
## 2  male 32627.30

# The mosaic version of mean allows a formula and "grouping" variable
mosaic::mean(~ income | gender, data = acs12, na.rm = TRUE)

##   female      male
## 14335.99 32627.30

# The mean income of females is about $14,335. The mean income of males is about
# $32,627. That's a difference of -$18,292. How certain are we about that
# difference? Would another random sample result in a difference in the opposite
# direction?

# The t.test() function can help us assess the difference.
t.test(income ~ gender, data = acs12)

##
## Welch Two Sample t-test
##
## data: income by gender
## t = -8.1362, df = 1142.1, p-value = 1.056e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -22702.24 -13880.38
## sample estimates:
## mean in group female mean in group male
## 14335.99 32627.30

# The 95% CI on the difference is (-22702.24, -13880.38)
#
# The t-test is testing that the data from the two groups came from the same
# Normal distribution. The t-test is pretty robust to this assumption with large
# sample sizes, say greater than 30. In other words, we can trust the results
# even if the Normality assumption is suspect.
#
# The p-value is virtually 0. There is just about no chance we would see a
# difference in means this large or larger if the two groups came from the same
# normally distributed population.
#
# Save to an object and extract the CI of the difference in means
t.out <- t.test(income ~ gender, data = acs12)

```

```

t.out$conf.int

## [1] -22702.24 -13880.38
## attr(,"conf.level")
## [1] 0.95

# It's time for a closer look at the data:
# Proportion of zeros
mean(acs12$income == 0, na.rm=TRUE)

## [1] 0.4491682

# Proportion of zeros by gender
table(acs12$gender, acs12$income == 0) %>%
  prop.table(margin = 1) %>%
  round(2)

##
##          FALSE TRUE
##   female  0.51 0.49
##    male   0.59 0.41

# Maybe we should drop the zeroes? We can use the subset() function to only use
# rows for which income is greater than 0.
t.out <- t.test(income ~ gender, data = subset(acs12, income > 0))
t.out

##
## Welch Two Sample t-test
##
## data: income by gender
## t = -7.9637, df = 700.29, p-value = 6.754e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -34161.16 -20648.53
## sample estimates:
## mean in group female    mean in group male
##          28007.63          55412.48

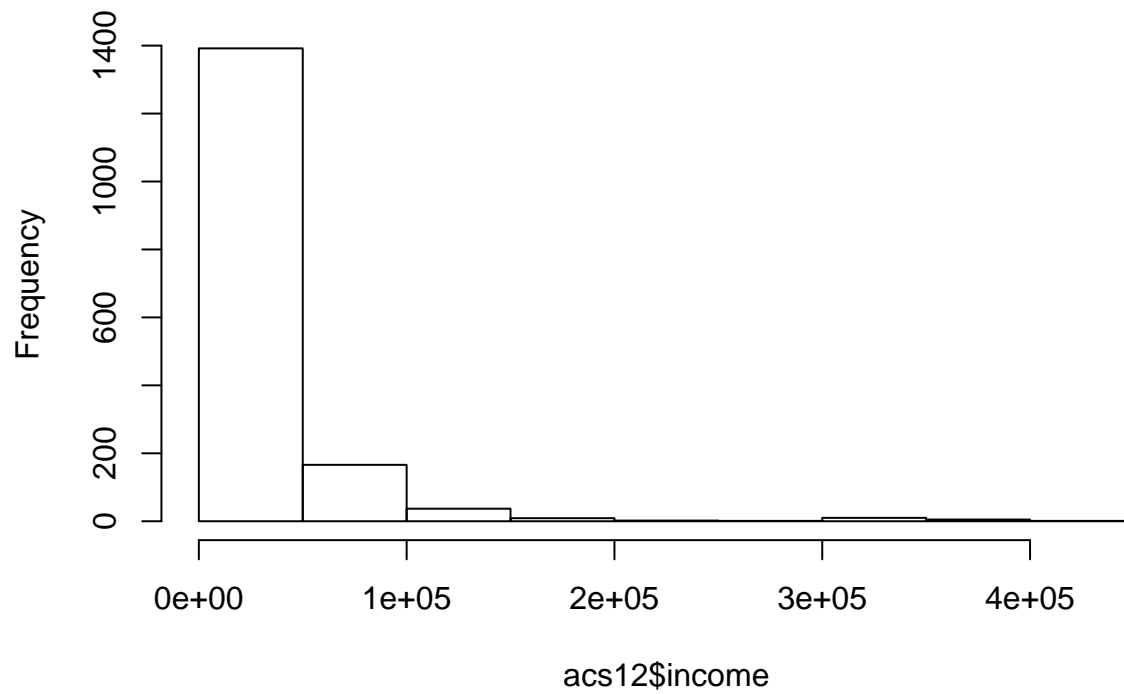
t.out$conf.int

## [1] -34161.16 -20648.53
## attr(,"conf.level")
## [1] 0.95

# The expected income difference for females is about (-$34,161, -$20,648) less
# than males.
#
# It's important to note the data is quite skew.
hist(acs12$income)

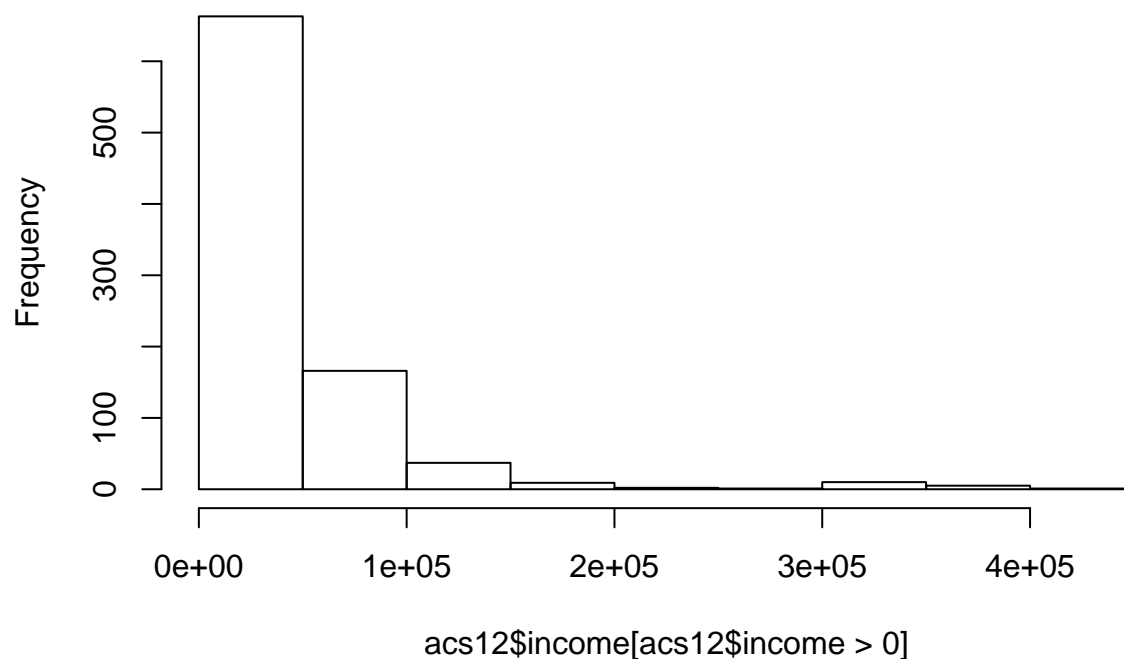
```

**Histogram of acs12\$income**



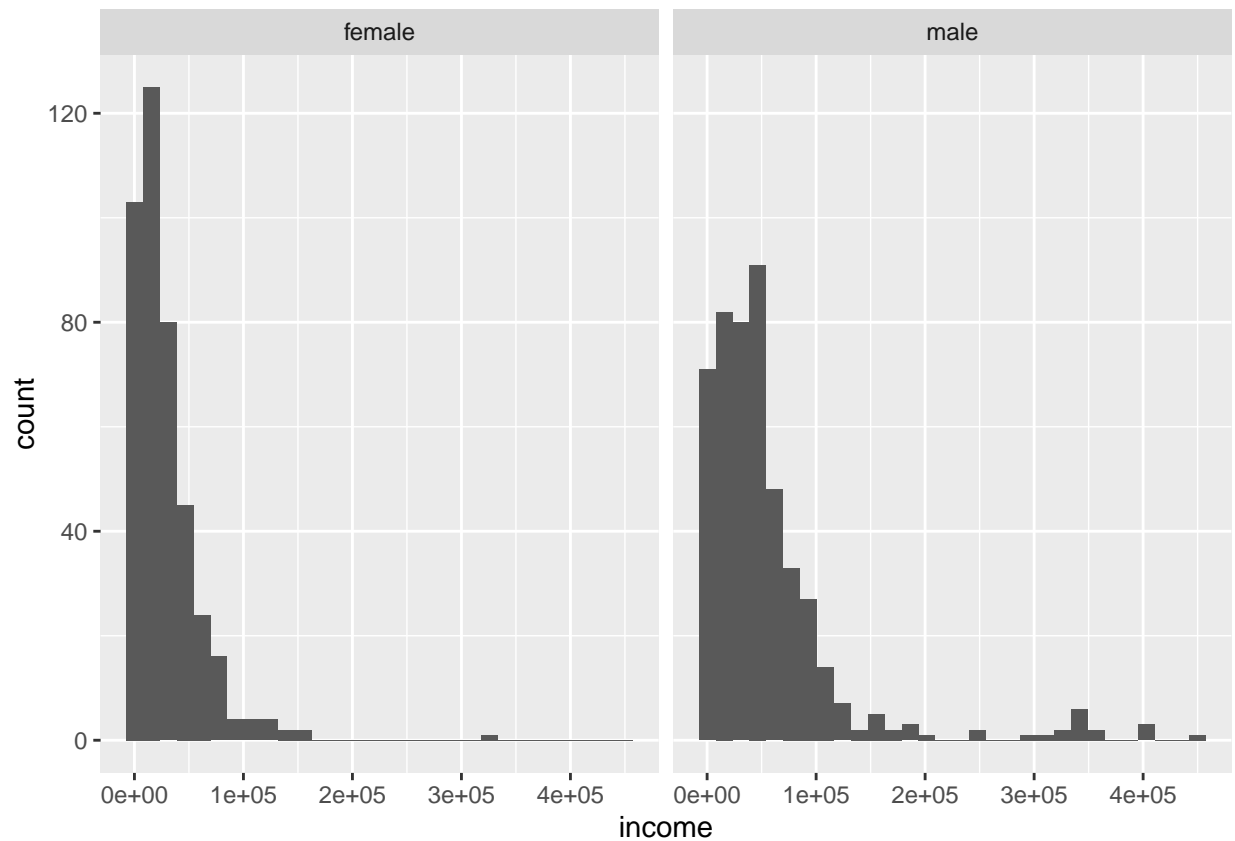
```
hist(acs12$income[acs12$income > 0])
```

**Histogram of acs12\$income[acs12\$income > 0]**



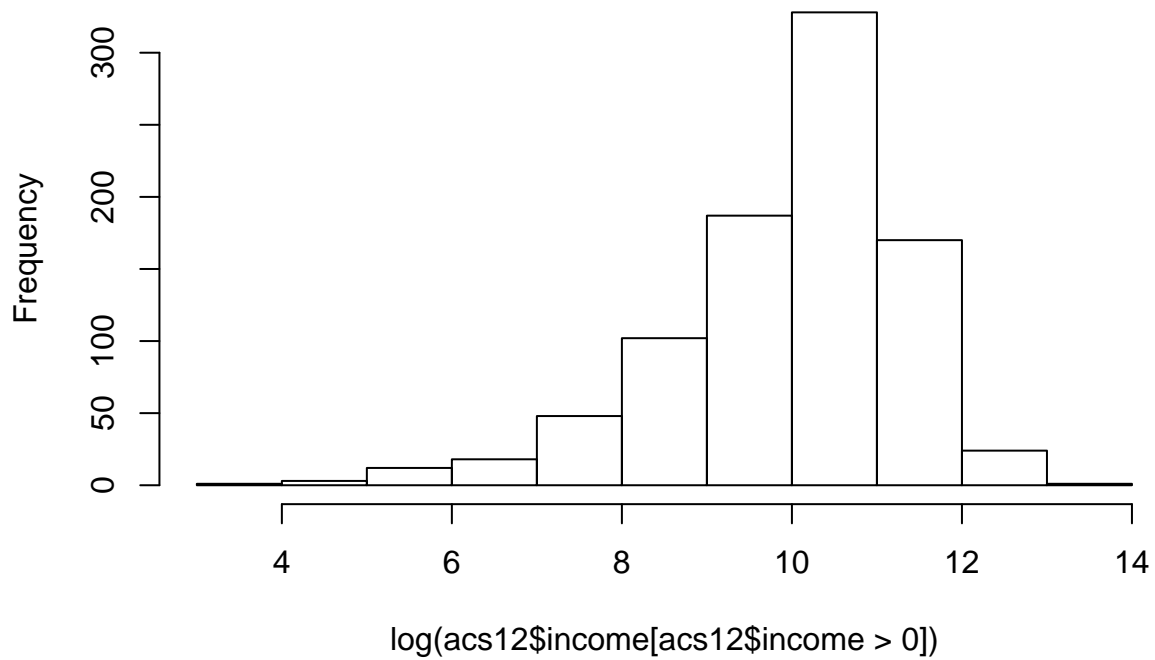
```
# ggplot makes it pretty easy to see the histograms for both genders
ggplot(subset(acs12, income > 0), aes(x = income)) +
  geom_histogram() +
  facet_wrap(~gender)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



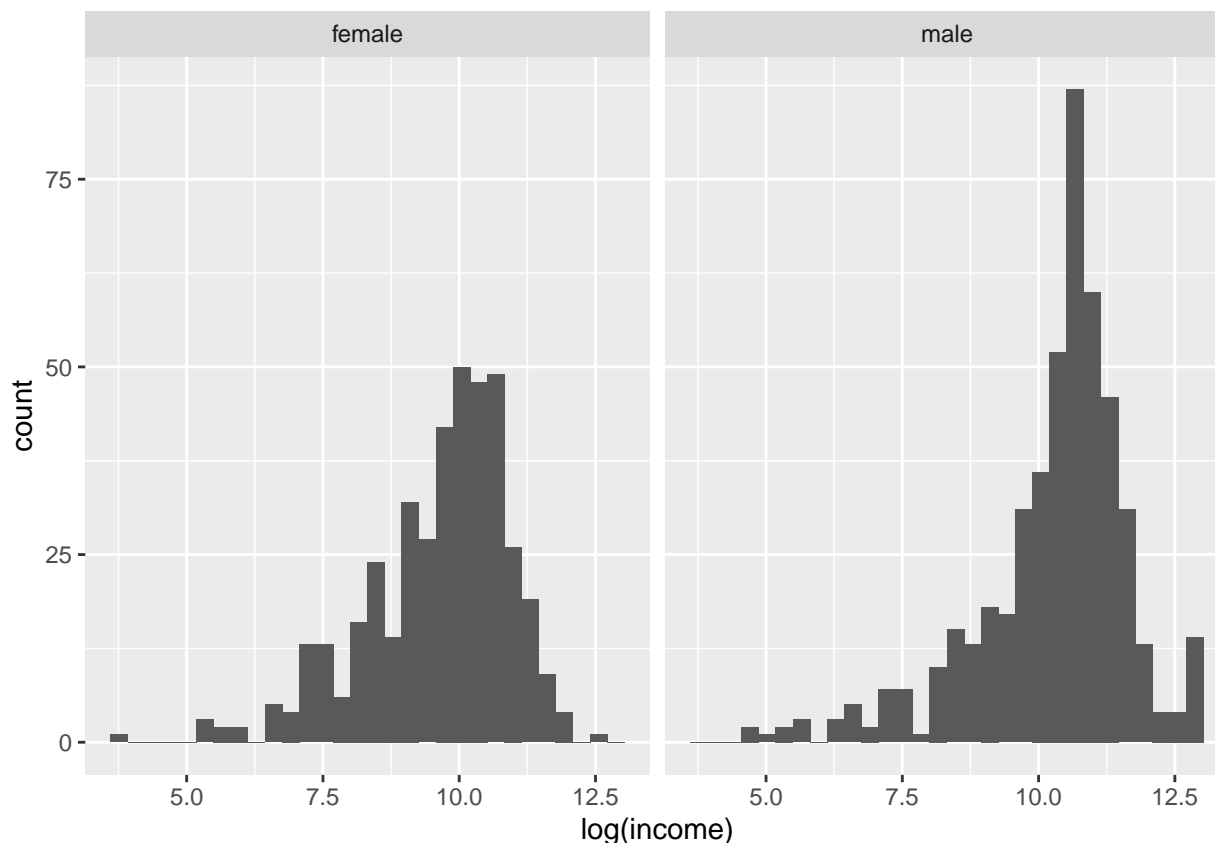
```
# One alternative approach might be a log transformation. This is useful for
# skewed data such as income. NOTE: you can only log transform values greater
# than 0
#
# after log transformation; natural log (base e)
hist(log(acs12$income[acs12$income > 0]))
```

## Histogram of $\log(\text{acs12\$income}[\text{acs12\$income} > 0])$



```
# for both genders  
ggplot(subset(acs12, income > 0), aes(x = log(income))) +  
  geom_histogram() +  
  facet_wrap(~gender)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# Now skewed the other direction...
```

```
#
```

```
# t-test for difference in log-transformed means
```

```
t.test(log(income) ~ gender, data = subset(acs12, income > 0))
```

```
##
```

```
## Welch Two Sample t-test
```

```
##
```

```
## data: log(income) by gender
```

```
## t = -7.1765, df = 879.47, p-value = 1.52e-12
```

```
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
```

```
## -0.8333371 -0.4754139
```

```
## sample estimates:
```

```
## mean in group female mean in group male
```

```
## 9.613135 10.267511
```

```
# The result is still quite large and significant, but difficult to interpret. We
```

```
# need to exponentiate to get data on original scale. First let's save the
```

```
# result.
```

```
t.out <- t.test(log(income) ~ gender, data = subset(acs12, income > 0))
```

```
t.out
```

```
##
```

```
## Welch Two Sample t-test
```

```
##
```

```
## data: log(income) by gender
```



```

## t = -7.1765, df = 879.47, p-value = 1.52e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.8333371 -0.4754139
## sample estimates:
## mean in group female    mean in group male
##          9.613135          10.267511

t.out$estimate

## mean in group female    mean in group male
##          9.613135          10.267511

# Now exponentiate the estimated means
exp(t.out$estimate)

## mean in group female    mean in group male
##          14960.00          28782.15

# exponentiating the confidence interval returns the multiplicative effect of
# gender on income.
t.out$conf.int

## [1] -0.8333371 -0.4754139
## attr("conf.level")
## [1] 0.95

exp(t.out$conf.int)

## [1] 0.4345966 0.6216277
## attr("conf.level")
## [1] 0.95

# females appear to earn about 0.43 to 0.62 times the income of males
#
# If you doubt the normality assumption, you can try a nonparametric test such
# as the wilcox.test. Instead of assuming the two means come from the same
# Normal distribution, we assume the two means just come from the same
# distribution.
wilcox.test(income ~ gender, data = subset(acs12, income > 0))

##
## Wilcoxon rank sum test with continuity correction
##
## data: income by gender
## W = 66680, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
# To get a confidence interval, set conf.int = TRUE
wilcox.test(income ~ gender, data = subset(acs12, income > 0),
            conf.int = TRUE)

##
## Wilcoxon rank sum test with continuity correction
##
## data: income by gender
## W = 66680, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:

```

```
## -20000 -12000
## sample estimates:
## difference in location
## -16000

# The "difference in location" estimates the median of the difference between a
# sample from x and a sample from y.

# YOUR TURN #4 -----

# How does mean hrs_work differ between gender? What is the confidence interval
# of the difference?

# Comparing more than 2 means -----

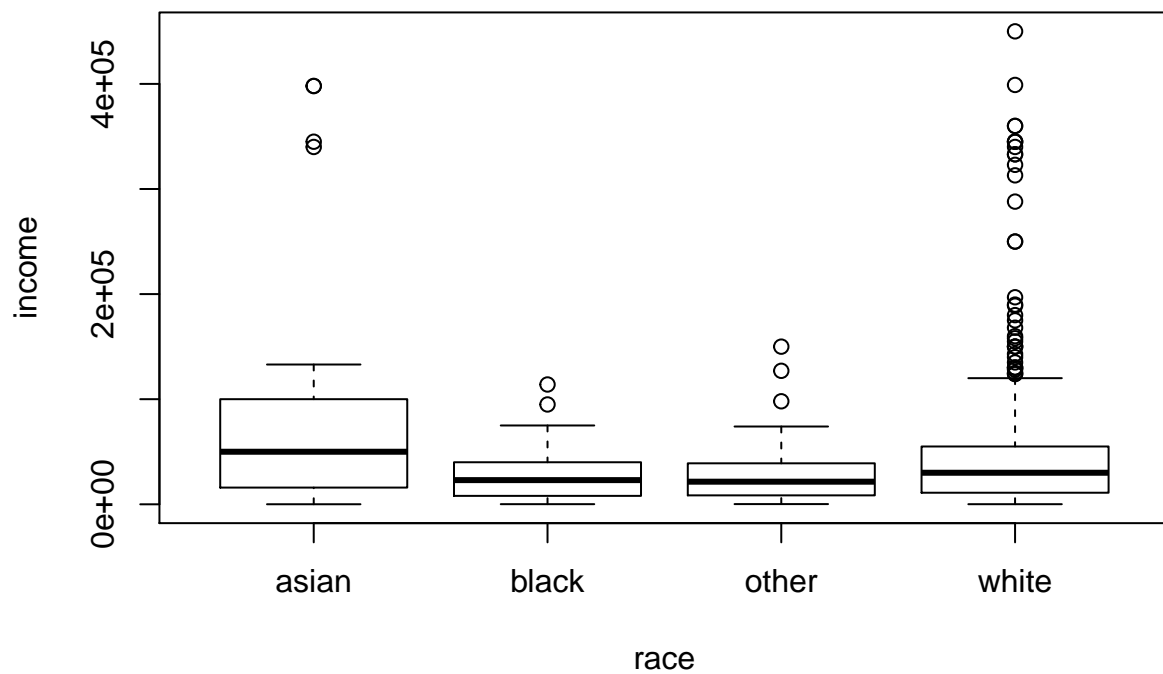
# We often want to compare means between more than two groups.
#
# Example: does mean income differ between race?
#
# Use aggregate to calculate mean income for each race. Let's drop the rows with
# 0 income.
aggregate(income ~ race, data = subset(acs12, income > 0), mean)

##    race    income
## 1 asian  79338.37
## 2 black  28003.84
## 3 other  28098.44
## 4 white  43772.60

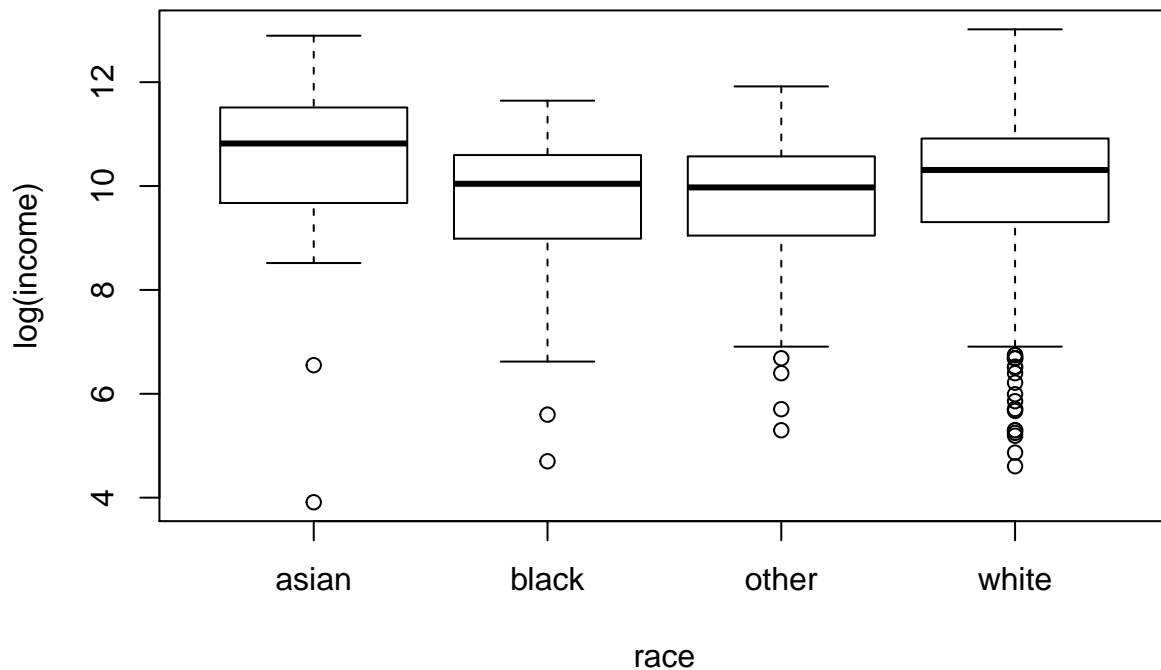
# Can also calculate median
aggregate(income ~ race, data = subset(acs12, income > 0), median)

##    race    income
## 1 asian   50000
## 2 black   23000
## 3 other   21500
## 4 white   30000

# boxplots are good for visualizing distribution of numeric variables by a
# grouping variable. The black bar is the median (not the mean). The "box" is
# the middle 50% of the data.
boxplot(income ~ race, data = subset(acs12, income > 0))
```



```
# We can also easily plot log(income)  
boxplot(log(income) ~ race, data = subset(acs12, income > 0))
```



```
# It appears mean income may be different between races. Is this due to chance?
#
# The ANOVA procedure is usually employed to determine if there is a
# statistically significant difference between the means.
#
# The aov() function works like the t.test() function. Best to save result and
# use summary()
aov.out <- aov(log(income) ~ race, data = subset(acs12, income > 0))
summary(aov.out)
```

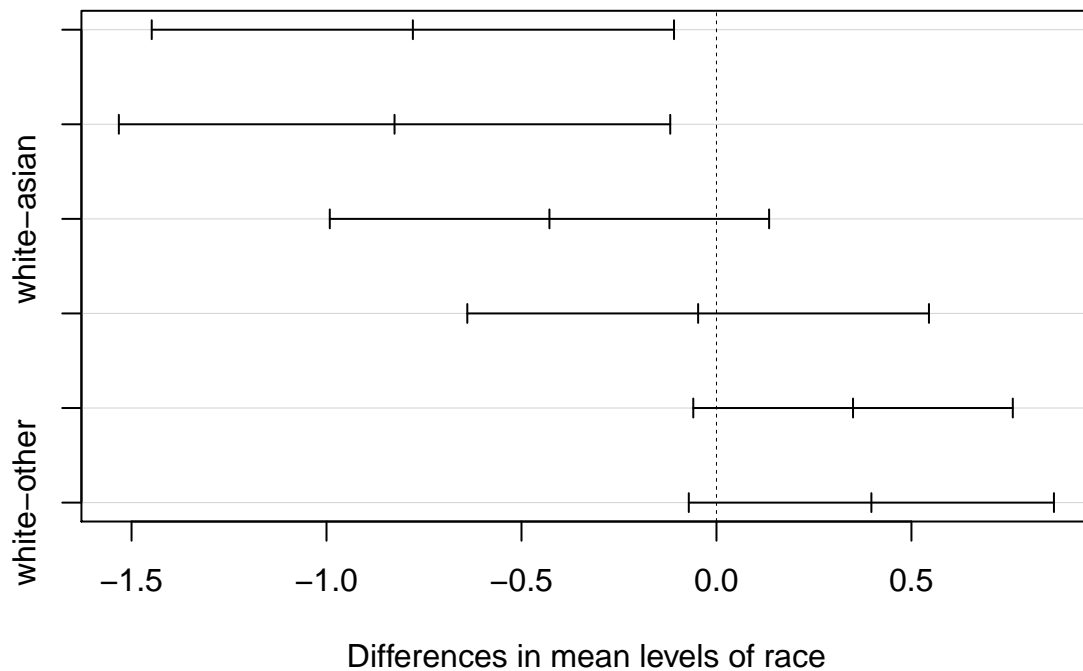
```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## race          3      27    8.998   4.637 0.00317 **
## Residuals    890    1727    1.941
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# A small p-value provides against the null hypothesis that all means are the
# same. So where are the differences? This leads to what are commonly called
# "post-hoc" procedures.
#
# A basic one built into R is Tukey's Honestly Significantly Difference,
# TukeyHSD(). It compares all combinations of pairs and tests whether the
# difference is 0. It also reports 95% confidence intervals on the differences.
#
# Call it on the aov object and save. The result is a list of all pairwise
# comparisons along with the estimated difference and confidence intervals and
# an "adjusted" p-value on the "significance" of the difference.
```

```
tukey.out <- TukeyHSD(aov.out)
tukey.out
```

```
## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = log(income) ~ race, data = subset(acs12, income > 0))
##
## $race
##              diff          lwr          upr      p adj
## black-asian -0.77851761 -1.44819702 -0.1088382 0.0150733
## other-asian -0.82553898 -1.53254517 -0.1185328 0.0144690
## white-asian -0.42829314 -0.99160485  0.1350186 0.2052936
## other-black -0.04702137 -0.63893994  0.5448972 0.9969750
## white-black  0.35022447 -0.05944602  0.7598950 0.1238799
## white-other   0.39724584 -0.07096048  0.8654522 0.1285042
# There is a plotting method for a TukeyHSD object.
plot(tukey.out)
```

### 95% family-wise confidence level



```
# Exponentiate to get multiplicative effect
exp(tukey.out$race[, "diff"])
```

```
## black-asian other-asian white-asian other-black white-black white-other
## 0.4590861 0.4379989 0.6516204 0.9540670 1.4193861 1.4877216
# race=black makes about 46 cents for every dollar race=asian earns.
```

```
# YOUR TURN #5 -----

# Run an ANOVA of log(income) on edu. Is there a difference in mean log(income)
# values between different levels of education? Go ahead subset acs12 to use
# income greater than 0.

# Association between categorical variables -----

# Let's create a cross tabulation of race and employment
xtabs(~ race + employment, data = acs12)

##           employment
## race   employed not in labor force unemployed
##   asian      39             31             3
##   black      76             66            20
##   other      58             39            11
##   white     670            520            72

# proportion of employment by race
xtabs(~ race + employment, data = acs12) %>%
  prop.table(margin = 1) %>%
  round(3)

##           employment
## race   employed not in labor force unemployed
##   asian    0.534             0.425            0.041
##   black    0.469             0.407            0.123
##   other    0.537             0.361            0.102
##   white    0.531             0.412            0.057

# Is there an association between race and employment? Does knowing race give us
# additional information about the proportions of employment levels?
#
# A common approach to answering this question is the chi-square test. The
# chisq.test() function simply requires a table
xtabs(~ race + employment, data = acs12) %>%
  chisq.test()

## Warning in chisq.test(.): Chi-squared approximation may be incorrect
##
## Pearson's Chi-squared test
##
## data:  .
## X-squared = 14.182, df = 6, p-value = 0.02767

# A small p-value provides evidence against the null hypothesis of no
# association. However it does not tell us where the association is or the
# magnitude of the association.
#
# Also notice the warning. That is due to small expected cell counts. Happens
```

```

# when the expected count of a cell is less than 5. Set simulate.p.value = TRUE
# to simulate a p-value using Monte Carlo simulation and not get warning.
c.out <- xtabs(~ race + employment, data = acs12) %>%
  chisq.test(simulate.p.value = TRUE)
c.out

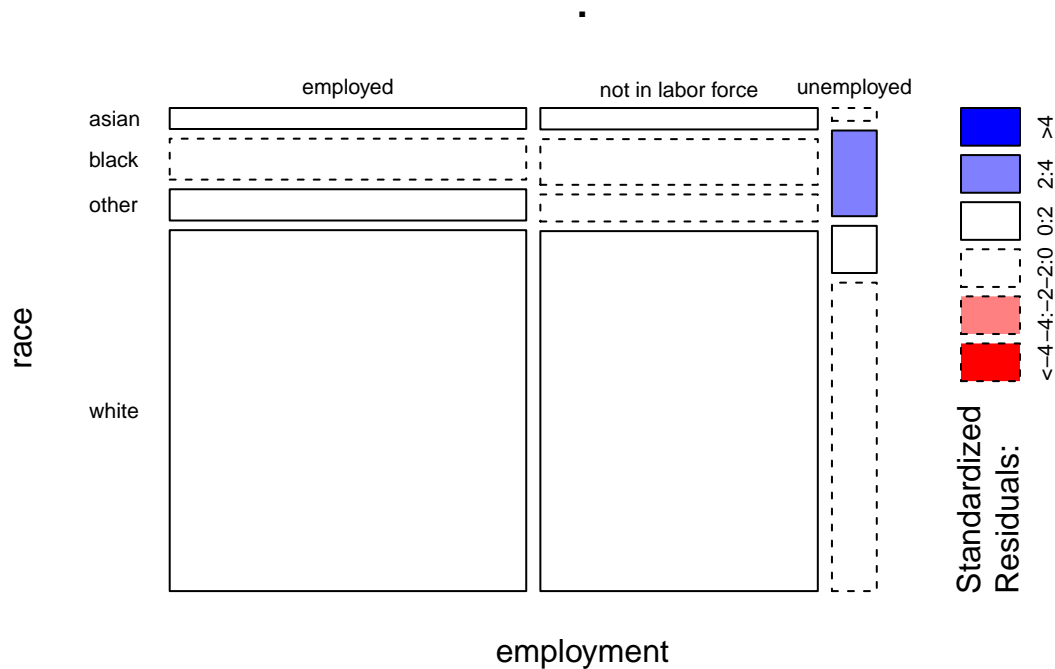
##
## Pearson's Chi-squared test with simulated p-value (based on 2000
## replicates)
##
## data: .
## X-squared = 14.182, df = NA, p-value = 0.02499

# Check the residuals of the chisq.test to see where the "unexpected" high/low
# counts are occurring. Residuals over 2 (or less than -2) are usually of
# interest.
c.out$residuals

##      employment
## race      employed not in labor force unemployed
##   asian  0.1062554          0.2129578 -0.8294246
##   black -0.9852068          -0.0261866  2.8435029
##   other  0.1692554          -0.7739458  1.4480362
##   white  0.2779151          0.1845724 -1.2429038

# It appears we have more unemployed for race=black than would be expected if
# there was no association between race and employment
#
# The mosaicplot() function with base R can help visualize residuals from a
# chi-square test. shade = TRUE colors the tiles where the residuals are
# unusually large. las=1 makes the labels horizontal.
xtabs(~ employment + race, data = acs12) %>%
  mosaicplot(shade = TRUE, las = 1)

```



# The width of the boxes corresponds to the proportions of employment. The  
 # height of the boxes correspond to the proportion of race, within each level of  
 # employment. Within unemployed, we see a higher than expected proportion of  
 # race=black.

# YOUR TURN #6 -----

# (1) Create a cross tabulation of employment and education

# (2) Run a chi-square test on the cross tabulation. Is there evidence of  
 # association?

# (3) Create a mosaic plot of the cross tabulation with residual shading

# Association between numeric variables -----

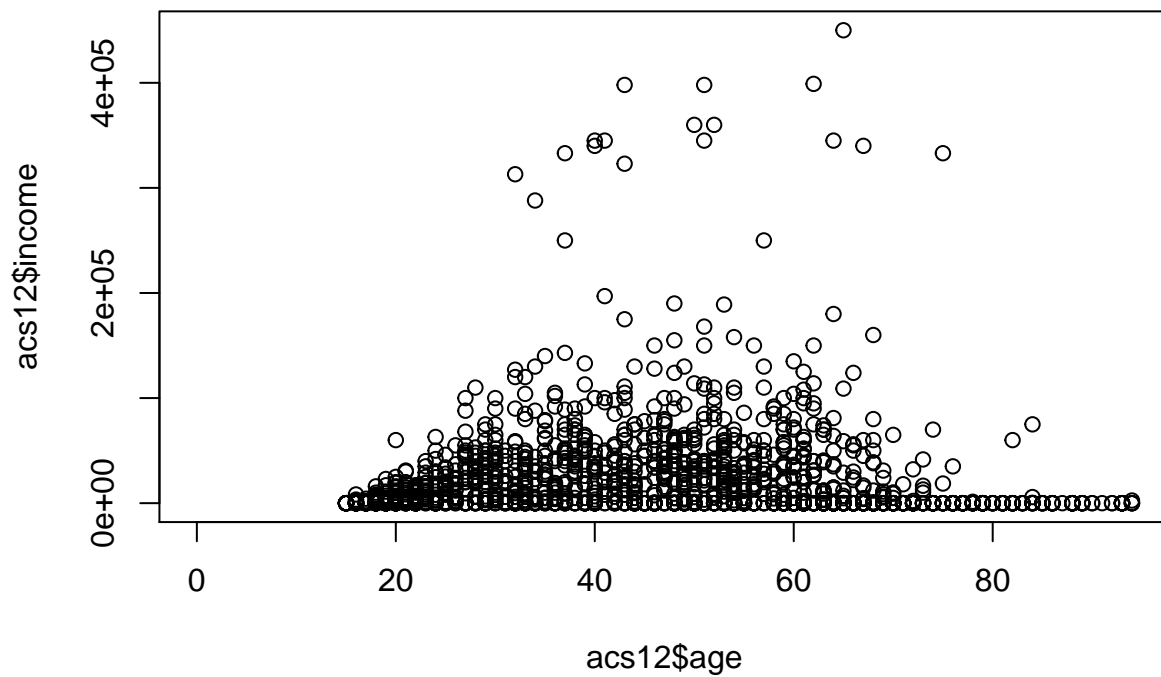
# Correlation summarizes linear association between two numeric variables.  
 # Ranges from -1 to 1. No correlation is 0.

#

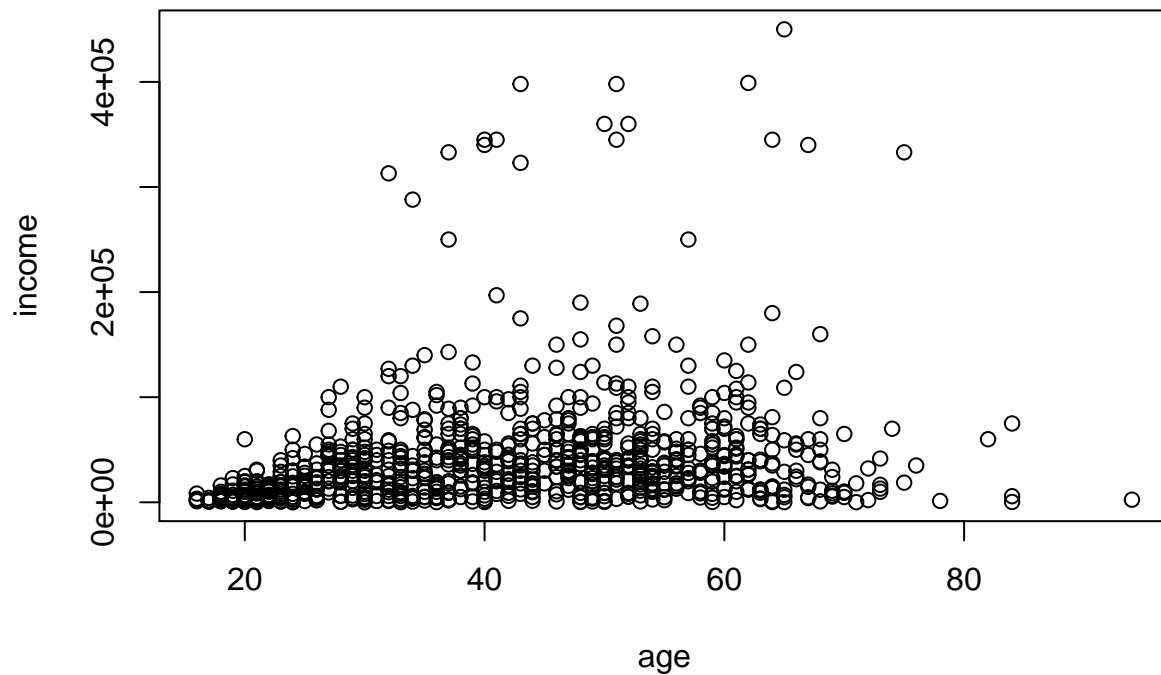
# See an illustration:



```
# https://en.wikipedia.org/wiki/Correlation\_and\_dependence#/media/File:Correlation\_examples2.svg
#
# Is age correlated with income?
#
# Let's first plot age versus income
plot(acs12$age, acs12$income)
```



```
# Plot age versus income for income > 0
plot(income ~ age, data = subset(acs12, income > 0))
```



```
# Correlation between age and income; returns NA because we have missing data.
cor(acs12$age, acs12$income)
```

```
## [1] NA
```

```
summary(acs12$income)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
##         0         0   3000   23600   33700   450000     377
```

```
# Set use = "pairwise.complete.obs" to calculate correlation using all complete
# pairs of observations
```

```
cor(acs12$age, acs12$income, use = "pairwise.complete.obs")
```

```
## [1] -0.03462251
```

```
# The mosaic version allows us to use the formula interface which allows us to
# subset acs12 for income > 0
```

```
mosaic::cor(age ~ income, use = "pairwise.complete.obs",
             data = subset(acs12, income > 0))
```

```
## [1] 0.2133743
```

```
# The cor.test function returns a confidence interval on the correlation
```

```
mosaic::cor.test(age ~ income, use = "pairwise.complete.obs",
                  data = subset(acs12, income > 0))
```

```
##
```

```
## Pearson's product-moment correlation
```

```
##
```

```

## data: age and income
## t = 6.5229, df = 892, p-value = 1.154e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.1499045 0.2750927
## sample estimates:
## cor
## 0.2133743

# YOUR TURN #7 -----

# Calculate the correlation, and associated 95% confidence interval, between age
# and hrs_work. Make a scatterplot as well.

# Simple Linear Regression -----

# Simple linear regression is basically summarizing the relationship between two
# variables as a straight line, using the familiar slope-intercept formula:
#
#  $y = a + bx$ 
#
# This implies we can approximate the mean of  $y$  for a given value of  $x$  by
# multiplying  $x$  by some number and adding a constant value.
#
# It allows us to answer the question, "how does the mean of  $y$  change as  $x$ 
# increases?"

# Example: how does mean income change as hrs_work increases? Regress income on
# hrs_work.
mod <- lm(income ~ hrs_work, data = acs12, subset = income > 0)
summary(mod)

##
## Call:
## lm(formula = income ~ hrs_work, data = acs12, subset = income >
## 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -130100  -23763  -10526    7092   382210
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13846.5     5311.9  -2.607  0.00929 **
## hrs_work      1484.3       131.3  11.301 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52220 on 892 degrees of freedom
## (377 observations deleted due to missingness)
## Multiple R-squared:  0.1252, Adjusted R-squared:  0.1243
## F-statistic: 127.7 on 1 and 892 DF, p-value: < 2.2e-16

```

```

# It appears mean annual income increases by about $1400 for each additional
# hour worked.
#
# The coefficient is just an estimate. Use confint() to get a confidence
# interval. 95% CI: ($1226, $1742)
confint(mod)

```

```

##              2.5 %    97.5 %
## (Intercept) -24271.75 -3421.341
## hrs_work      1226.52  1742.089

```

```

# Is this a "good" estimate? One way to judge is to plot the fitted straight
# line over the raw data.

```

```

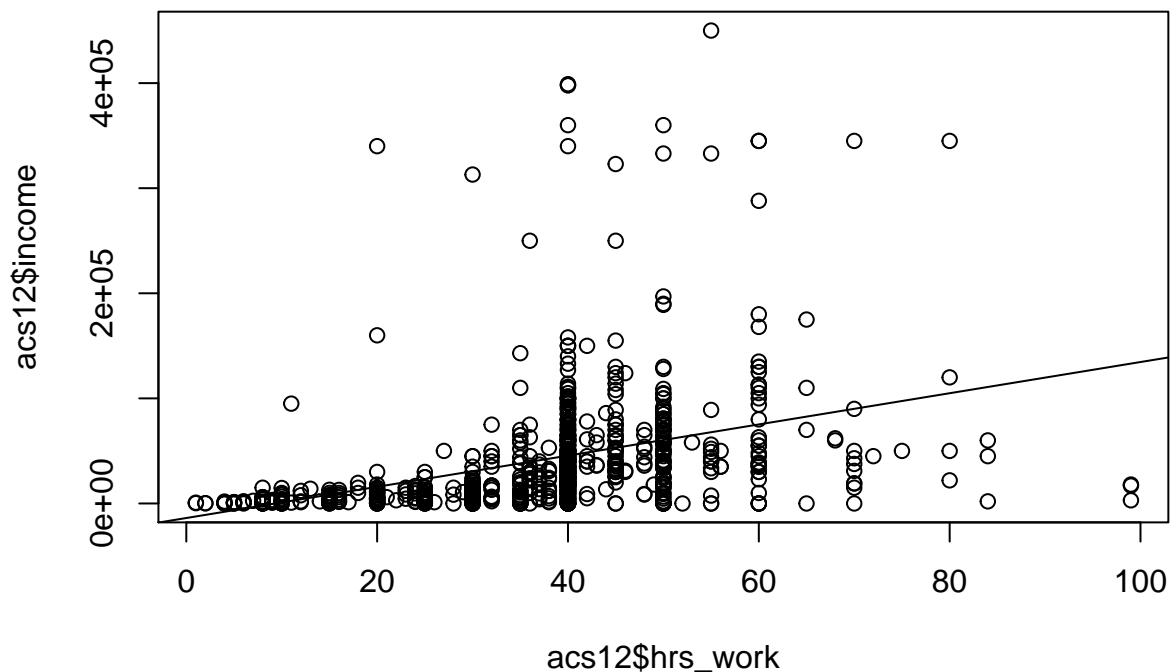
plot(acs12$hrs_work, acs12$income)

```

```

# add fitted straight line using the abline() function
abline(mod)

```

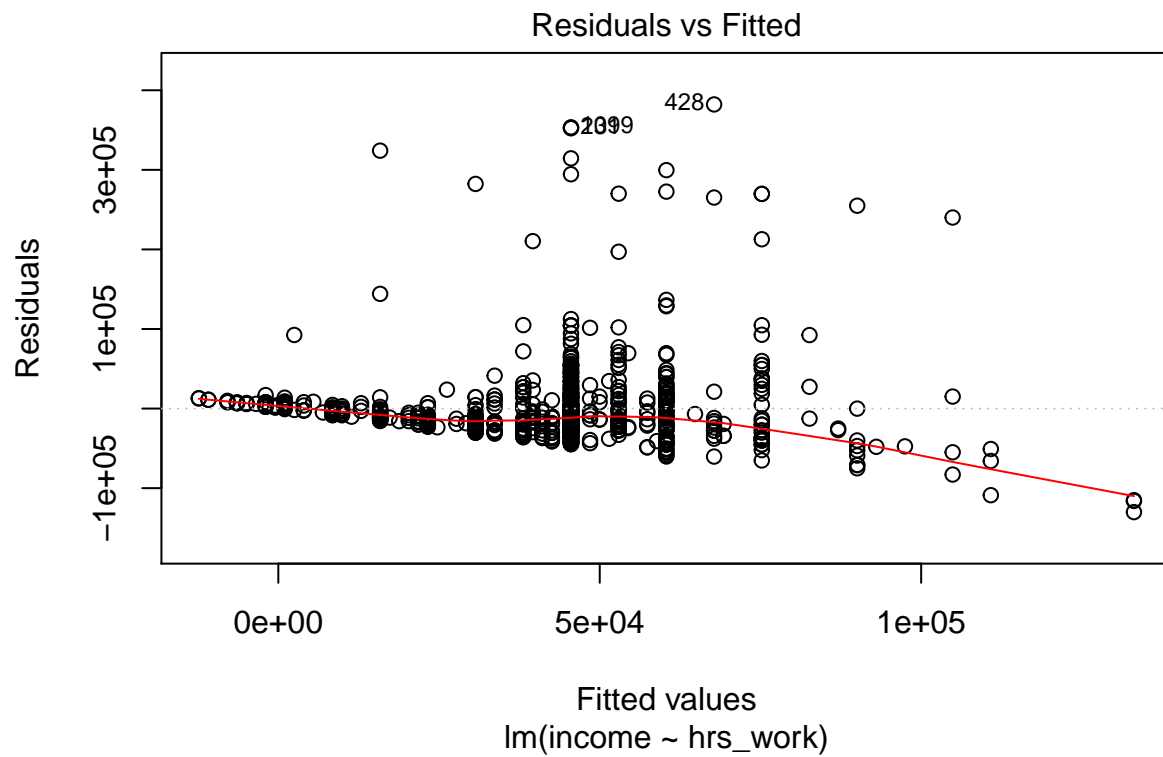


```

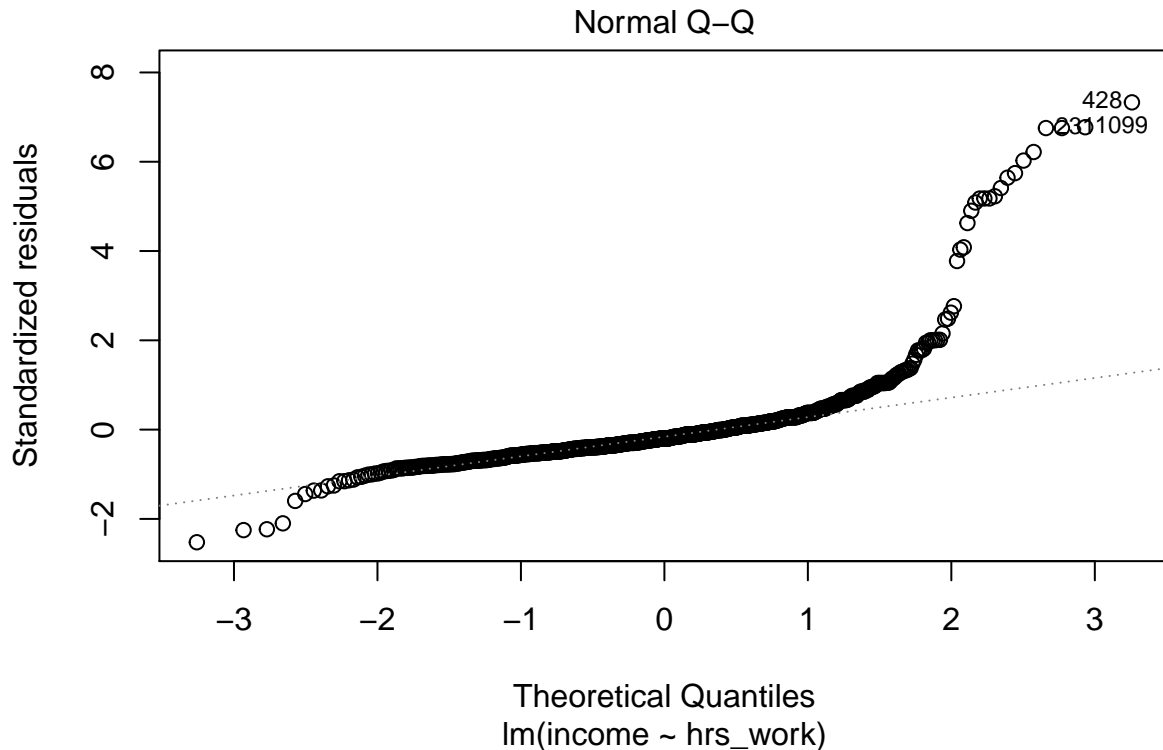
# It doesn't look like a great estimate. The fitted line overpredicts and
# underpredicts over the range of hrs_work.
#
# The differences between the fitted straight line and observed data are called
# the residuals. An assumption of a simple linear model is that these residuals
# are a random sample from a normal distribution with mean 0 and some finite
# standard deviation. This implies we expect an even scatter of raw data around
# the fitted line (ie, constant variance). We can check this assumption by
# calling plot() on the model object.

```

```
#
# The first plot assess the constant variance assumption. The second
# assesses the Normality assumption.
plot(mod, which = 1)
```



```
plot(mod, which = 2)
```



*# Neither look very good. It appears the constant variance assumption is violated. We like to see a mostly horizontal red line in the first plot, and a mostly diagonal line in the second plot.*

*#*

*# One approach to address this is to try a log transformation*

```
mod2 <- lm(log(income) ~ hrs_work, data = acs12, subset = income > 0)
```

*# get estimated coefficients*

```
summary(mod2)
```

```
##
```

```
## Call:
```

```
## lm(formula = log(income) ~ hrs_work, data = acs12, subset = income > 0)
```

```
##
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
```

```
## -5.8446 -0.5067  0.1663  0.6880  3.9169
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  7.558321   0.114263   66.15  <2e-16 ***
```

```
## hrs_work     0.063076   0.002825   22.32  <2e-16 ***
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
```

```
## Residual standard error: 1.123 on 892 degrees of freedom
## (377 observations deleted due to missingness)
## Multiple R-squared:  0.3585, Adjusted R-squared:  0.3577
## F-statistic: 498.4 on 1 and 892 DF,  p-value: < 2.2e-16

# For every extra hour worked, income increases by about (exp(0.063) - 1) * 100
# = 6.5 percent.
exp(0.063)

## [1] 1.065027

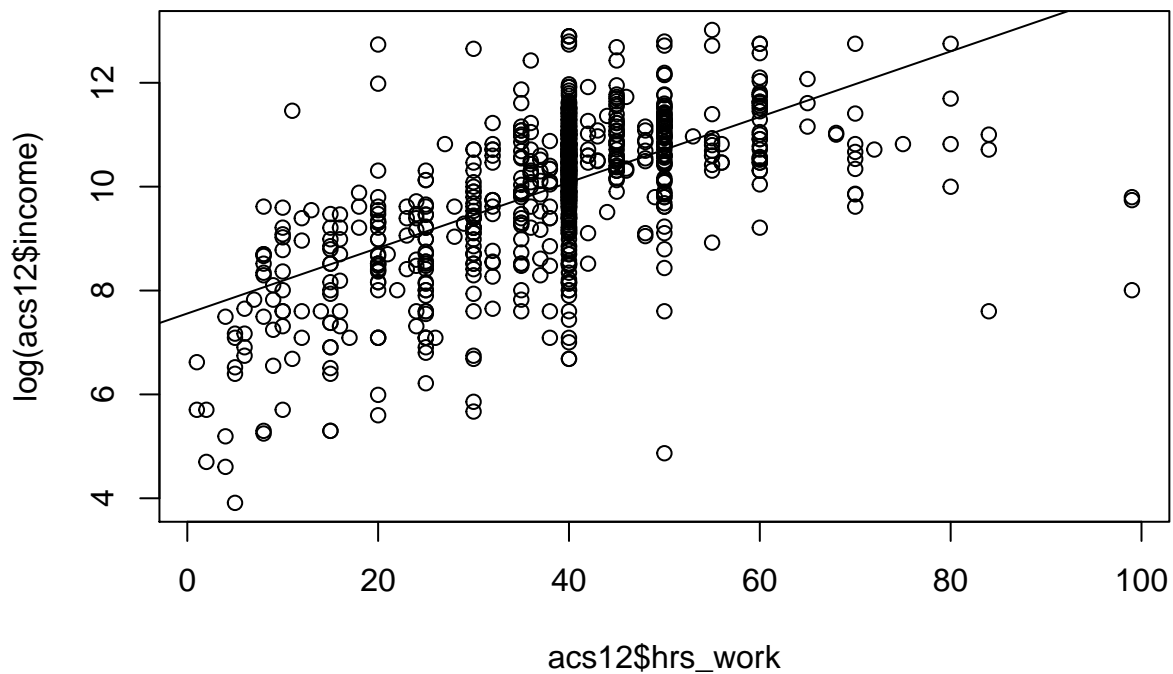
(exp(0.063) - 1) * 100

## [1] 6.502684

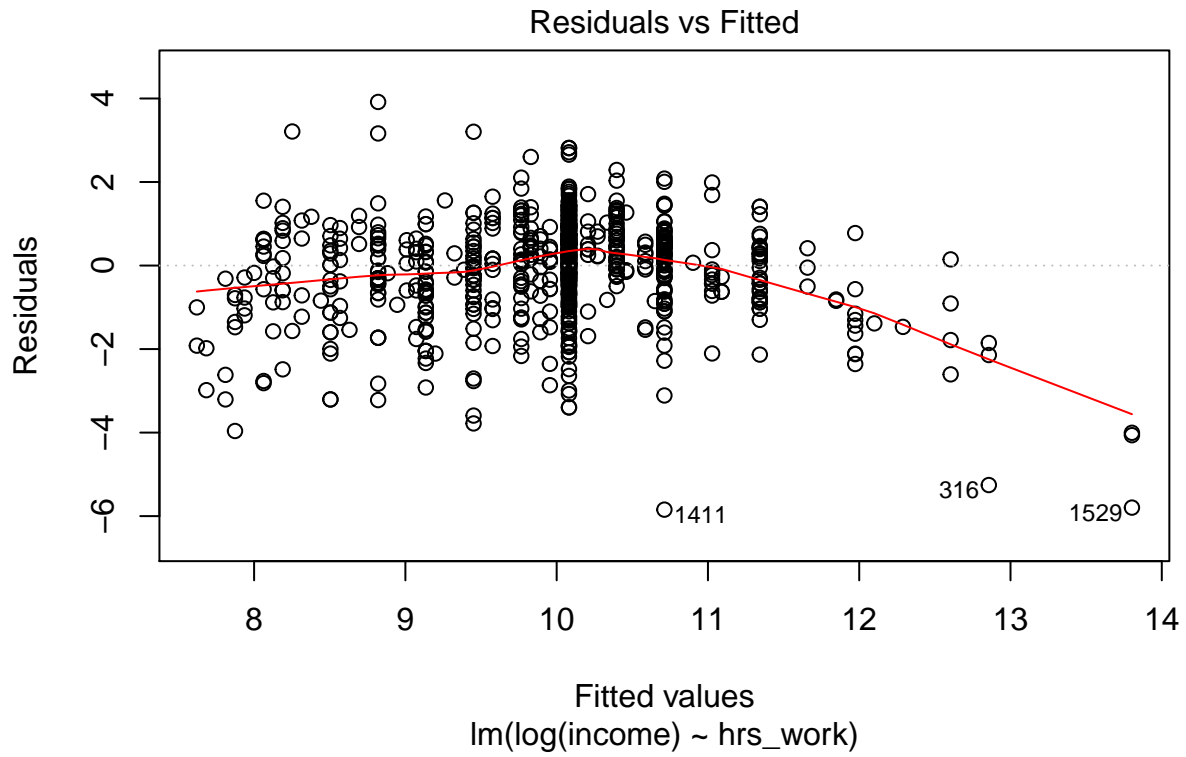
# see confidence intervals
confint(mod2)

##              2.5 %    97.5 %
## (Intercept) 7.33406617 7.7825764
## hrs_work    0.05753058 0.0686209

# scatterplot with fitted line
plot(acs12$hrs_work, log(acs12$income))
abline(mod2)
```

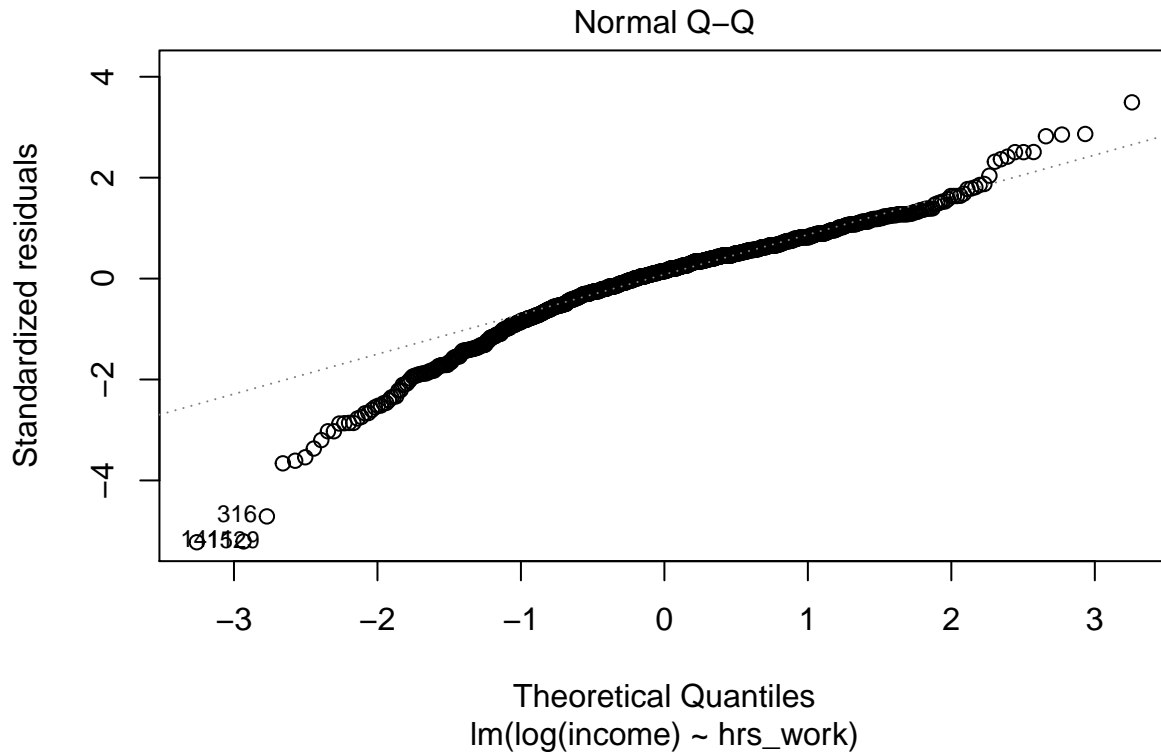


```
# check assumptions
plot(mod2, which = 1)
```



```
plot(mod2, which = 2)
```





```
# There still appears to be a violation of the constant variance assumption.
# This is basically because there is a non-linear relationship between hrs_work
# and log(income).
#
# We can try to fit a smooth line that follows the data. One approach is called
# regression splines.
#
# 1) Load the splines package (comes with R)
# 2) use the ns() function and specify the number of times the line can "change
# direction" (usually 3 - 5) as the degrees of freedom (df).
# ns = natural spline
```

```
library(splines)
mod3 <- lm(log(income) ~ ns(hrs_work, 3), data = acs12, subset = income > 0)
summary(mod3)
```

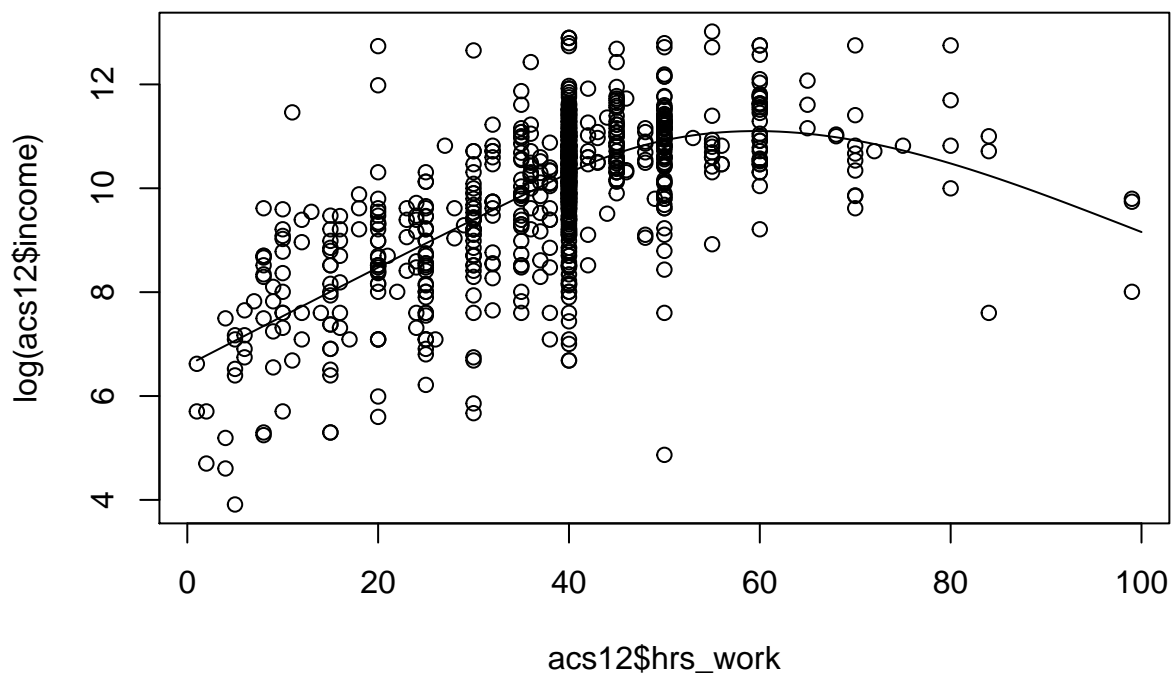
```
##
## Call:
## lm(formula = log(income) ~ ns(hrs_work, 3), data = acs12, subset = income >
## 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.0550 -0.4766  0.0788  0.5854  4.2620
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.6861     0.1865  35.847 < 2e-16 ***
## ns(hrs_work, 3)1  4.2222     0.1958  21.563 < 2e-16 ***
## ns(hrs_work, 3)2  6.1903     0.4629  13.374 < 2e-16 ***
## ns(hrs_work, 3)3  1.4197     0.4230   3.356 0.000824 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 890 degrees of freedom
## (377 observations deleted due to missingness)
## Multiple R-squared:  0.4641, Adjusted R-squared:  0.4623
## F-statistic: 257 on 3 and 890 DF, p-value: < 2.2e-16

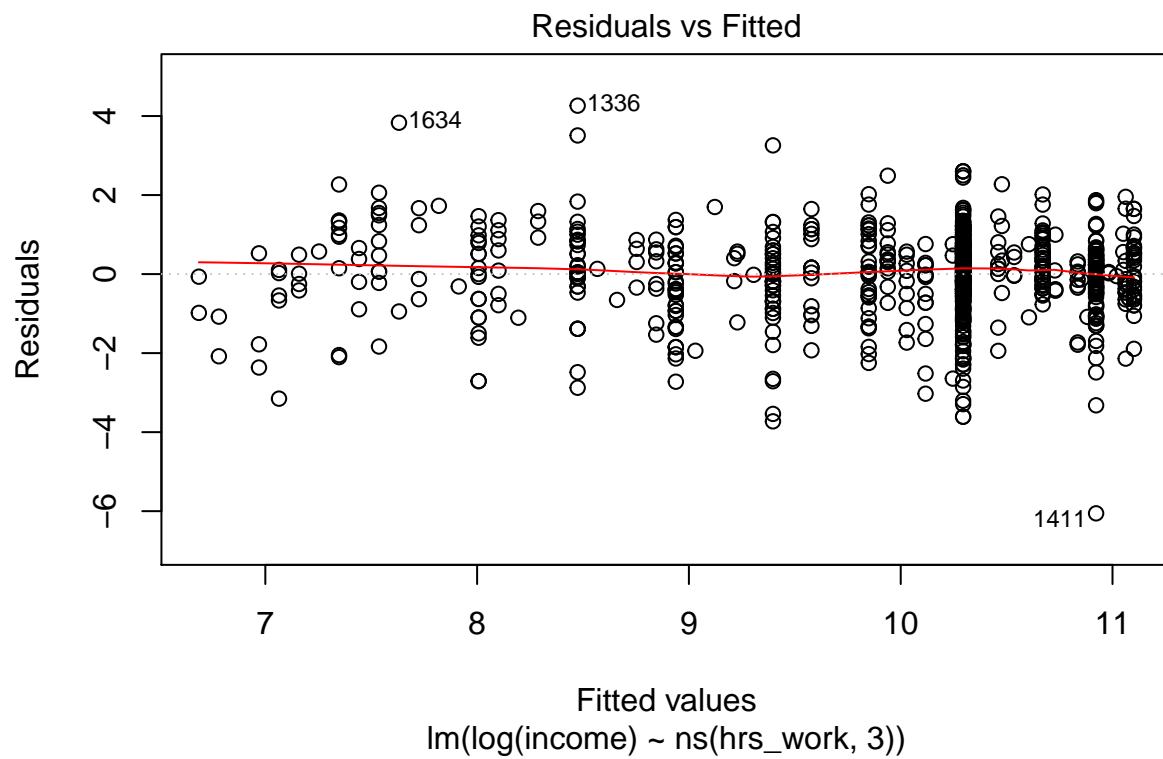
# There is really no model interpretation
#
# Plot the fitted line over the raw data
plot(acs12$hrs_work, log(acs12$income))

# Since the prediction is no longer a straight line we cannot use abline().
# Use the predict() function to get fitted values on the range of 0 - 100.
# Notice the newdata argument requires a data frame
y <- predict(mod3, newdata = data.frame(hrs_work = 1:100))

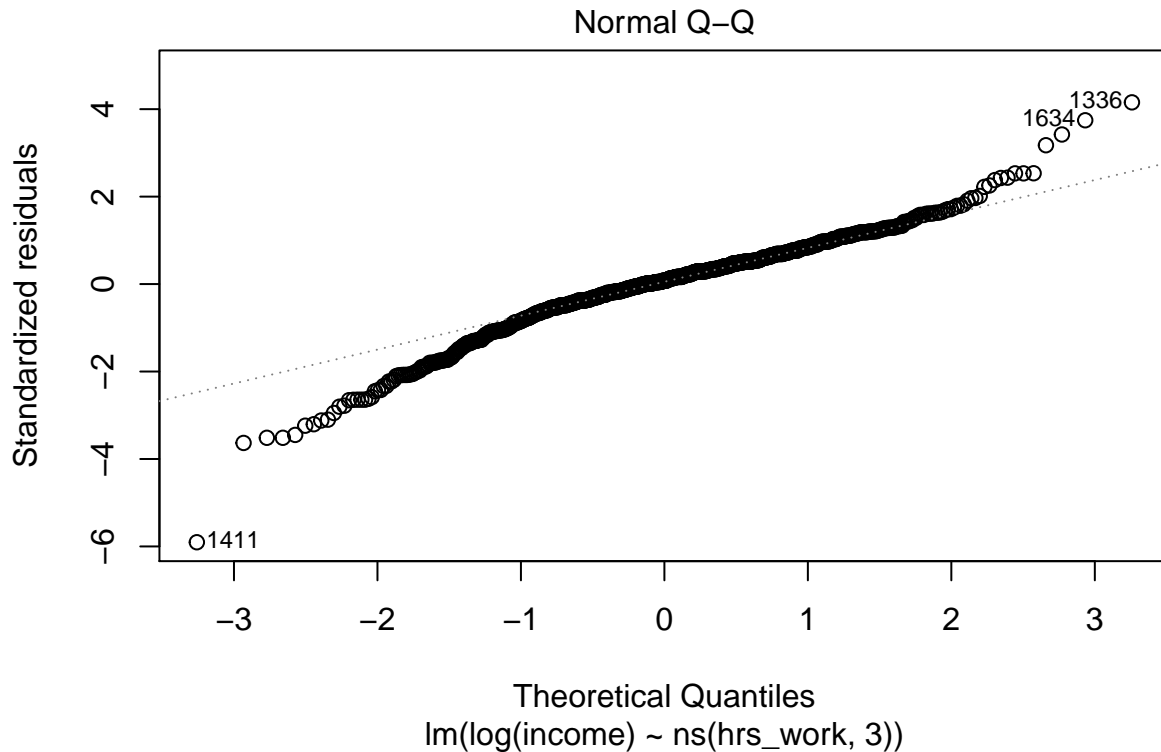
# add line to plot
lines(1:100, y)
```



```
# The assumptions appear to be met though row 1411 seems to be unusual
plot(mod3, which = 1)
```



```
plot(mod3, which = 2)
```

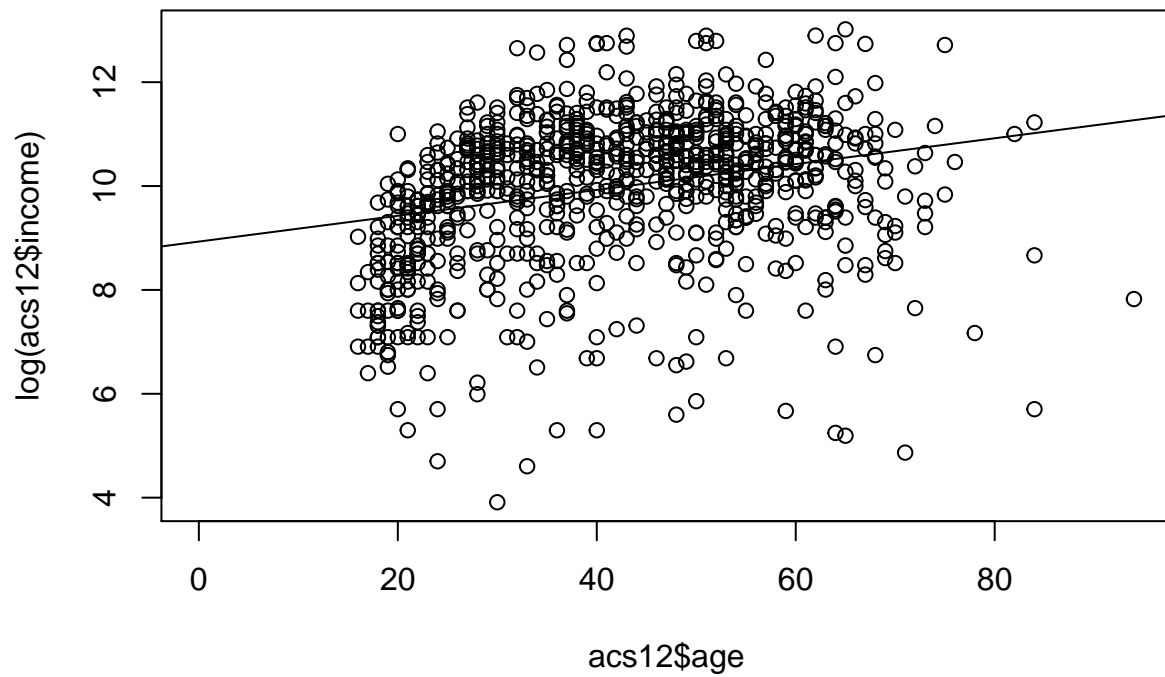


```
# Use model to make a prediction of expected income given hours worked.
predict(mod3, newdata = data.frame(hrs_work = c(20,30,40,50)),
       interval = "confidence") %>%
  exp()
```

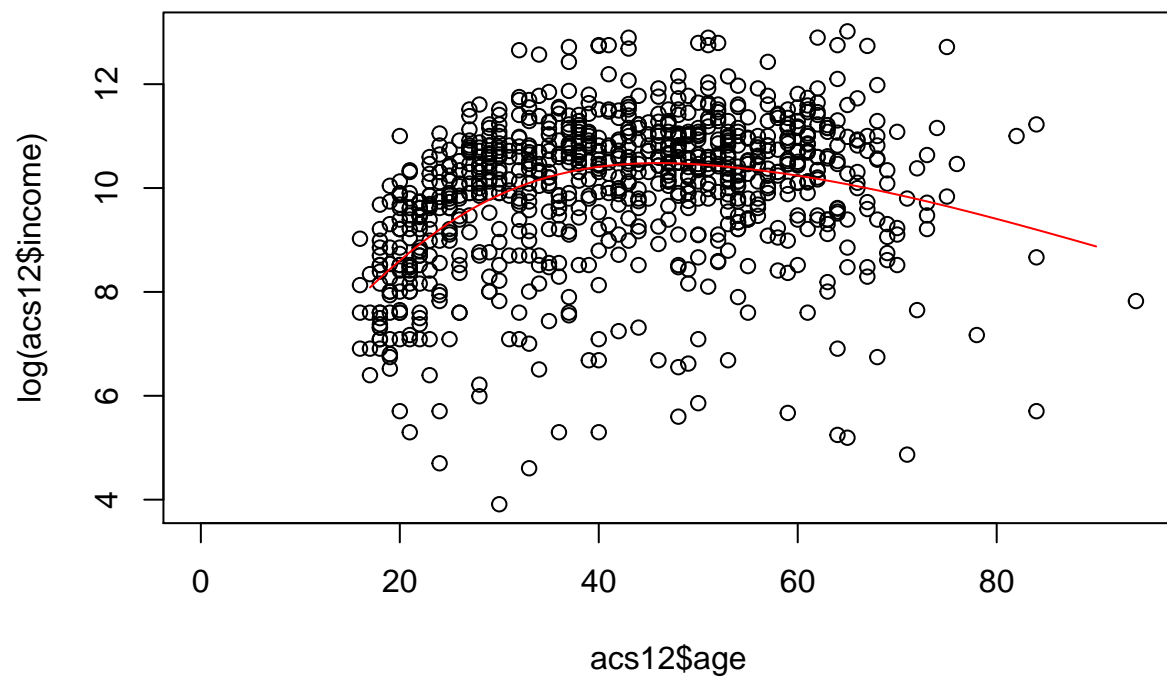
```
##          fit      lwr      upr
## 1  4791.953  4208.156  5456.741
## 2 12049.213 10670.627 13605.905
## 3 29574.489 27377.243 31948.083
## 4 55413.389 48720.156 63026.147
```

```
# YOUR TURN #8 -----

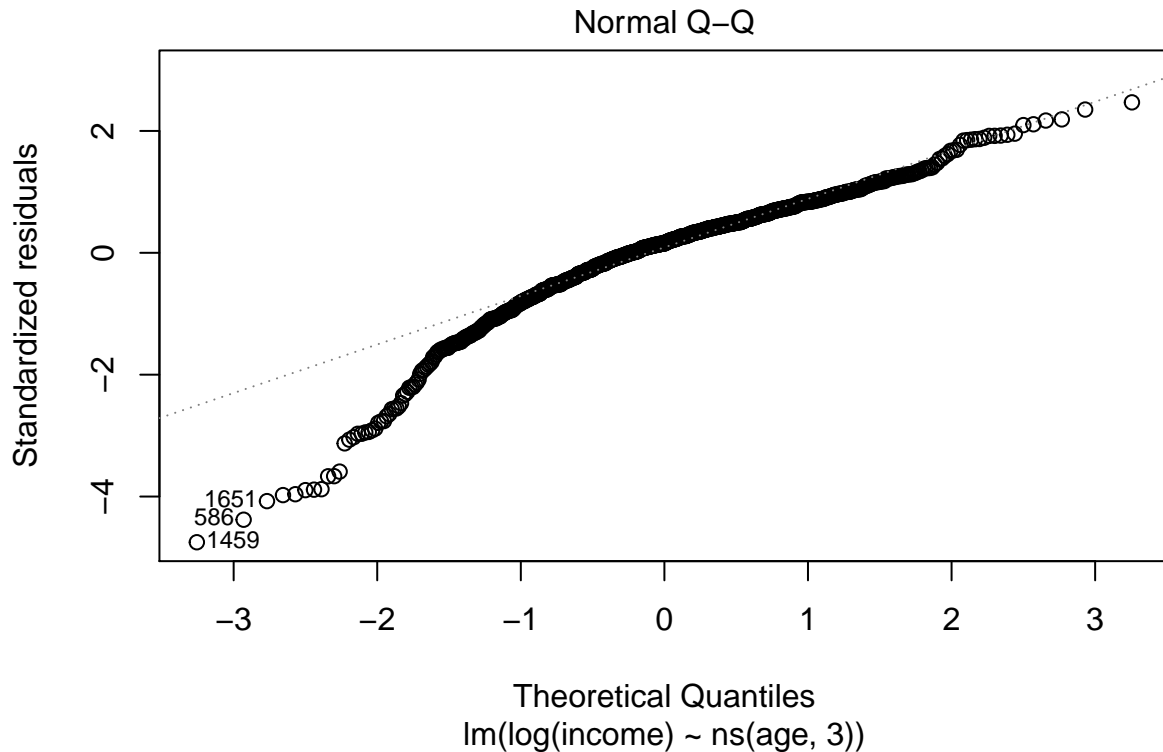
# Regress log(income) on age. That is, fit lm(log(income) ~ age, data = acs12).
# Subset the data: subset = income > 0 & age > 17
# Plot the data and fitted line.
# Maybe try using a natural spline with 3 changes of direction?
mod_age <- lm(log(income) ~ age, data = acs12, income > 0 & age > 17)
plot(acs12$age, log(acs12$income))
abline(mod_age)
```



```
mod_age2 <- lm(log(income) ~ ns(age, 3), data = acs12,  
               subset = income > 0 & age > 17)  
plot(acsc12$age, log(acsc12$income))  
y <- predict(mod_age2, newdata = data.frame(age = 17:90))  
lines(17:90, y, col = "red")
```



```
plot(mod_age2, which = 2)
```

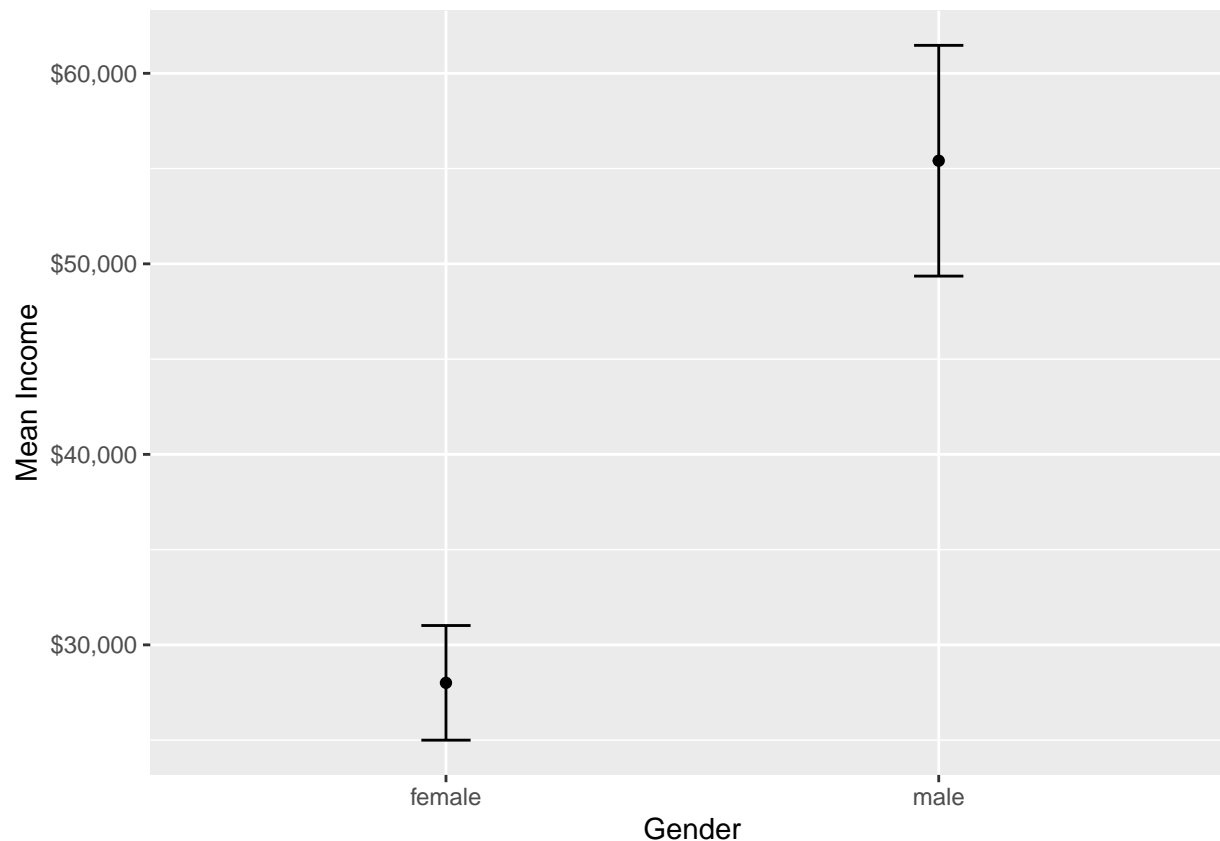


*# Appendix: Plotting means and confidence intervals -----*

```
# difference in mean income by gender.
# create a data frame of means and upper and lower CIs
income.ci <- acs12 %>%
  filter(income > 0) %>%
  group_by(gender) %>%
  summarise(mean = mean(income, na.rm = TRUE),
            lower = smean.cl.normal(income)["Lower"],
            upper = smean.cl.normal(income)["Upper"])
income.ci
```

```
## # A tibble: 2 x 4
##   gender  mean lower upper
##   <fct>  <dbl> <dbl> <dbl>
## 1 female 28008. 24999. 31016.
## 2 male  55412. 49356. 61468.
```

```
library(ggplot2)
ggplot(income.ci, aes(x = gender, y = mean)) +
  geom_point() +
  geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.1) +
  scale_y_continuous(labels = scales::dollar) +
  labs(y = "Mean Income", x = "Gender")
```

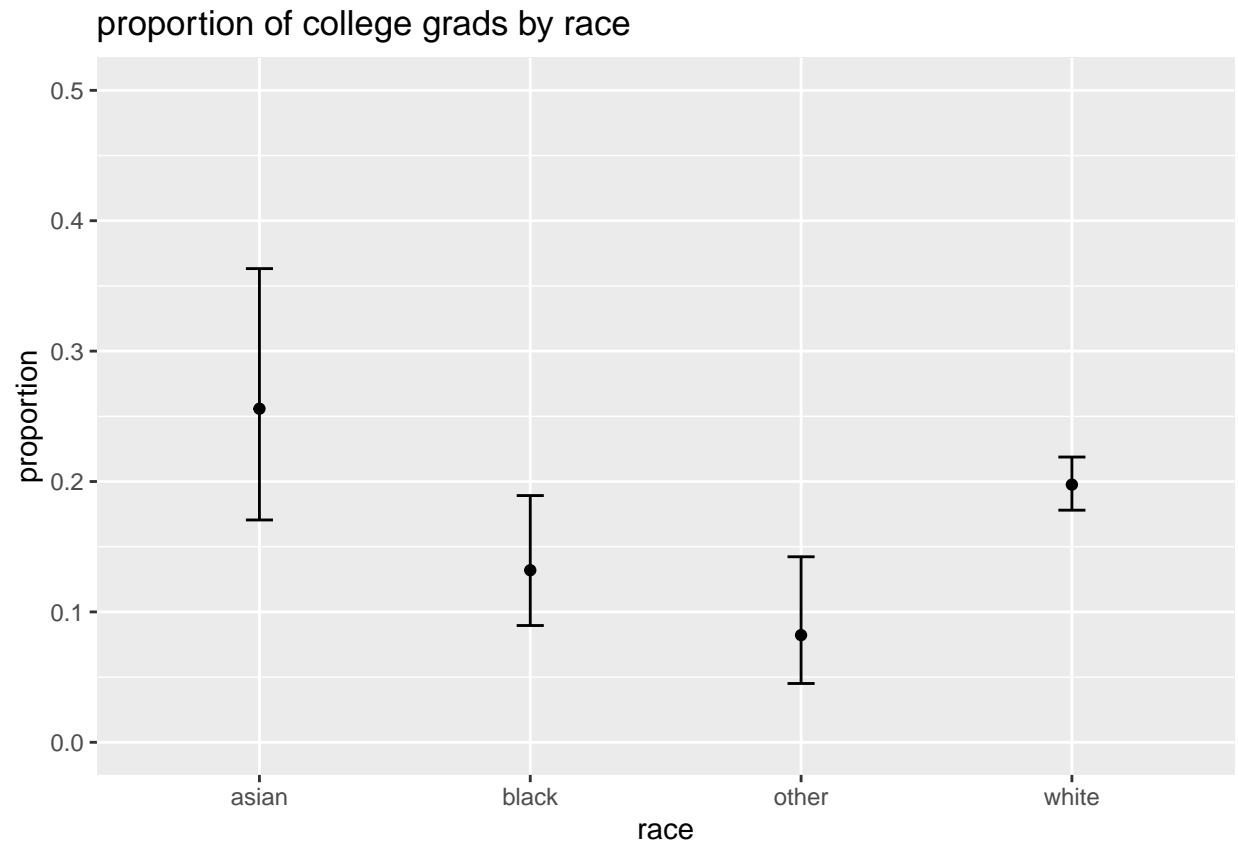


```
# plot proportion of college graduates by race
# first get counts of college grads and total race
tab <- xtabs(~ race + edu, data = acs12) %>%
  addmargins(margin = 2)
# get confidence intervals
binom.out <- binom.confint(x = tab[, "college"],
                           n = tab[, "Sum"],
                           methods = "prop.test")
# add race to the binom.out data frame
binom.out$race <- rownames(tab)
binom.out

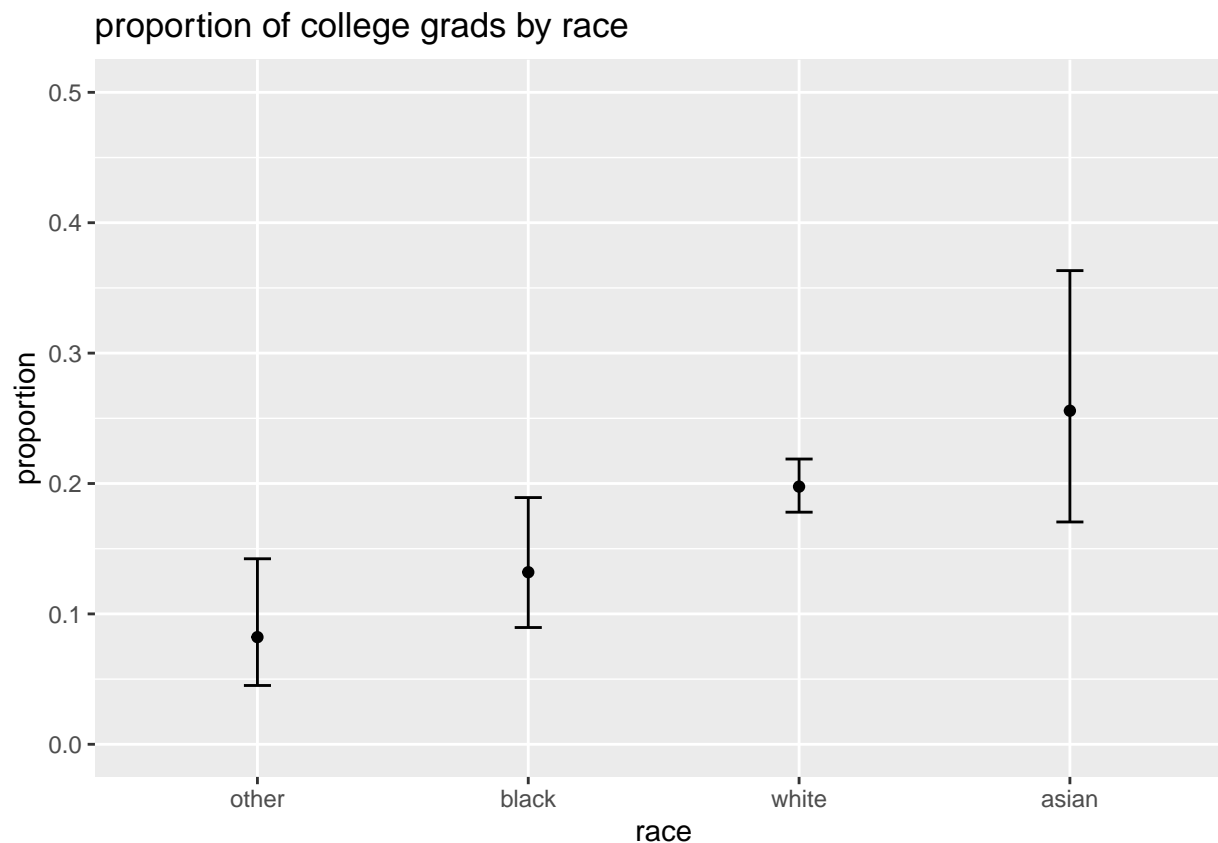
##      method  x    n   mean   lower   upper race
## 1 prop.test 22   86 0.25581395 0.17051512 0.3632744 asian
## 2 prop.test 26  197 0.13197970 0.08956575 0.1892122 black
## 3 prop.test 12  146 0.08219178 0.04510384 0.1422820 other
## 4 prop.test 299 1513 0.19762062 0.17801538 0.2187826 white

# create the plot
ggplot(binom.out, aes(x = race, y = mean)) +
  geom_point() +
  geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.1) +
  ylim(0, 0.5) +
  labs(y = "proportion", title = "proportion of college grads by race")
```





```
# reorder race by mean
ggplot(binom.out, aes(x = reorder(race, mean), y = mean)) +
  geom_point() +
  geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.1) +
  ylim(0,0.5) +
  labs(y = "proportion", x = "race",
  title = "proportion of college grads by race")
```



```
# Appendix: Demonstration of confidence intervals -----

# 95% Confidence Interval theory: sample the data, calculate a confidence
# interval, repeat many times. About 95% of confidence intervals will contain
# the "true" value you're trying to estimate.

# Simulation to demonstrate

# simulate a population of 10,000 from a Normal distribution with mean 10 and
# standard deviation 1.5.
population <- rnorm(10000, mean = 10, sd = 1.5)

# The TRUE mean value is 10. Pretend we don't know that.

# sample the population (size = 30) and calculate 95% confidence interval of
# mean:
s <- sample(population, size = 30, replace = TRUE)
ci <- t.test(s)$conf.int
ci

## [1]  9.605766 10.612248
## attr(,"conf.level")
## [1] 0.95

# Does confidence interval contain mean?
ci[1] < 10 & ci[2] > 10

## [1] TRUE
```

```

# Let's do this 1000 times
result <- logical(length = 1000) # empty vector to store result (TRUE/FALSE)
for(i in 1:1000){
  s <- sample(population, size = 30, replace = TRUE)
  ci <- t.test(s)$conf.int
  result[i] <- ci[1] < 10 & ci[2] > 10
}

# what proportion of times did confidence interval contain 10?
mean(result)

## [1] 0.944

# The process works about 95% of the time. Hence, the reason we call it a
# "confidence" interval. We're confident the process will return a confidence
# interval that contains the true value.

# Here's another demonstration that produces a plot.

# a random seed that ensures you get the results I got
set.seed(6)

# create an empty data frame containing a counter and upper and lower CI
# values.
dat <- data.frame(i = 1:50, lwr = NA, upr = NA)

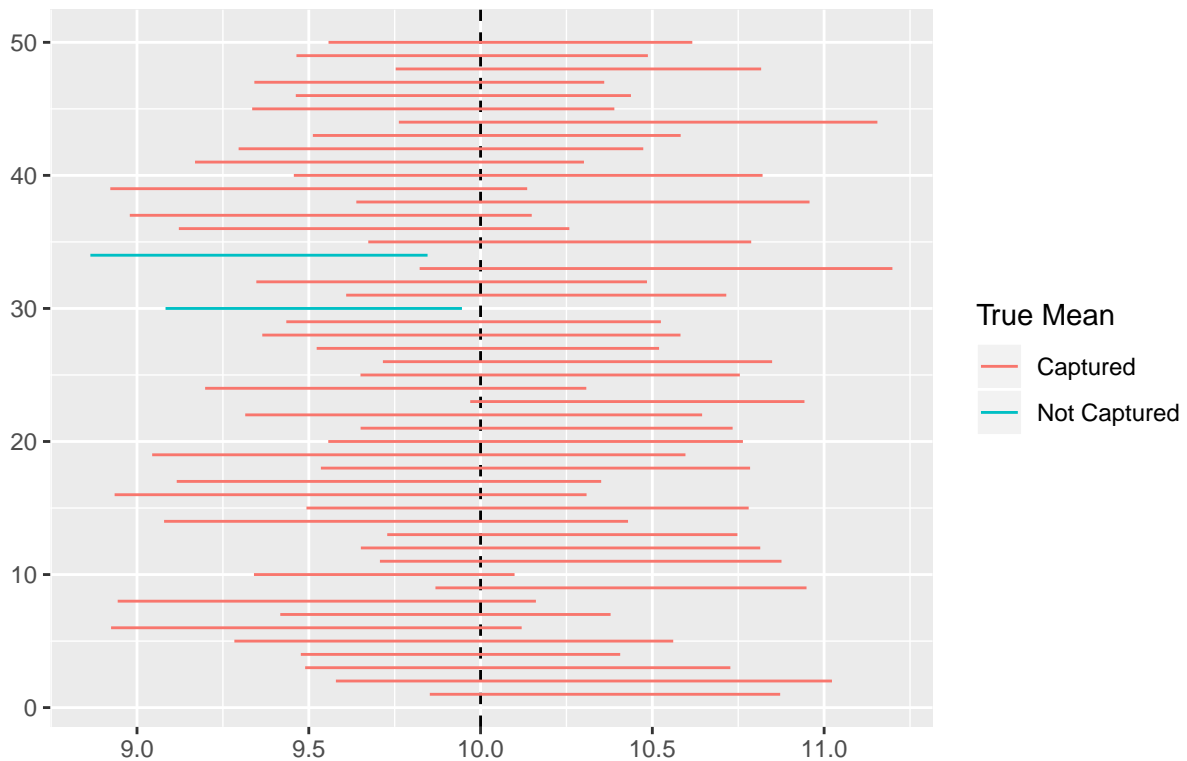
# Sample from "population" and calculate CI 50 times
for(i in 1:50){
  s <- sample(population, size = 30, replace = TRUE)
  ci <- t.test(s)$conf.int
  dat[i,"lwr"] <- ci[1]
  dat[i,"upr"] <- ci[2]
}

# create an indicator that identifies whether or not the CI captured the "True"
# mean of 10.
dat$grp <- ifelse(dat$lwr < 10 & dat$upr > 10, "Captured", "Not Captured")

# Generate plot using the ggplot2 package
library(ggplot2)
ggplot(dat) +
  geom_vline(xintercept = 10, linetype = 2) +
  geom_segment(aes(x = lwr, y = i, xend = upr, yend = i, color = grp)) +
  labs(x = "", y = "", title = "50 random confidence intervals") +
  scale_color_discrete("True Mean")

```

## 50 random confidence intervals



```
# Appendix: Basic bootstrapping -----
# Calculate a bootstrap CI of hrs_work using the Hmisc function smean.cl.boot.
# If you run it multiple times you will get slightly difference intervals.
smean.cl.boot(acs12$hrs_work, B = 1500)
```

```
##      Mean      Lower      Upper
## 37.97706 37.11718 38.84583
```

```
# To see the bootstrapped estimates, set reps = TRUE
smean.cl.boot(acs12$hrs_work, B = 100, reps = TRUE)
```

```
##      Mean      Lower      Upper
## 37.97706 37.12229 38.76137
## attr(,"reps")
##      [1] 37.07716 37.94473 37.89781 38.52450 38.05318 37.14911 37.65902 37.28676
##      [9] 37.09802 37.25339 38.01356 37.65798 37.74035 38.41710 38.36809 38.21794
##     [17] 37.77894 38.40563 37.76434 38.69343 38.14390 38.31908 37.70386 37.66111
##     [25] 38.08968 38.40146 38.17101 38.26590 38.36913 37.51199 38.58498 38.05839
##     [33] 37.28467 38.63399 38.11157 38.36809 38.06986 37.48801 38.20021 37.83733
##     [41] 38.69969 38.54432 38.75808 37.96663 38.08759 37.58707 38.13139 38.25756
##     [49] 37.87070 38.64964 38.20021 37.56204 37.82690 37.52868 39.16997 37.80918
##     [57] 38.82795 38.12304 37.40042 37.54849 38.20438 37.77998 38.27529 37.75078
##     [65] 37.85297 38.45568 37.92179 37.65276 38.76434 37.65172 38.07299 37.84984
##     [73] 37.42753 38.35558 38.51721 37.21689 38.45360 37.97914 37.60584 38.38060
##     [81] 37.52346 38.68509 38.68717 37.09489 37.87487 37.59854 38.01981 38.11783
##     [89] 37.98853 38.53076 37.77894 37.45464 37.78311 37.72888 38.03441 37.93743
```

```
## [97] 37.91241 38.44108 38.11783 37.54015
# The reps represent the 100 means that resulted from the 100 bootstrapped
# samples.

# The boot package that comes with R provides the boot() function that allows
# you to bootstrap just about any statistic as long as you can write a function
# for it.

# Here we bootstrap the median of income. How certain is that?
median(acs12$income, na.rm = TRUE)

## [1] 3000
# load the boot package
library(boot)

##
## Attaching package: 'boot'
## The following object is masked from 'package:mosaic':
##
##      logit
## The following object is masked from 'package:survival':
##
##      aml
## The following object is masked from 'package:lattice':
##
##      melanoma
# Write a function that calculates the median of a bootstrapped sample. The [i]
# is required to sample the selected indices (rows) generated by the
# re-sampling.
bootMedian <- function(x, i)median(x[i], na.rm = TRUE)

# Run the bootstrap 999 times
boot.out <- boot(data = acs12$income,
                 statistic = bootMedian,
                 R = 999)

# Get the percentile confidence interval of the bootstrapped estimates using the
# boot.ci() function, also in the boot package
boot.ci(boot.out, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%      (1200, 4800 )
## Calculations and Intervals on Original Scale
```

```

# Another example:
# standard deviation. How to get a confidence interval for SD of income?
sd(acs12$income, na.rm = TRUE)

```

```
## [1] 46567.23
```

```

# Use the bootstrap. First write a function:
bootSD <- function(x, i)sd(x[i], na.rm = TRUE)

```

```

# Run the bootstrap.
boot.out <- boot(data = acs12$income,
                 statistic = bootSD,
                 R = 999)

```

```

# Calculate the percentile confidence interval
boot.ci(boot.out, type = "perc")

```

```

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%      (40200, 52919 )
## Calculations and Intervals on Original Scale

```

```

# Another example:
# difference in medians

```

```

# How to get a confidence interval on the difference in medians?
mosaic::median(~ income | gender, data = acs12, na.rm = TRUE)

```

```

## female    male
##      680    8000

```

```

# diff() subtracts first element from the second
diff(mosaic::median(~ income | gender, data = acs12, na.rm = TRUE))

```

```

## male
## 7320

```

```

# Use the bootstrap. First write a function:
bootMedianDiff <- function(x, i)diff(mosaic::median(~ income | gender,
                                                    data = x[i,],
                                                    na.rm = TRUE))

```

```

# Run the bootstrap.
boot.out <- boot(data = acs12,
                 statistic = bootMedianDiff,
                 R = 999)

```

```

# Calculate the percentile confidence interval
boot.ci(boot.out, type = "perc")

```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```

## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%      ( 3400, 12800 )
## Calculations and Intervals on Original Scale

# Another example:
# 75th percentile

# How to get a confidence interval on the 75th percentile of income?
quantile(acs12$income, probs = 0.75, na.rm = TRUE)

##      75%
## 33700

# Use the bootstrap. First write a function:
boot75p <- function(x, i)quantile(x[i], probs = 0.75, na.rm = TRUE)

# Run the bootstrap.
boot.out <- boot(data = acs12$income,
                 statistic = boot75p,
                 R = 999)

# Calculate the percentile confidence interval
boot.ci(boot.out, type = "perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.out, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%      (30000, 36000 )
## Calculations and Intervals on Original Scale

```