R\_Data\_Analysis\_2

Ryan Womack

2024-09-18

Copyright Ryan Womack, 2024. This work is licensed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)

**Data Analysis 2**

*statistical tests, distributions, regression, bootstrap, and more + comparison with Python*

# 1. Overview

This workshop reviews the implementation of basic statistical tests and methods in R, with discussion contextualizing the appropriate use of these tests. For comparison purposes, an alternative Python approach to performing similar statistical analysis is illustrated in some cases. This workshop takes inspiration from the **Introduction to Modern Statistics** available via [github](https://github.com/OpenIntroStat/ims) and [online text](https://openintro-ims2.netlify.app/), part of the [OpenIntro](https://openintro.org) project.

Note that “packages” in R are the equivalent of “modules” in Python.

# 2. Setup and preparing data

## 2.1 R packages required

We will use the [*pak*](https://pak.r-lib.org/) package for installation as a more complete approach to package management. Replace the pkg commands with *install.packages()* versions if you prefer.

This session relies on the [*tidyverse*](https://tidyverse.org) suite of packages for data manipulation as a preference, although the same tasks could be accomplished in base R. The statistical functions used are those from base R. Install *pak* and *tidyverse* if you don’t already have them on your system. We will also need *reticulate* to run Python code chunks.

install.packages("pak", dependencies=TRUE)  
library(pak)  
pkg\_install("tidyverse")  
pkg\_install("reticulate")  
pkg\_install("infer")  
pkg\_install("TOSTER")  
devtools::session\_info()

Now let’s load the tidyverse, infer, TOSTER, and reticulate (for Python support only)

library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.5  
✔ forcats 1.0.0 ✔ stringr 1.5.1  
✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
✔ purrr 1.0.2   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(infer)  
library(TOSTER)  
Sys.setenv(RETICULATE\_PYTHON = "/usr/bin/python3")  
library(reticulate)

## 2.2 Data import and preparation

Now let’s grab some data. We will use a realistic data example, the [World Bank’s Gender Statistics database](https://genderdata.worldbank.org/), whose [raw data](https://databank.worldbank.org/data/download/Gender_Stats_CSV.zip) is directly downloadable. Other [World Bank Open Data](https://data.worldbank.org/) is available as well. See [genderdata.worldbank.org](https://genderdata.worldbank.org) for more background on the Gender Data portal. Note that we have to inspect the data and understand the variables first before manipulating in R – this is not an automatic process.

getOption("timeout")

[1] 60

options(timeout=6000)  
download.file("https://databank.worldbank.org/data/download/Gender\_Stats\_CSV.zip", "gender.zip")  
unzip("gender.zip")  
gender\_data <- read\_csv("Gender\_StatsCSV.csv")

Rows: 305545 Columns: 68  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (4): Country Name, Country Code, Indicator Name, Indicator Code  
dbl (64): 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, ...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

Now we’ll perform a few steps to clean the data, focusing on generating a useable file for a few countries (Central Asia and Mongolia plus selected high population or high income countries), from the latest available data year with complete data, typically the year before the last in the data set. For this session, we’ll just run these steps without explaining them. Data cleaning and wrangling is covered in more detail in the Data Analysis 1 workshop (forthcoming). The final filtered output is the *gender\_data\_final* file, which we *attach* so that a copy of the data is made the default dataset for this session.

# clean the data to remove superfluous columns  
names(gender\_data)  
gender\_data <- gender\_data[,c(-2,-4)]  
names(gender\_data)  
  
# select countries of interest  
country\_list <- c("China", "Germany", "India", "Japan", "Kazakhstan", "Kyrgyz Republic", "Mongolia", "Russian Federation", "Tajikistan", "Turkmenistan", "United States", "Uzbekistan")  
gender\_data2 <-  
 gender\_data %>%  
 filter(`Country Name` %in% country\_list)  
  
# clean the data to focus on a recent more complete time period  
gender\_data3 <-  
 gender\_data2 %>%  
 pivot\_longer(3:66, names\_to = "Year", values\_to = "Value")  
  
#filter by year  
gender\_data2022 <-  
 gender\_data3 %>%  
 filter(Year=="2022")  
  
gender\_data2022 <- gender\_data2022[,-3]  
  
gender\_data2022wide <-  
 gender\_data2022 %>%  
 pivot\_wider(names\_from = "Indicator Name", values\_from = "Value")  
  
# now use a little sapply trick to select variables that don't have much missing data - here the proportion is 0.75 (the 0.25 in the function is 1-proportion desired)  
  
gender\_data\_filtered <- gender\_data2022wide[,!sapply(gender\_data2022wide, function(x) mean(is.na(x)))>0.25]  
  
# and lastly simplify the dataset by removing some of the topics we won't use  
  
phrases <- c("Worried", "Made", "Received", "Saved", "Used", "Coming", "Borrowed")  
  
gender\_data\_final <-   
 gender\_data\_filtered %>%  
 select(!starts\_with(phrases))  
  
# we'll also generate a couple of variables for future use  
  
gender\_data\_final$female\_high\_labor <- gender\_data\_final$`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`>70  
gender\_data\_final$male\_high\_labor <- gender\_data\_final$`Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)`>70   
  
attach(gender\_data\_final)

Let’s test for some basic relationships between labor participation, gender, fertility, and income, using the [World Bank’s own definition of gender](https://genderdata.worldbank.org/en/about):

“Gender refers to the social, behavioral, and cultural attributes, expectations, and norms associated with being male or female.”

Note that this is just for the purposes of demonstration, and not a serious investigation into these important research issues.

# 3. Statistical Tests

## 3.1 t-test

In statistics there are many **hypothesis tests**. Per the Wikipedia entry on [statistical hypothesis tests](https://en.wikipedia.org/wiki/Statistical_hypothesis_test),

“a statistical hypothesis test is a method of statistical inference used to decide whether the data sufficiently supports a particular hypothesis. A statistical hypothesis test typically involves a calculation of a test statistic. Then a decision is made, either by comparing the test statistic to a critical value or equivalently by evaluating a p-value computed from the test statistic. Roughly 100 specialized statistical tests have been defined.”

The hypothesis is usually framed as a “Null hypothesis”, describing the situation where there is no statistically significant difference, and the alternative hypothesis. Then a test is chosen appropriate to the situation, which often involves invoking a statistical distribution.

The *t-test* is one of the most common tests, used to determine if the difference between two groups, according to some numerical measure, is statistically significant. The Null hypothesis, usually denoted , is that there is no difference (a difference of zero). The Alternative, usually denoted or , is that the difference is not zero.

The t-test was originally, and sometimes still is called [“Student’s t-test”](https://en.wikipedia.org/wiki/Student's_t-test). The story of the student who invented this test, and his affiliation with a certain well known Irish stout, is interesting from both a statistical and human perspective. The -test has a few variants: a one sample test, a two sample test with unpaired results, and two sample test with paired results. The one sample test simply checks whether one measure is significantly different from the null. The two sample test with unpaired results compares whether two separate sets of observations are different, such as sampling the populations of two different cities. The two sample test with paired results implies that we have the same subjects in the dataset who are measured twice, perhaps at different intervals or via different measures. The -distribution is the underlying statistical distribution determining the test statistic and critical value that we check for. These are conveniently summarized by the *p-value*, which expresses the level of significance. We typically look for a <.05 to determine statistical significance, but this is a convention: other -value cutoffs can be used.

## 3.2 One Sample t-test

We start with a one sample -test to check the labor force participation rates of females and males across the countries in our dataset.

t.test(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`)

One Sample t-test  
  
data: Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)  
t = 12.495, df = 11, p-value = 7.667e-08  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 48.54567 69.30399  
sample estimates:  
mean of x   
 58.92483

This first example returns a statistically significant, but not very interesting result. The default is to test whether the variable is different than zero. It is not surprising that female labor force participation is higher than zero.

t.test(`Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)`, mu=70)

One Sample t-test  
  
data: Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)  
t = 1.4864, df = 11, p-value = 0.1653  
alternative hypothesis: true mean is not equal to 70  
95 percent confidence interval:  
 67.52371 82.77829  
sample estimates:  
mean of x   
 75.151

We refine the test by passing the option =70, to test whether the male labor participation rate is significantly different than 70. With a >.05 and a 95% confidence interval of (66,82), it is *not* significantly different than 70 in a statistical sense.

Note the use of options is a common R syntax technique. Base execution of the command without options gives sensible results, but one can pass many options to tweak the function’s behavior. Most of the things demanded by statisticians are available through these option tweaks. Typing *?t.test* (the question mark followed by the function name) will pull up the help that displays the possibilities.

?t.test

## 3.3 Paired t-test

Perhaps a more interesting question is to see if there is a significantly significant difference in male and female labor participation rates. We can compare them, country-by-country, using the paired=TRUE option, in the paired two-sample test below. For one explanation of this see [STHDA](http://www.sthda.com/english/wiki/paired-samples-t-test-in-r).

t.test(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`,`Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)`, paired = TRUE, alternative = "two.sided")

Paired t-test  
  
data: Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate) and Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)  
t = -3.9856, df = 11, p-value = 0.002138  
alternative hypothesis: true mean difference is not equal to 0  
95 percent confidence interval:  
 -25.186732 -7.265602  
sample estimates:  
mean difference   
 -16.22617

There is a statistically significant difference, with the 95% confidence interval of female participation being between 7 and 25 percentage points lower than male rates, and a mean difference of -16 points.

## 3.4 Python t-test

We will be giving a flavor of the Python approach to these problems, without going in depth. Python is covered in other workshops in greater detail.

Running Python in RStudio requires the *reticulate* package. Your Python installation should also have *numpy* and *scipy* installed. Python will not have direct access to the R data structures, so we use a simplified example here from [Builtin.com](https://builtin.com/data-science/t-test-python)

How to pass data from R to Python and vice-versa is described in this [blog post by Dima Diachkov](https://medium.com/data-and-beyond/how-to-seamlessly-integrate-python-into-r-rmarkdown-codes-2fe09cfdd0ee)

Note that our code chunk is labeled Python. For more *reticulate* package, see [this post](https://www.r-bloggers.com/2022/04/getting-started-with-python-using-r-and-reticulate/)

import numpy as np  
import pandas as pd  
from scipy import stats  
  
# import from R  
gender\_python = r.gender\_data\_final  
  
# print(gender\_python)  
  
labor = gender\_python.loc[:,"Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)"]  
  
# Hypothesized population mean  
mu = 70  
  
# Perform one-sample t-test  
# t\_stat, p\_value = stats.ttest\_1samp(labor, mu)  
t\_stat, p\_value = stats.ttest\_1samp(labor, mu)  
print("T statistic:", t\_stat)

T statistic: -2.348578772454398

print("P-value:", p\_value)

P-value: 0.038583930428602614

# Setting significance level  
alpha = 0.05  
  
# Interpret the results  
if p\_value < alpha:  
 print("Reject the null hypothesis; there is a significant difference between the sample mean and the hypothesized population mean.")  
else:  
 print("Fail to reject the null hypothesis; there is no significant difference between the sample mean and the hypothesized population mean.")

Reject the null hypothesis; there is a significant difference between the sample mean and the hypothesized population mean.

## 3.5 Chi-squared test (in R and Python)

Many operations in R are a question of finding the appropriate function, either using the built-in R help via the ? (for help on a specific command) or ?? (to search across functions by keywords), or by using the built-in help system. Using a search engine like DuckDuckGo for a topic plus “in R” or a help site like stackoverflow.com will also turn up useful leads. Or one can consult books such as the many [R-related book series](https://link-springer-com.proxy.libraries.rutgers.edu/search?new-search=true&query=Use+R&dateFrom=&dateTo=&sortBy=relevance) in [SpringerLink](https://login.proxy.libraries.rutgers.edu/login?url=https://link.springer.com) for more comprehensive how-to’s. I would not recommend using AI tools for anything important, since one must understand the results well enough to check them for errors.

So, for example, if we wanted to run a [chi-squared test](https://en.wikipedia.org/wiki/Chi-squared_test) instead of a t-test, some noodling about will turn up the *chisq.test* function. We won’t discuss the distribution or mathematical background of the chi-squared test here.

Take a look at this table of the number of countries with “high” labor participation rates of over 70%, for males and females.

table(female\_high\_labor,male\_high\_labor)

male\_high\_labor  
female\_high\_labor FALSE TRUE  
 FALSE 2 5  
 TRUE 0 5

We can test whether this is a statistically significant pattern via the chi-squared test, implemented with:

chisq.test(table(female\_high\_labor,male\_high\_labor))

Warning in chisq.test(table(female\_high\_labor, male\_high\_labor)): Chi-squared  
approximation may be incorrect

Pearson's Chi-squared test with Yates' continuity correction  
  
data: table(female\_high\_labor, male\_high\_labor)  
X-squared = 0.27429, df = 1, p-value = 0.6005

Which is *not* statistically significant, probably due to the small size of the matrix.

## 3.6 How to choose the appropriate statistical test?

This is a question that depends on many factors, that we won’t go into detail about in this introductory workshop, but you can consult the following resources:

* [UCLA’s quick guide](https://stats.oarc.ucla.edu/other/mult-pkg/whatstat/)
* [a nice brief summary article](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8327789/)
* You can also consult [Sage Research Methods’ Which Stats Test?](https://methods-sagepub-com.proxy.libraries.rutgers.edu/which-stats-test) for more information on choosing the appropriate statistical test. The other guides in [Sage Research Methods](https://methods-sagepub-com.proxy.libraries.rutgers.edu/) are good too for more in-depth exploration of specific tests and topics! - *Rutgers-restricted*

## 3.7 Distributions

One great aspect of R is the availability of tools to work with almost any statistical distribution. The density, distribution function, quantile function and random generation of a number are easily available with R functions, via the prefixes *d, p, q, and r*. For example for the normal distribution, we have:

# the density of a standard normal distribution at the value 2 (2 above mean of zero)  
  
dnorm(2)

[1] 0.05399097

# the cumulative percentage of a standard normal distribution below the value 2  
  
pnorm(2)

[1] 0.9772499

# mean and standard deviation of the distribution can also be specified  
# in this example the cumulative percentage of a normal distribution with mean 100 and s.d. 20, below the value 90  
  
pnorm(90, mean=100, sd=20)

[1] 0.3085375

# provides the numeric value of a particular quantile of the distribution (again standard normal in this example)  
  
qnorm(.9)

[1] 1.281552

# a single random draw from the standard normal distribution  
  
rnorm(1)

[1] -0.1275865

# five random draws from a normal distribution with mean 100 and s.d. 20  
  
rnorm(5, mean=100, sd=20)

[1] 129.80854 107.66022 83.41175 111.24976 109.33654

# getting help  
  
?rnorm

As usual the question mark will provide details about implementation, variables, and options.

Other distributions and their functions can be found on the [TaskViews on Distributions](https://cran.r-project.org/web/views/Distributions.html) page.

# 4. Correlation

We can also perform simple correlations to evaluate the strength of a relationship between two variables, but there are some cautions.

cor(`GDP per capita (constant 2010 US$)`,`Fertility rate, total (births per woman)`, na.rm=TRUE)

Why won’t this work? The defaults and options of R commands are not very standardized. Instead try this option:

cor(`GDP per capita (constant 2010 US$)`,`Fertility rate, total (births per woman)`, use="complete.obs")

[1] -0.5795293

The *cor.test* function gives more complete output

cor.test(`GDP per capita (constant 2010 US$)`,`Fertility rate, total (births per woman)`)

Pearson's product-moment correlation  
  
data: GDP per capita (constant 2010 US$) and Fertility rate, total (births per woman)  
t = -2.2488, df = 10, p-value = 0.04828  
alternative hypothesis: true correlation is not equal to 0  
95 percent confidence interval:  
 -0.865553906 -0.008432122  
sample estimates:  
 cor   
-0.5795293

# 5. Regression

*Regression* is a common and fundamental technique to model a relationship between a *response* variable (also called dependent or outcome variable) and one or more *explanatory* variables (also called independent or predictor variables). Proper implementation of regression requires careful attention to the data and examination of the model fit, variable selection, and more. The quick and dirty approach below is simply designed to show R syntax in action.

## 5.1 Linear Regression

We start at the beginning, with linear regresion and the *lm* command. Let’s see if there is a relationship between female labor participation and GDP:

lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`GDP per capita (constant 2010 US$)`)

Call:  
lm(formula = `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)` ~   
 `GDP per capita (constant 2010 US$)`)  
  
Coefficients:  
 (Intercept) `GDP per capita (constant 2010 US$)`   
 5.123e+01 4.685e-04

Which predicts about a 0.5 increase in the percentage labor participation of females for every $1000 rise in GDP. However, the presentation of the result is a bit odd. R only presents the coefficients by default. The *summary* command must be used to tease out the additional information that we expect in a regression table to evaluate fit and significance.

summary(lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`GDP per capita (constant 2010 US$)`))

Call:  
lm(formula = `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)` ~   
 `GDP per capita (constant 2010 US$)`)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-22.291 -10.311 4.305 8.679 16.604   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 5.123e+01 5.289e+00 9.687 2.13e-06 \*\*\*  
`GDP per capita (constant 2010 US$)` 4.685e-04 2.085e-04 2.247 0.0484 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 13.97 on 10 degrees of freedom  
Multiple R-squared: 0.3355, Adjusted R-squared: 0.2691   
F-statistic: 5.05 on 1 and 10 DF, p-value: 0.04841

Which is not significant at the 0.05 level. Note that we must decide on the appropriate -value to test for in advance and resist the temptation to say that this is “almost significant”. Note that this is heavily influenced by the fact that we are using a small sample of 12 countries, which could be an argument to use a 0.10 signficance level.

Let’s check some additional relationships:

summary(lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`Fertility rate, total (births per woman)`))

Call:  
lm(formula = `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)` ~   
 `Fertility rate, total (births per woman)`)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-31.482 -1.573 2.473 4.759 23.896   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)  
(Intercept) 84.981 12.334 6.890 4.25e-05  
`Fertility rate, total (births per woman)` -11.723 5.243 -2.236 0.0494  
   
(Intercept) \*\*\*  
`Fertility rate, total (births per woman)` \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 13.99 on 10 degrees of freedom  
Multiple R-squared: 0.3333, Adjusted R-squared: 0.2666   
F-statistic: 4.999 on 1 and 10 DF, p-value: 0.04936

summary(lm(`Fertility rate, total (births per woman)`~`GDP per capita (constant 2010 US$)`))

Call:  
lm(formula = `Fertility rate, total (births per woman)` ~ `GDP per capita (constant 2010 US$)`)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-1.1598 -0.5151 0.2188 0.5433 0.7865   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 2.602e+00 2.604e-01 9.991 1.6e-06 \*\*\*  
`GDP per capita (constant 2010 US$)` -2.308e-05 1.026e-05 -2.249 0.0483 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.6876 on 10 degrees of freedom  
Multiple R-squared: 0.3359, Adjusted R-squared: 0.2694   
F-statistic: 5.057 on 1 and 10 DF, p-value: 0.04828

Also note that if want to run the regression without an intercept, we can just add a -1 to the equation.

summary(lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`GDP per capita (constant 2010 US$)`-1))

Call:  
lm(formula = `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)` ~   
 `GDP per capita (constant 2010 US$)` - 1)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-45.42 22.33 37.70 51.06 54.50   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
`GDP per capita (constant 2010 US$)` 0.0017757 0.0004883 3.637 0.00391 \*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 42.91 on 11 degrees of freedom  
Multiple R-squared: 0.5459, Adjusted R-squared: 0.5047   
F-statistic: 13.23 on 1 and 11 DF, p-value: 0.003911

## 5.2 Multiple Regression

And if we need to explore *multiple regression*, with multiple explanatory variables, this is as easy as using a + sign in the equation:

summary(lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`Fertility rate, total (births per woman)`+`GDP per capita (constant 2010 US$)`))

Call:  
lm(formula = `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)` ~   
 `Fertility rate, total (births per woman)` + `GDP per capita (constant 2010 US$)`)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-26.297 -4.967 3.314 6.118 21.839   
  
Coefficients:  
 Estimate Std. Error t value  
(Intercept) 70.4503342 17.2103704 4.093  
`Fertility rate, total (births per woman)` -7.3868624 6.3067567 -1.171  
`GDP per capita (constant 2010 US$)` 0.0002980 0.0002512 1.186  
 Pr(>|t|)   
(Intercept) 0.0027 \*\*  
`Fertility rate, total (births per woman)` 0.2716   
`GDP per capita (constant 2010 US$)` 0.2659   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 13.71 on 9 degrees of freedom  
Multiple R-squared: 0.4234, Adjusted R-squared: 0.2953   
F-statistic: 3.305 on 2 and 9 DF, p-value: 0.08392

## 5.3 Stored Regression Objects

A very important feature of R is the flexibility that derives from the ability to store (and use) our regression output as an R object. The output becomes an R object, and we can access its variables/components using the familiar $ notation. This can allow us to easily store and compare multiple models and use regression data such as residuals or predicted values as inputs into other equations and analyses.

regoutput<-lm(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`Fertility rate, total (births per woman)`+`GDP per capita (constant 2010 US$)`)  
names(regoutput)

[1] "coefficients" "residuals" "effects" "rank"   
 [5] "fitted.values" "assign" "qr" "df.residual"   
 [9] "xlevels" "call" "terms" "model"

regoutput$residuals

1 2 3 4 5 6   
 5.6933288 3.5491144 -26.2966144 3.0798884 21.8391887 6.5276402   
 7 8 9 10 11 12   
 5.9814379 8.0805563 -14.9933760 -0.5648145 -9.4097100 -3.4866397

Once we have store the regression output, numerous quick functions are available that build off of the regression results, such as *predict* for predicted values and *anova* for the Analysis of Variance.

# predicted values  
predict(regoutput)

1 2 3 4 5 6 7 8   
65.21567 72.62389 56.22061 71.93111 51.28281 50.12836 51.22556 62.97944   
 9 10 11 12   
47.65238 53.64881 77.13971 47.04964

# analysis of variance (anova)  
anova(regoutput)

Analysis of Variance Table  
  
Response: Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)  
 Df Sum Sq Mean Sq F value Pr(>F)   
`Fertility rate, total (births per woman)` 1 978.27 978.27 5.2022 0.0485 \*  
`GDP per capita (constant 2010 US$)` 1 264.65 264.65 1.4073 0.2659   
Residuals 9 1692.46 188.05   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## 5.4 Regression Diagnostics

*Regression diagnostics* are used to evaluate the fit and appropriateness of the regression model as applied. Once again, we are providing some illustrative examples without going into depth here.

One thing that we may wish to test is the normality of the regression residuals (i.e., the difference between predicted and actual values of the observations is normally distributed). See this [discussion of Normality at U Wisconsin](https://sscc.wisc.edu/sscc/pubs/RegDiag-R/normality.html) and this [article available in PubMedCentral](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3693611/) for a brief explanation.

We can apply a typically used test for normality, the Shapiro-Wilk Normality Test, as follows. We get standardized residuals from the model with *rstandard* (or more simply the residuals accessed with *regoutput$residuals*), and then use the *shapiro.test* function to perform the test, or the (less powerful) Kolmogorov-Smirnov Test with *ks.test*, which also allows us to test against different distributions (here we just specific *‘pnorm’* for the normal distribution).

Note the interpretation of these two tests is different. A <.05 for Shapiro implies *rejection* of normality. A <.05 for K-S is *acceptance* of normality.

shapiro.test(rstandard(regoutput))

Shapiro-Wilk normality test  
  
data: rstandard(regoutput)  
W = 0.93422, p-value = 0.427

ks.test(regoutput$residuals, 'pnorm')

Exact one-sample Kolmogorov-Smirnov test  
  
data: regoutput$residuals  
D = 0.5823, p-value = 0.000215  
alternative hypothesis: two-sided

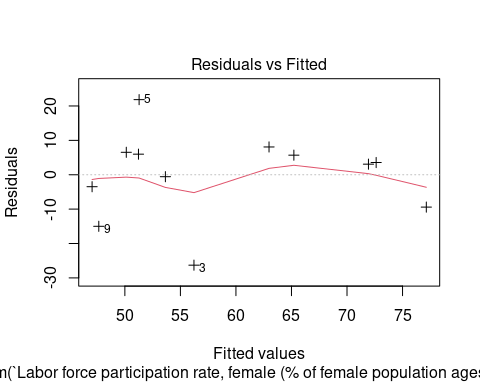
ks.test(rstandard(regoutput), 'pnorm', mean=0, sd=1)

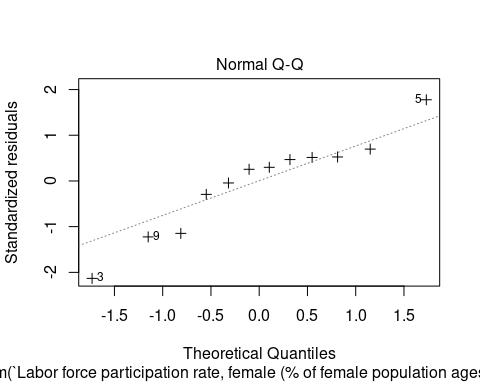
Exact one-sample Kolmogorov-Smirnov test  
  
data: rstandard(regoutput)  
D = 0.184, p-value = 0.7471  
alternative hypothesis: two-sided

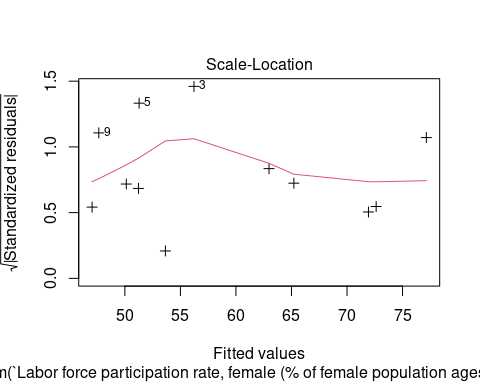
Note that standardizing the residuals makes a big difference for the Kolmogorov-Smirnow test!

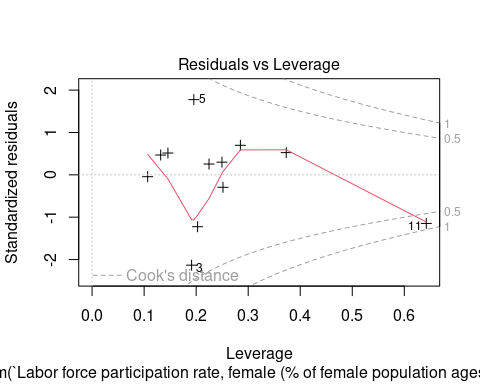
We can also easily pull up regression diagnostic plots:

plot(regoutput, pch=3)









## 5.5 glm and Logistic Regression

R is developed by statisticians for statisticians, so you will find every variant of regression that you might eventually need. The *glm* command implements generalized linear models (or alternatively, the [*glm2*](https://cran.r-project.org/package=glm2) package).

One commonly used alternative, necessary where the response variable is binary (categorical) is *logistic regression*. The *glm* command is used, with “family=binomial” as an argument. The binomial distribution represents the fact that the response variable is binary (either a two-level factor or directly coded as 0/1 - referred to as “one-hot” coding in certain circles).

We can try this out with our categorized variable that labels female labor participation as high if over 70% (otherwise low):

logistic\_output <- glm(female\_high\_labor ~ `Fertility rate, total (births per woman)`+`GDP per capita (constant 2010 US$)`, family=binomial)  
  
summary(logistic\_output)

Call:  
glm(formula = female\_high\_labor ~ `Fertility rate, total (births per woman)` +   
 `GDP per capita (constant 2010 US$)`, family = binomial)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-1.2787 -0.6362 -0.3845 0.6530 2.2036   
  
Coefficients:  
 Estimate Std. Error z value  
(Intercept) 4.925e+00 3.387e+00 1.454  
`Fertility rate, total (births per woman)` -2.313e+00 1.340e+00 -1.726  
`GDP per capita (constant 2010 US$)` -1.823e-05 4.076e-05 -0.447  
 Pr(>|z|)   
(Intercept) 0.1459   
`Fertility rate, total (births per woman)` 0.0843 .  
`GDP per capita (constant 2010 US$)` 0.6547   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 16.301 on 11 degrees of freedom  
Residual deviance: 11.227 on 9 degrees of freedom  
AIC: 17.227  
  
Number of Fisher Scoring iterations: 4

We won’t delve further into the interpretation of logistic regression or other regression alternatives here, but just know that nearly anything is possible with R!

## 5.6 Python regression

Linear Regression is part of the [scikit-learn](https://scikit-learn.org/) module which forms the core of machine learning in Python.

Some basic information on linear regression in Python is at [RealPython](https://realpython.com/linear-regression-in-python/#simple-linear-regression)

We include this code as a quick illustration of the approach. As is typical, the output in Python must be individually extracted with commands. The [stargazer](https://github.com/StatsReporting/stargazer) module is at least one approach to automating the process of generating output tables. The [statsmodels](https://www.statsmodels.org/stable/index.html) module is less commonly used, but provides an alternative in Python that generates output tables more easily.

import numpy as np  
import pandas as pd  
from scipy import stats  
from sklearn.linear\_model import LinearRegression  
  
# import from R  
gender\_python = r.gender\_data\_final  
  
# specify variables  
labor = gender\_python.loc[:,"Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)"]  
gdp = gender\_python.loc[:,"GDP per capita (constant 2010 US$)"]  
  
# this step is important to put the data in Python-friendly form  
labor2 = labor.array.reshape(-1, 1)  
gdp2 = gdp.array.reshape(-1, 1)  
model = LinearRegression().fit(labor2,gdp2)  
  
print(model.intercept\_, model.coef\_)

[-25782.0414102] [[716.19818691]]

r\_sq = model.score(labor2, gdp2)  
print(f"R-squared: {r\_sq}")

R-squared: 0.33554087609678596

# 6. Bootstrap

This section is modeled off the approach used by [Introduction to Modern Statistics](https://openintro-ims.netlify.app/), Second Edition by Mine Çetinkaya-Rundel and Johanna Hardin, which goes into greater depth and provides many useful code exercises. We’ll refer to this text as *IMS* below. As elsewhere, we are providing a quick and dirty demo, but please use these resources to go further!

In particular take a look at the section on [bootstrapping](https://openintro-ims.netlify.app/foundations-bootstrapping) and the [supplementary tutorials](https://openintrostat.github.io/ims-tutorials/).

Why the “bootstrap”? Many traditional statistical techniques rely on assumptions about the behavior or distribution of the data, assumptions that we cannot always be sure of satsifying. According to IMS, “some statistics do not have simple theory for how they vary, and bootstrapping provides a computational approach for providing interval estimates for almost any population parameter.”

The bootstrap works by sampling and resampling the original collection of data. The variability of the samples can be used to estimate the variability of the underlying population, without requiring any further assumptions about its characteristics.

Essentially, we resample our data to construct an empirical distribution, which we can then use to estimate standard errors and corresponding confidence intervals, as well as other parameters. We call it the “bootstrap” because we are “pulling ourselves up by our bootstraps” and creating the standard error ourselves via this replicate sampling process.

R makes this easy via the [*infer*](https://infer.netlify.app/) package from the tidyverse.

The bootstrap in R is constructed by a three-part sequence of commands: *specify*, *generate*, and *calculate*: - we *specify* the variable we want to sample - then *generate* the number of sample replicates we want to generate - then *calculate* the statistic of interest on the replicates

Before any sampling we can simply generate the point estimate of our proportion, in this case one of our indicator variables for the gender data, whether a woman can work in dangerous jobs in the same way as a man.

Note that |> is the “pipe” in base R (relatively newly introduced, so you won’t see it as often yet), versus the %>% “pipe” in tidyverse via *magrittr* (traditional).

point\_estimate <- gender\_data\_final %>%  
 specify(response = `A woman can work in a job deemed dangerous in the same way as a man (1=yes; 0=no)`) %>%  
 calculate(stat = "mean")

Warning: Removed 1 rows containing missing values.

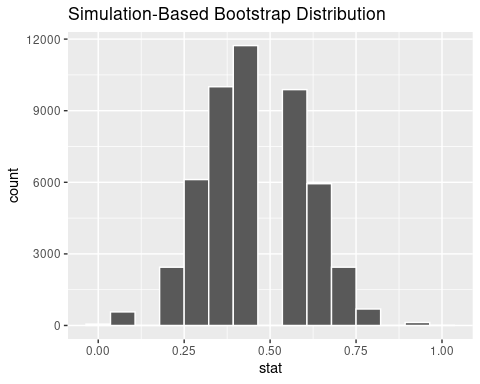
Then run replicate samples that we will use to estimate the confidence interval.

boot\_dist\_dangerous <- gender\_data\_final %>%  
 specify(response = `A woman can work in a job deemed dangerous in the same way as a man (1=yes; 0=no)`) %>%  
 generate(reps = 50000, type = "bootstrap") %>%  
 calculate(stat = "mean")

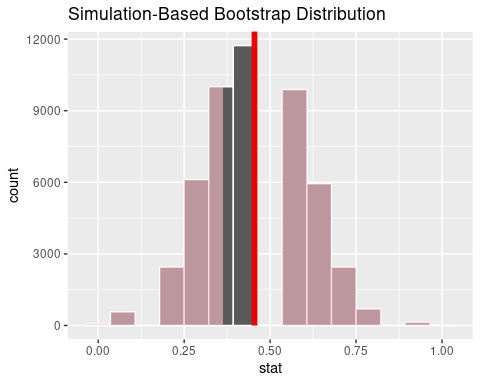
Warning: Removed 1 rows containing missing values.

We can easily visualize the distribution of the replicate samples.

boot\_dist\_dangerous %>%  
 visualize()



boot\_dist\_dangerous %>%  
 visualize() +  
 shade\_p\_value(obs\_stat = point\_estimate, direction = "two-sided")



boot\_dist\_dangerous %>%  
 get\_confidence\_interval(  
 point\_estimate = point\_estimate,  
 level = 0.95,  
 type = "se"  
 )

# A tibble: 1 × 2  
 lower\_ci upper\_ci  
 <dbl> <dbl>  
1 0.160 0.749

Note that we are resampling from a small population of 12, so even with large replications, our confidence interval is wide. Using this technique on the type of country data we have is atypical, and is just for demonstration purposes. A more typical usage would be on survey data with individual responses selected from a larger population. Again to quote *IMS*, “bootstrapping is best suited for modeling studies where the data have been generated through random sampling from a population.”

## 6.1 t-test via simulation

*IMS* has some examples of using *infer* to simulate t-test type results, however this is not as well-suited to the case of a paired t-test, for which the TOSTER package has a built in function that quickly implements a bootstrapped version of the t-test. This is just one example of how we can switch among the many ways of accomplishing a task in R.

First, instead of preserving the pairs, we get at the same idea by computing the difference in labor participation for males and females for each country.

labor\_diff <- `Labor force participation rate, male (% of male population ages 15-64) (modeled ILO estimate)` - `Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`

Then apply the bootstrap t-test from the *TOSTER* package, specifying 1000 replications.

boot\_t\_test(labor\_diff, R=1000)

Bootstrapped One Sample t-test  
  
data: x  
p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 8.914682 31.377117  
sample estimates:  
mean of x   
 16.22617

## 6.2 bootstrap to estimate variability of the slope of the regression

We can also bootstrap our way to some of the results of a regression model. Here we also use the *hypothesize* option of the *infer* package

var\_slope <- gender\_data\_final |>  
 specify(`Labor force participation rate, female (% of female population ages 15-64) (modeled ILO estimate)`~`Fertility rate, total (births per woman)`) |>  
 hypothesize(null = "independence") |>  
 generate(reps = 500, type = "permute") |>  
 calculate(stat = "slope")  
  
var\_slope |>   
 # Ungroup the dataset  
 ungroup() |>   
 # Calculate summary statistics  
 summarize(  
 # Mean of stat  
 mean\_stat = mean(stat),   
 # Std error of stat  
 std\_err\_stat = sd(stat)  
 )

# A tibble: 1 × 2  
 mean\_stat std\_err\_stat  
 <dbl> <dbl>  
1 -0.198 6.01

## 6.3 Python bookstrap

Finally we’ll briefy look at the [Python approach to bootstrapping](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bootstrap.html).

from scipy.stats import bootstrap  
import numpy as np  
  
labor = r.labor\_diff  
  
#convert array to sequence  
data = (labor,)  
  
#calculate 95% bootstrapped confidence interval for median  
bootstrap\_ci = bootstrap(data, np.mean, confidence\_level=0.95,  
 random\_state=1, method='percentile')  
  
#view 95% boostrapped confidence interval  
print(bootstrap\_ci.confidence\_interval)

ConfidenceInterval(low=9.203395833333337, high=24.488916666666658)

There’s [more that can be done](https://machinelearningmastery.com/calculate-bootstrap-confidence-intervals-machine-learning-results-python/) of course, but we’ll stop here.

*Enjoy R!* *(and Python!)*