CHAPTER 1: Intro to R and RStudio

- **ceiling** () **:**returns the ceiling of your input
- **exp** () : takes in one (or more) numeric values and exponentiates them
- **floor** () : returns the floor
  of your input
- **getwd():** returns the filepath of the current working directory
- **help(topic, package):** searches the documentation of the specified package(s)to find information on the desired topic
- **install.packages('name of package'):** downloads and installs the desired package
- **installed.packages():** returns all packages installed in the specified libraries
- **library(package):** loads the desired package for use
- **read.csv():** reads a csv file in table format and creates a data frame from it
- **round** () **:**round your input to the closest integer
- **setwd(dir):** sets the working directory to the 'dir' file path
- **?topic:** a shortcut for the help() function

CHAPTER 2: Data Structures in R

BASE R:

- **as.character(x):** converts x to the type 'character'
- **as.factor(x):** converts x to a factor
- **as.numeric(x):** converts x to the type 'numeric'
- **c():** combines arguments to form a vector
- **class(x):** returns the names of the classes from which x inherits
- **factor(x, levels):** converts x to a factor with the specified levels
- **length(x):** returns the length of the object x
- **levels(x):** returns the levels of the factor x
- **names(x):** returns the names of the object x
- **print(x):** prints the value of the variable x
- **rep(x, times):** repeats x the specified number of times
- **seq(from, to, by):** creates a regular sequence of numbers
- **typeof(x):** returns the type of the object x
- **vector(mode, length):** creates a vector with the given length and mode

BASIC CALCULATIONS:

- **exp(x):** exponentiates x
- **log(x):** takes the natural log of x
- **max():** returns the maxima of the specified arguments
- **min():** returns the minima of the specified arguments
- **mean(x):** returns the arithmetic mean of x
- **median(x):** returns the median of x
- **order(x):** returns the index of each element of x in sorted order
- **range():** returns the minimum and maximum values for a numeric vector x
- **quantile(x, probs):** returns the quantiles of x corresponding to probs (the given vector of probabilities)
- **IQR():** returns the interquartile range for a numeric vector
- **sd(x):** returns the standard deviation of x
- **sort(x, decreasing):** sorts the vector x into ascending or descending order
- **sum():** returns the sum of the specified arguments
- **var(x):** returns the variance of x
- **which.max():** returns the index of the maxima of a vector
- **which.min():** returns the index of the minima of a vector
- **sqrt(x):** computes the (principal) square root of x
- **abs(x):** computes the absolute value of x

- **round(number):** specifies a number of digits to be displayed
- **ceiling(x):** takes a single numeric argument x and returns a numeric vector containing the smallest integers not less than the corresponding elements of x.
- **floor(x):** takes a single numeric argument x and returns a numeric vector containing the largest integers not greater than the corresponding elements of x

FACTORS:

- **is.factor(variable name):** checks if the column is a factor column
- **as.factor(variable name):** factorized variables

VECTORS:

- **is.vector(x):** returns TRUE if x is a vector and FALSE otherwise
- **x*y:** returns the elementwise product of x and y

MATRICES:

- **cbind(x, y):** joins x and y by columns
- **matrix(x, nrow, ncol, byrow):** returns a matrix of the values in x with the specified number of rows and columns
- **rbind(x, y):** joins x and y by rows
- **t(x):** returns the transpose of x
- **x %*% y:** returns the matrix multiplication of x and y

ARRAYS:

- **array(x, dim):** returns an array of the values in x with the specified dimensions
- **is.array(x):** returns TRUE if x is an array and FALSE otherwise

DATA FRAMES:

- **as.data.frame(x):** converts x to a data frame
- **colnames(x):** returns the names of the columns of x
- **data(x):** loads the dataset x from a loaded package
- **data.frame(tag = value):** returns a data frame with column names equal to the specified tags and column elements equal to the specified values
- **dim(x):** returns the dimensions of x
- **head(x):** returns the first 6 rows of x
- **ncol(x):** returns the number of columns in x
- **nrow(x):** returns the number of rows in x
- **rownames(x):** returns the names of the rows of x
- **tail(x):** returns the last 6 rows of x

LISTS:

- **list()**: creates a list of (possibly named) objects

CHAPTER 3: Working with Data Files

IMPORTING FILES:

- **prop.table(x)**:convert the counts to proportions
- **read_csv(file)**: reads a delimited file with comma separated values into a tibble
- **read_dta(file, encoding = NULL)**: reads in a .dta file from STATA
- **read_log()**: reads in a log file
- **read_por(file, user_na = FALSE)**: reads in a .por file from SPSS
- **read_sas(data_file, catalog_file = NULL, encoding = NULL)**: reads in a .sas file from SAS
- **read_sav(file, user_na = FALSE)**: reads in a .sav or .zsav file from SPSS
- **read_spss(file, user_na = FALSE)**: reads in a .spss file from SPSS
- **read_stata(file, encoding = NULL)**: reads in a .stata file from STATA
- **read_table()**: reads in a textual data file where each column is separated by one or more columns of space
- **read_tsv()**: reads a delimited file with tab separated values into a tibble
- **readxl()**: imports an excel file into R
- **write.csv()**:to write a data frame from R to a .csv file, you can use the write.csv() function.
- **write_dta(data, path, version = 14)**: writes a STATA .dta file
- **write_sas(data, path)**: writes a SAS file
- **write_sav(data, path)**: writes a .sav or .zsav (if compress = TRUE) SPSS file

MISSING DATA AND NaNs:

- **is.na(x)**: returns 1 (TRUE) if x is NA and 0 otherwise
- **is.nan(x)**: returns 1 (TRUE) if x is NaN and 0 otherwise
- **is.finite(x)**: returns 1 (TRUE) if x is finite and 0 otherwise
- **is.infinite(x)**:returns 1 (TRUE) if x is infinite and 0 otherwise
- **na.omit(object)**: returns the object with incomplete cases removed
- **complete.cases(x)**: returns a logical vector indicating which rows/observations of x are complete

BASIC SUMMARIES:

- **summary(object)**: returns the length, class, and mode if x is a character

or factor and the 5 number summary (min, 1st quartile, median, 2nd quartile, max) if x is numeric
- **table(x):** returns the counts for each level of x

OPERATORS AND STARTING LOGIC:

- **which(x):** returns the TRUE indices of a logical object
- **any(...):** returns TRUE if at least one of the values is TRUE and FALSE otherwise
- **all():** returns TRUE if all of the values are TRUE and FALSE otherwise
- **xor()** : returns TRUE if values are the same, FALSE otherwise

OTHER:

- **apply(X, MARGIN):** returns the values obtained by applying a function to the margins (1 = rows, 2 = columns) of an array or matrix
- **which.max(...):** returns the largest number
- **pmax(...):** returns the maximum value from one or more vectors
- **pmin(...):** returns the minimum value from one or more vectors
- **rowSums(x):** returns the sums of the rows of x
- **colSums(x):** returns the sums of the columns of x
- **unique()** : returns x with all duplicate rows or values removed

CHAPTER 4: Exploratory Data Analysis (Basics)

- **subset(x, subset):** returns subsets of vectors, matrices, and data frames which meet conditions
- **ifelse(test, yes, no):** returns a value with the same shape as test, filled with elements equal to yes or no depending on whether the element of test is TRUE or FALSE
- **barplot(height, names, col):** creates a bar plot with vertical or horizontal bars with heights given by the values in the vector 'height'
- **boxplot(x):** produces a box and whisker plot of the given values
- **hist(x, breaks, col):** plots a histogram of the given data values with some optional arguments to update
- **plot(x, y):** plots points with x-coordinates equal to 'x' and y-coordinates equal to 'y'
- **abline(a, b):** adds a straight line with slope 'a' and y-intercept 'b' to the current plot
- **cor(x, y):** returns the correlation of x and y
- **par(mfrow = c(x,y)):** plots graphs in a single figure with x rows and y columns
- **GGally::ggcorr():** creates a graphical display of the correlation matrix between all pairs of columns
- **GGally::ggpairs(data):** creates a matrix of pairwise distributions for all columns for the given dataset
- **gt::gt():** creates a polished table that includes footnotes, titles, etc.
- **gt:::as.tags.gt_tbl():** displays the table as HTML within a Jupyter notebook
- **gtsummary::tbl_summary(data, include):** summarizes a list of columns specified in the *include* argument
- **as_gt():** creates a gt table from the summary output

CHAPTER 5: Data Transformation and Summaries

- **tidyverse::as_tibble(data):** converts data frames to tibbles
- **as.data.frame(data):** converts data to data frames
- **dplyr::select(names or indices of columns):** selects a subset of columns in the data frame(or tibble)
- **dplyr::subset():** selects and filters data using row and column indices
- **dplyr::filter(data, vector of booleans):** chooses a subset of rows
- **dplyr::slice(data, vector of indices):** selects a slice of rows by index
- **dplyr::_slice_sample(n):** specifies the number of random rows to sample from the data
- **dplyr::slice_max(order_by, n):** specifies a column through the argument order_by and returns the n rows with the highest values in that column
- **dplyr::slice_min(order_by, n):** specifies a column through the argument order_by and returns the n rows with the lowest values in that column
- **dplyr::rename(new name = old name):** changes the names of individual variables
- **dplyr::case_when(condition 1 ~ the value associated with a TRUE for condition 1):** specifies more than two cases' conditions and multiple alternative values based on whether we meet or do not meet those conditions
- **dplyr::mutate(column name):** adds columns by taking in a data frame and a set of columns with associated names to add to the data
- **dplyr::arrange(desc(column name)):** takes in a data frame and a vector of columns used to sort the data. By default, this function sorts the data in increasing order, but we can use desc() ro sort in descending order
- **dplyr::count(data, column(s)):** counts the number of rows for each combination of unique values of one or more variables
- **dplyr::summarize(summary functions given column names):** computes summary statistics of the data and compute multiple statistics
- **dplyr::group_by():** specifies one or more columns with which to group the data by
- **dplyr::ungroup():** removes the group structure from the data and restores the data to a single data frame

CHAPTER 6: Merging and Reshaping Data

- **as.Date(data$date, formula):** converts between character representations and objects of class "Date" representing calendar dates.
- **lubridate::month():** Gets/sets months component of a date-time
- **lubridate::week():** returns the number of complete seven day periods that have occurred between the date and January 1st, plus one.

Reshaping Data:

- **tidyr::pivot_longer(cols, names_to, values_to):** changes the data from wide form to long form by increasing the number of rows and decreasing the number of columns
- **tidyr::pivor_wider(data, names_from, values_from):** changes the data from long form to wide form

Merging Data:

- **dplyr::left_join(table1, table2, by):** joins each row of table1 with all matches in table2 (keeps all observations in table1)
- **dplyr::right_join(table1, table2, by):** joins each row of table2 with all matches in table1 (keeps all observations in table2)
- **dplyr::inner_join(table1, table2, by):** only keeps observations from table1 that have a matching key in table2
- **dplyr::full_join(table1, table2, by):** keeps all rows from both tables and joins those that match
- **dplyr::semi_join(table1, table2, by):** returns all rows from table1 with a match in table2
- **dplyr::anti_join(table1, table2, by):** returns all rows from table1 with*out* a match in table2
- **dplyr::between(x, left, right):** detects where values fall in a specified range

CHAPTER 7: Visualization with ggplot2

- **ggplot2::ggplot():** creates the base object (a gray box) which can be added layers to this object
- **ggplot2::gemo_point(aes(x, y), color, size, shape):** adds a scatter plot layer and specifies the color, size and shape of the points
- **ggplot2::geom_histogram(aes(x), y, binwidth, alpha, color, fill, linetype, size, weight):** plots the distribution of one continuous variable and updates the color, fill, opacity of histogram bars and number of bins
- **ggplot2::geom_boxplot():**displays the distribution of a continuous variable and visualizes five summary statistics and all "outlying" points individually.
- **ggplot2::geom_density():** Computes and draws kernel density estimate, which is a smoothed version of the histogram
- **ggplot2::geom_smooth():** adds the estimated regression line to original plot
- **ggplot2::geom_bar():** creates a bar proportional to the number of cases in each group. If you want the heights of the bars to represent values in the data, use **geom_col()** instead.
- **ggplot2::geom_errorbar():** adds vertical error bars
- **ggplot2::labs(x, y, title):** adds different labels and a title
- **ggplot2::theme():** updates any of the theme options
- **ggplot2::theme_minimal():** changes the background color used
- **ggplot2::theme_bw():** changes the theme of the plot
- **ggplot2::scale_x_continuous(limits, breaks):** changes the x-axis and specifies limits, breaks and labels for a continuous variable
- **ggplot2::scale_y_continuous():** updates the y-axis for a continuous variable
- **ggplot2::scale_x_discrete():** sets the values for discrete x scale aesthetics
- **ggplot2::scale_y_discrete():** sets the values for discrete y scale aesthetics
- **ggplot2::scale_color_gradient():** specifies a two color gradient (low and high end)
- **ggplot2::scale_color_gradient2():** creates a diverging color gradient (low-mid-high)
- **ggplot2::scale_color_gradientn():** creates a n-color gradient
- **ggplot2::coord_cartesian():** sets limits on the

coordinate system which clips the values and zooms the plot

- **ggplot2::scale_fill_brewer(palette, guide):** controls the color palette of a discrete variable used for the fill aesthetic
- **ggplot2::scale_fill_manual(values, name):** specifies the colors used for multiple variables
- **ggplot2::scale_color_discrete():** updates the legend for grouping
- **ggplot2::facet_grid(row = vars(), col = vars()):** arranges plots as a grid where the rows and/or columns correspond to the variables grouped by
- **ggplot2::facet_wrap(facet):** wraps the plots into a rectangular format and specifies the columns
- **ggplot2::geom_abline():** adds a diagonal line
- **ggplot2::geom_hline():** adds a horizontal line
- **ggplot2::geom_vline():** adds a vertical line
- **ggplot2::annotate():** adds a text annotation
- **ggplot2::ggsave():** saves the last plot generated under the file name provided
- **patchwork::patchwork:** incorporates multiple plots together using "+" sign to put them side by side and "/" to stack them

CHAPTER 8: Probability Distributions in R

- **tidyverse::ecdf()**: returns a function, which can then be used to find the sample cumulative distribution for different values similar to the p[dist]() functions
- **set.seed()**: specifies a numeric seed value
- **sample(x, size, replace, prob)**: takes a random sample of a specified size from the elements of x using either with or without replacement
- **r[dist]()**: generates random samples from a given distribution
  - **rnorm(n, mean, sd)**
  - **rbinom(n, p, size)**
- **d[dist]()**: density function for the distribution
  - **dnorm(n, mean, sd)**
  - **dbinom(n, p, size)**
- **p[dist]()**: cumulative distribution function for the distribution
  - **pnorm(n, mean, sd)**
  - **pbinom(n, p, size)**
- **q[dist]()**: quantile function for the distribution
  - **qnorm(n, mean, sd)**
  - **qbinom(n, p, size)**
- **rbeta(n, shape1, shape2, ncp = 0)**: generates beta distribution with shape parameter
- **rbinom(n, size, prob)**: generates binomial distribution with probability of success and number of trails
- **rcauchy(n, location = 0, scale = 1)**: generates cauchy distribution with location parameter and scale parameter
- **rchisq(n, df, ncp = 0)**: generates Chi-square distribution with degrees of freedom
- **rexp(n, rate)**: generates exponential distribution with rate
- **rchisq(n, df, ncp = 0)**: generates Chi-square distribution with degrees of freedom
- **rgamma(n, shape, rate, scale)**: generates gamma distribution with parameters shape and scale (or alternatively specified by rate)
- **rnbinom(n, size, prob, mu)**: generates negative binomial distribution with size and probability parameters
- **rpois(n, lambda)**: generates Poisson distribution with parameter lambda
- **runif(n, min, max)**: generates uniform distribution with minimum value and maximum value
- **rgeom(n, prob)**: generates geometric distribution with probability parameter
- **rhyper(nn, m, n, k)**: generates hypergeometric distribution

- **rweibull(n, shape, scale):** generates Weibull distribution with shape and scale
- **rlnorm(n, meanlog, sdlog):** generates log normal distribution with mean and standard deviation on the log scale

CHAPTER 9: Hypothesis Testing

- **paste():** creates a single character string from multiple inputs
- **t.test(x, alternative, mu, conf.level):** runs a hypothesis test to compare the mean to a predetermined value
- **wilcox.test(x, alternative, mu, conf.level):** performs the one-sample Wilcoxon signed rank test and compares the median value of a sample to a theoretical value without normality assumption
- **cor(x, y):** calculates the correlation between two variables
- **cov(x, y):** calculates the covariance between two variables
- **cor.test(x, y, method):** tests for association between paired samples and observe whether this correlation is significantly different from zero
- **t.test(x, y):** performs a two sample paired t-test (or two-sided test)
- **t.test(x ~ y, data)/t.test(x ~ y, paired = FALSE):** performs a two sample two sided t-test when the data is not paired
- **kruskal.test():** compared two or more independent samples
- **aov():** compared two or more independent samples
- **summary(aov()):** prints the result of ANOVA function with the p-value
- **var.test(x ~ y, data):** implements an F test to test if the variance in both groups is equal
- **car::leveneTest(x ~ y, data):** implements a Levene's test for homogeneity of variance across more than two groups
- **fisher.test(x, y):** takes in either a contingency table as a matrix or two factor vectors x and y and compare distributions of categorical variables
- **chisq.test(x, y, correct):** performs a Peason's Chi-Squared test for large sample sizes
- **add_p():** adds p-values for hypothesis tests across populations

CHAPTER 10: Linear Regression

- **lm(formula = y ~ x, data):** fits a linear model; y is the outcome variable and x represents the independent variable; for multiple variables, separate them with a "+"
- **summary(model):** returns a summary of the model, which includes the estimated coefficients, the R-squared and adjusted R-squared values, p-values and so on
- **broom::tidy(model, conf.int, conf.level):** returns a data frame that consists of the estimated regression coefficients, standard errors, p-values and confidence interval (if specified) from the model
- **resid(model):** returns the residual values for the model
- **fitted(model):** returns the fitted values or estimated y values from the model
- **predict(model):** predicts on new data using the model and return the predicted values
- **anova(model1, model2, test):** performs a nested hypothesis test between two models and the exact test
- **print(anova(...)):** shows the two tested models along with the associated p-value
- **relevel(variable name, ref):** reorders the factor variable and specifies the new reference level using the *ref* argument

Diagnostic Measurements:

- **plot(model):** returns four diagnostic plots for linear regression models
- **hist(resid(model)):** plots a histogram of the residuals
- **qqnorm(resid(model)):** plots a qq-plot of the residuals from the model
- **qqline(resid(model), col):** adds the theoretical line for reference
- **rstandard(model):** uses the standardized residuals to create plots
- **car::vif(model):** calculates the variance inflation factors to quantify any collinearity between the included covariates
- **influence.measures(model):** provides a set of measures that quantity the influence of each observation on a model

Model Evaluation:

- **AIC(model):** finds the Akaike information criterion (AIC) values
- **BIC(model):** finds the Bayesian information criterion (BIC) values

Model Selection:

- **step(mode, direction, scope):** takes in an initial model to perform stepwise selection on along with a specified direction and a scope

CHAPTER 11: Logistic Regression

- **glm(formula, data, family):** fits a generalized linear model
- **broom::tidy(model, exponentiate = TRUE):** displays the estimated exponentiate coefficients from the model
- **resid(model, type = "pearson"):** finds the Pearson residuals
- **resid(model, type = "deviance"):** finds the deviance residuals
- **predict(model, type = "response"):** finds the predicted probabilities from the logistic model
- **pROC::roc(predictor, response, levels, direction):** builds an ROC curve by specifying a response and predictor vector
- **plot(ROC, print.acu. print.thres):** plots the ROC curve and adds some extra information about AUC and threshold that maximizes sensitivity and specificity
- **coords(ROC, x):** finds the coordinates for each threshold used to create the curve

Model Selection:

- **step(glm()):** performs stepwise variable selection with generalized linear model objects

Model Comparison:

- **lmtest::lrtest(model1, model2):** performs a Chi-squared likelihood ratio test for two nested models

CHAPTER 12: Writing Reports in R Markdown

- **knitr::opts_chunk$set:** changes the default values of chunk options in a document
- **kableExtra::kable():** produces a nicely formatted table from a data frame
- **kableExtra::kable_styling() :** specifies additional options
- **bookdown::html_document2:** knit file to an html document
- **bookdown::word_document2:** knit file to a word document

CHAPTER 13: Expanding your R Skills

- **stringr::str_trim(string):** removes whitespace from start and end of string
- **stringr::str_squish(string) :** reduces repeated whitespace inside a string
- **stringr::str_split(string, pattern):** vectorised over string and pattern
- **grepl(x, pattern):** takes in a character vector x and a pattern to search for then returns a logical vector for whether or not each element of x has a match for that pattern
- **stringr::str_replace(string , pattern, replacement):** replaces matched patterns in a string
- **stringr::str_replace_all(st ring, pattern, replacement):** performs multiple replacements in each element of string
- **ls():** finds all current objects and check whether we use the same name for different objects or different names for the same object