|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Basics print("Hello world!")  # assign value to object  myNumber <- 5  # combine values into vector  myVector <- c(1, 2, 3)  # first element in vector  myVector[1]  # orders values or cases  sort(myVector)  # arithmetic operations  sum(1, 2, 100), +, -, \*, /  sqrt(157)  abs(data\_set$Y)  # logical operations  >, <, >=, <=, ==, !=, |, &  # results in a variable with values  # of TRUE or FALSE  data\_set$C <- data\_set$A > data\_set$B | Summary Tables # compute five-number summary  favstats(~ Y, data = data\_set)  # create frequency table  tally(data\_set$Y)  tally(~ Y, data = data\_set)  # tally by condition  tally(~ Y < 1900, data = data\_set)  # two-way frequency table  tally(Y ~ X, data = data\_set, margin = TRUE, format = “proportion”) | | | Simple Statistics mean(data\_set$Y)  var(data\_set$Y)  sd(data\_set$Y)  cohensD(Y ~ X, data = data\_set)  cor(Y ~ X, data = data\_set)  b1(Y ~ X, data = data\_set)  b1(one\_model)  pre(Y ~ X, data = data\_set)  f(Y ~ X, data = data\_set) | |
| Data Frame # structure of data frame  str(data\_set)  # view first/last six rows  head(data\_set)  tail(data\_set)  # select multiple variables  select(data\_set, Y1, Y2)  # first six rows of selected variables  head(select(data\_set, Y1, Y2))  # select variable (a column)  data\_set$Y  # find rows that meet condition  data\_set[data\_set$Y > 40]  filter(data\_set, Y > 300)  filter(data\_set, Y != "NA") | | | # arrange rows by variable  arrange(data\_set, Y)  # creates data frame from csv file  data\_set <- read.csv("file\_name", header = TRUE)  # convert quantitative variable  # to categorical  factor(data\_set$Y)  factor(data\_set$Y, levels = c(1,2), labels = c("A", "B"))  # transform values  recode(data\_set$Y, "0" = 0, "1" = 50, "2" = 100)  # creates two equal sized groups  ntile(data\_set$Y, 2)  # convert categorical variable  # to quantitative as.numeric(data\_set$Y) | |
| Probability Distribution  # calculate the probability area  xpnorm(65.1, data\_set$mean, data\_set$sd)  zscore(data\_set$Y)  # returns t at this probability  qt(.975, df = 999)  # returns F at this probability  qf(.95, df1 = 1, df2 = 100)  # CI using t distribution confint(empty\_model) # calculate p-value using F-distribution  xpf(sample\_F, df1 = 2 , df2 = 10) |
| Simulation # sample without replacement  sample(data\_set, 6)  # sample with replacement  resample(data\_set, 10)  do(3) \* resample (data\_set, 10)  # mixes up values in a variable  shuffle(data\_set$Y)  # simulate sampling 10000 Ys  # from normal distribution  sim\_Y <- rnorm(10000, Y\_stats$mean, Y\_stats$sd)  # put simulated Ys into dataframe  data\_set<- data.frame(sim\_Y)  # simulate sampling distribution of means  sim\_SDoM <- do(10000) \* mean(rnorm(157, Y\_stats$mean, Y\_stats$sd))  # bootstrap sampling distribution of means  bootSDoM <- do(10000) \* mean(resample(data\_set$Y, 157))    # randomize sampling distribution  # of b1s, centered on 0  sdob1 <- do(1000) \*  b1(shuffle(Y) ~ X, data = data\_set) | # bootstrap sampling distribution of b1s,  # centered on sample b1  sdob1\_boot <- do(1000) \*  b1(Y ~ X, data = resample(data\_set))  # count the number of b1s at the upper  # and lower extreme  tally(sdob1$b1 > sample\_b1 |  sdob1$b1 < -sample\_b1)  # return TRUE for middle 95% of distribution  middle(sdob1$b1, .95)  # randomize sampling distribution of PREs  sdoPRE <- do(1000) \* PRE(shuffle(Y) ~ X,  data = data\_set)  # randomize sampling distribution of Fs  sdoF <- do(1000) \*  fVal(shuffle(Y) ~ X, data = data\_set)  # counts extreme Fs  tally(~fVal > sample\_F, data = sdoF) | | Fitting and Evaluating Models # empty model  empty\_model <- lm(Y ~ NULL,  data = data\_set)  # use one expanatory variable  one\_model <- lm(Y ~ X, data = data\_set)  # create a function from a formula  one\_model\_fun <- makeFun(one\_model)  one\_model\_fun(x\_level\_1)  # model predictions and residuals  data\_set$empty\_predict <- predict(empty\_model)  data\_set$empty\_resid <- resid(empty\_model)  # produce ANOVA table  anova(empty\_model)  supernova(one\_model)  # t-test, using pooled variance  t.test(Tip ~ Condition, data = data\_set, var.equal=TRUE)  # pairwise comparison  # corrections: "Bonferroni" or "none" pairwise(one\_model, correction = "none") | | |
| Visualizations gf\_histogram(~ Y, data = data\_set) %>%  # change labels    gf\_labs(title = "Graph Title", x = "Y\_Name", y = "Frequency")  Chart, histogram  Description automatically generated  # faceted grid of histograms  gf\_histogram(~ Y, data = data\_set) %>%    gf\_facet\_grid(X ~ .)  Chart, histogram  Description automatically generated  gf\_dhistogram(~ Y, data = data\_set, fill = "orange", color = "slategray") %>%  gf\_density()  Chart, histogram  Description automatically generated  gf\_bar( ~ Y, data = data\_set)  Chart, bar chart  Description automatically generated | | gf\_boxplot(Y ~ X, data = data\_set)  Chart, box and whisker chart  Description automatically generated  gf\_point(Y ~ X, data = data\_set)  Graphical user interface, chart  Description automatically generated  gf\_jitter(Y ~ X, data = data\_set)  Chart, scatter chart  Description automatically generated  gf\_boxplot(Y ~ X, data = data\_set,  color = "orange") %>%  gf\_jitter(height = 0, color = "grey"),  alpha = .5, size = 3)  Diagram  Description automatically generated | | | # sampling distribution of b1  gf\_histogram(~b1, data = sdob1,  fill = ~middle(b1, .95)) %>%  # modify the limits on x- and y-axes   gf\_lims(x = c(-12, 12), y = c(0, 70))  Chart, histogram  Description automatically generated  gf\_point(Y ~ X, data = data\_set) %>%  # add model predictions as red points   gf\_point(Y ~ X , shape = 1, size = 3,  color = "firebrick") %>%  # add best fitting model as a red line   gf\_model(one\_model, color = “red”)  Chart, scatter chart  Description automatically generated  pairwise(one\_model, plot = TRUE)  Chart, box and whisker chart  Description automatically generated |