R Functions

- grep(pattern, data)
  - Searches data for specific pattern, can find the index
- list()
- matrix()
- which()
  - Gets indices of a specific item
- select()
- replace()
- sort()
- read.table() or read.csv()

Random Variables

- d, p, q, r
  - Using different distributions (norm, binorm, chi-squared)
  - t-distribution (used in hypothesis testing)
    - dt(x, df, ncp, log = FALSE)
    - pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
    - qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
    - rt(n, df, ncp)
- qqplot
  - Plot that compares distributions
- qqnorm
  - Plot that compares a distribution to the normal
- edfun
  - Creates an empirical distribution (in this example: to represent 1000 random numbers generated from a chi-squared distribution)
  - install.packages('edfun')
  - library(edfun)
  - x <- rchisq(n=1000, df=100)
  - x_dist <- edfun(x)
- Showing how to prove law of large numbers and CLT
  - Increase n = prove law of large numbers
  - Increase df (i.e. increase distributions) = prove CLT

Hypothesis Testing

- t-test
  - t.test(x=rnorm(100, mean=1.5), mu=2, alternative ="less")
    - output = p-value
    - If p-value < alpha → significant

- z-test
  - Compare to normal, therefore do d, p, q, r
  - Use power
    - install.packages('pwr')
    - library(pwr)
    - pwr.norm.test(d=1/2, sig.level = 0.05, n = 25, alternative = "greater")
    - # power = 0.804
- chi-squared test
  - chisq.test
- Permutation test
  - install.packages('perm')
  - library(perm)
  - permTS(a, b, alternative = c("two.sided", "less", "greater"))
    - permTS performs two sample permutation test

<u>Statistical Inferences</u>

- library("stats4")
- mle
  - Use mle to solve for unknown parameters
  - # the likelihood
  - nLL <- function(dof) -sum(dchisq(x, df = dof,log=TRUE))
  - dof_estimation1 <-mle(nLL,start = list(dof=1), method = "L-BFGS-B")
  - # The degrees of freedom estimation is 4.905831 (close to 5)
  - dof_estimation1
- mle variations
  - BFGS = default
  - L-BFGS-B = if you know the lower and upper bounds
  - Brent = if you only have one parameter
  - SANN = when we have 3+ unknowns
- Design mle to solve a linear regression
  - Only known is epsilon (error rate)
    - Epsilon → mean = 0 and standard deviation = unknown
  - Use SANN

<u>Linear Regression</u>

- lm
  - lm(formula = y~x)
- glm
- summary()
- Training and testing
  - :(

Power

- library(“pwr”)
  - pwr.norm.test()
  - pwr.t.test()
  - Different from power such that alternative uses “two.sided”, “less”, and “greater”.
- power.t.test
  - SD is an option here. (if you want to change sd use power not pwr)
  - No power.norm.test()
  - Different from power such that alternative uses “one.sided” and “two.sided”.
- 1 - pnorm(-(u1-u0)/(σ/sqrt(n)) + qnorm(1-α))