Intermediate Statistics CA RStudio Notes

Welcome to MA 331 Spring 2023!

These notes are meant to help you succeed in this course. These will include necessary tips for *R* formulas you will need throughout this course. Feel free to give feedback on these via our emails: akassymo@stevens.edu, aleather@stevens.edu.

These notes are not meant to replace the slides. For more info, please refer to the slides.

I highly suggest installing *R Markdown*! It will make your life so much easier. Here's a great guide how to install *R Markdown*.

Lecture 1

R formulas are **very** similar to Python. If you ever used *pandas* or *scipy*, you will easily get into *RStudio*.

To separate code blocks from text in RMarkdown, you need to type:

```
```{r}

*code here*

```
```

You have to close the code block if you want to have normal text after.

Some useful R formulas create a variable: `x = ...`

Tip: if you want to see what is stored in your variable, just type its name.

create a vector: `vector1 = c(x1, ..., xn)`

length of a vector: `length(v1)`

create a table / vector of vectors: `table1 = cbind(v1, ..., vn)`

Tip: if you want to see what is stored in your vector, just type its name.

five-number summary: `summary(x)`

mean of a sample: `mean(x)`

median of a sample: `median(x)`

standard deviation of a sample: `sd(x)`

variance of a sample: `var(x)`

correlation coefficient between two samples: `cor(x,y)`

> **Tip**: a good practice would be to store these statistics in a separate variables. Let's say you test effectiveness of some product and store the data in a vector `data`. Use one-liners like `dataMean = mean(x)` or `dataSD = sd(x)` in the beginning of each Markdown file so you could always access those.

scatterplot: `plot(x)`

boxplot, histogram, pie-chart: `boxplot(x), hist(x), pie(x)`

Normal Distribution

cdf of a normal distribution: `pnorm(x, mean, sd)`

quantile of a normal distribution: `qnorm(q, mean, sd)`

normal QQ plot: `qqnorm(x)`

detrended QQ plot: `qqline(x)`

> **Tip**: you can always see the history of your console in the upper right corner, so if you ever need to remember a command you used last time you opened *RStudio*, you can always see it there.

Lecture 2

In *R Markdown* you can easily format math formulas! To do that, you just have to use $ signs from both sides of an equation. There are a couple more cool things you could do inside the math equation:

1. Use an underscore _{<index>} to add an index to your equation: `$H_{0}$` would give you H_0.
2. Use the tag \overline{<equation>} to add a bar to your equation: `$\overline{X}$` would give you $\overline{X}$. **Don't forget the figure parentheses**.

3. You can directly type the greek letters in the formula using a backslash: `$\sigma$` would give you σ. If you need the capitalized letter Σ, just change it to `$\Sigma$`.
4. To do fractions, type \frac{numerator}{denominanor}, so `$\frac{1}{n}$` would give you $\frac{1}{n}$.

   Tip: Use copy-paste. At first, this might seem tedious, but it's really easy when you get the grasp of it.

A very cool thing *R* can do is to generate a random sample of trials.

To generate a random sample on a **normal** distribution: `rnorm(sample size, mean, sd)`

Binomial Distribution

cdf of a binomial distribution: `pbinom(x, number of trials, probability of a successful trial)`

quantile of a binomial distribution: `qbinom(q, number of trials, probability of a successful trial)`

pmf of a binomial distribution: `dbinom(x, number of trials, probability of a successful trial)`

> Be aware that pmf is not pdf! pmf (probability mass function) is used to describe discrete probability distributions while pdf (probability density functions) is used to describe continuous probability distributions.

To generate a random sample on a binomial distribution: `rbinom(number of observations, number of trials, probability of a successful trial)`

> **Tip**: At this point, you probably have noticed that both normal and binomial distributions have similar *R* function calls. For example, cdf always starts with a `p..` and quantile always starts with `q...` This pattern will repeat in the future!

Chi-square distribution

Chi-square distribution is a sum of squares of random variables. With chi-square distribution, you are introduced to a new parameter - degrees of freedom *df*.

In the case of chi-square distribution, $df = \mu$.

cdf of a chi-square distribution: `pchisq(x, df)`

quantile of a chi-square distribution: `qchisq(q, df)`

Student τ-distribution

T-distribution is a ratio between a normal and a chi-square distribution.

In the case of t-distribution, $df = n - 1$, $\mu = 0$ and $var = \frac{n}{n-2}$ for $n > 2$.

cdf of a t-distribution: `pt(x, df)`

quantile of a t-distribution: `qt(q, df)`

F-distribution

F-distribution is a ratio between to chi-square distributions. Because of that, it has two *df* parameters.

In the case of F-distribution, $\mu = \frac{df_2}{df_2 - 2}$ for $m > 2$.

cdf of an F-distribution: `pf(x, df1, df2)`

quantile of an F-distribution: `qf(q, df1, df2)`

Note: df_1 refers to the degrees of freedom of the chi-square distribution in the numerator and df_2 refers to the degrees of freedom in the denominator.

> At this point, you might be really overwhelmed with the amount of theory. **I understand how you might feel**, but later all of these will start making sense. Each one of these distributions has a use, and you will be introduced to them shortly.

Lecture 3

Confidence levels **are usually 95%**, if not stated otherwise.

Confidence interval with a known σ

To find a confidence interval of a population mean with a **known** standard deviation (z-interval), there is a very easy procedure:

1. Let's say, you're given a vector `data`, standard deviation `sd` and confidence level `cf`.
2. Find the sample size of your dataset: `n = length(data)`.
3. Given confidence level `cf`, your quantile alpha is `alpha = 1 - cf`.
4. Then your z-interval is in between bounds 1 and 2, where: `bound1, bound2 = mean(data) ± qnorm(alpha / 2) * sd / sqrt(n)`.
5. `zinterval = c(bound1, bound2)`.

Confidence interval with an unknown σ

If you don't known standard deviation of your population, you can use sample variance as an **estimate** of your population variance. This is the main trick to get t-interval:

1. You're given a dataset `data` and confidence level `cf`.
2. Find the sample size of your dataset: `n = length(data)`.
3. Given confidence level `cf`, your quantile alpha is `alpha = 1 - cf`.
4. Find the sample standard deviation and sample mean: `sampleSD = sd(data)`.
5. Then your t-interval is in between bounds 1 and 2, where: `bound1, bound2 = mean(data) ± qt(alpha / 2, n - 1) * sampleSD / sqrt(n)`.
6. `tinterval = c(bound1, bound2)`.

   Tip: you can reassign a new value to a variable like `bound1`, *R Markdown* follows the order of commands in which they're typed in.

Confidence interval irrespective of μ

In the case, where we don't have both population mean and variance and we would like to find a confidence interval for the population variance, we can find chi-interval. We are using sample variance as an **estimate** for our population variance.

1. You're given a dataset `data` and confidence level `cf`.
2. Find the sample size of your dataset: `n = length(data)`.
3. Given confidence level `cf`, your quantile alpha is `alpha = 1 - cf`.
4. Find the sample variance: `samplevar = var(data)`.
5. Then your chi-interval is in between bounds 1 and 2, where: `bound1 = (n - 1) * samplevar / qchisq(1 - alpha / 2, n - 1)` & `bound2 = (n - 1) * samplevar / qchisq(alpha / 2, n - 1)`.
6. `chinterval = c(bound1, bound2)`.

Population proportion interval

Given a dataset `data` we could convert it to binary with: `bdata = ifelse(data > ref, 1, 0)`, where `ref` is a reference point. Example: given a dataset `grades`

```
= [98, 56, 76, 47, 86]:

passorfail = ifelse(grades > 70, 1, 0)
passorfail

[1] 1, 0, 1, 0, 1
```

To create a population proportion interval, given the binary dataset`bdata`:

1. Find the sample size of your dataset: `n = length(bdata)`.
2. Given confidence level `cf`, your quantile alpha is `alpha = 1 - cf`.
3. Find the sample proportion: `phat = mean(data)`.
4. Then your prop-interval is in between bounds 1 and 2, where: `bound1, bound2 = phat ± qnorm(alpha / 2) * sqrt(phat * (1 - phat)) / sqrt(n)`.
5. `propinterval = c(bound1, bound2)`.

Lecture 4

H_0 is the null hypothesis and proposes that any difference between groups is due to statistical chance.

H_a is the alternative hypothesis and contradicts the null hypothesis.

The null and alternative hypotheses must always be stated before a hypothesis test.

The goal of a hypothesis test to either reject or fail to reject the null hypothesis given a sample of the population. To come to a conclusion about the null hypothesis, we must compare the *p-value* p of the sample with the significance level α of the test.

The *p value* of a sample is the probability that you obtain a sample that is at least as extreme as the one obtained.

When $p < \alpha$, we **reject** the null hypothesis in favor of the alternative hypothesis.

When $p > \alpha$, we **fail to reject** the null hypothesis.

The alternative hypothesis can either be **two-tailed** $\mu \neq \mu_0$, **left-tailed** $\mu < \mu_0$, or **right-tailed** $\mu > \mu_0$.

Hypothesis test of a population mean with known σ

When we wish to conduct a hypothesis test of a population mean when population standard deviation is known, we will use a z-test.

1. First, we must identify the assumed mean of the population `mu`, mean on the sample `xbar`, sample size `n`, and population standard deviation `sd`.
2. Now we calculate the z-score of the sample using the formula `z = sqrt(n) * (xbar - mu) / sigma`.
3. Then we calculate the p-value of the sample based on the z-score.
    a. for two-sided tests, we can use `p = 2 * pnorm( -abs(z))`.
    b. for left-tailed tests, we can use `p = pnorm(z)`.
    c. for right-tailed tests, we can use `p = 1 - pnorm(z)`.
4. Lastly, compare p to α and write a proper conclusion.

Hypothesis test of a population mean with unknown σ

When we wish to conduct a hypothesis test of a population mean, but we do not know the population standard deviation, we must use a one sample t-test.

1. We first must have our sample `x`, the side of the test (two-sided, right-tailed, or left-tailed) `side`, the assumed population mean `mu0`, and the confidence level of the test `1 - alpha`
2. To conduct a one sample t test in we can use the function `t.test(x, alternative = c("side"), mu = mu0, conf.level = 1 - alpha`
    a. For the side of the test `side`, use `"two.sided"` for two-sided tests, `"less"` for left-tailed tests, and `"greater"` for right-tailed tests.
    b. If you do not enter a confidence level for the t-test, R will default to a confidence level of 95%.
3. R will automatically compute the p-value for you, so now you just compare p with α and write a proper conclusion.

Hypothesis test of the difference of two population means

When we want to test for a difference of two population means, we can conduct a two sample paired t-test.

1. We first need our two samples `x` and `y`, the side of the test `side`, and the confidence level of the test `1 - alpha`
2. We can use the R function `t.test(x, y, alternative = c("side"), paired = TRUE, conf.level = 1 - alpha`

a. We do not need to enter a mean for the difference between the population means, since under the null hypothesis, we assume there is no difference between the means $H_0 : \mu_x - \mu_y = 0$.

b. For the side of the test `side`, use `"two.sided"` for two-sided tests, `"less"` for left-tailed tests, and `"greater"` for right-tailed tests.

c. If you do not enter a confidence level for the t-test, R will default to a confidence level of 95%.

3. R will calculate the p-value, so now you compare p with $alpha$ and write a proper conclusion.

Hypothesis test of a population proportion

When conducting a hypothesis about a population of a proportion, we use a z-test.

1. We first need a binary sample `x` where "1" indicates a success and "0" indicates a failue (refer to Population proportion intervals in Lecture 3 on how to convert a data set to binary)

2. We must also know the assumed population proportion `p0`, the side of the test `side`, and the confidence level `1 - alpha`

3. To run the test in R, we can use the funciton `prop.test = (sum(x), length(x), p = p0, alternative = c("side"), conf.level = 1 - alpha)`

   a. For the side of the test `side`, use `"two.sided"` for two-sided tests, `"less"` for left-tailed tests, and `"greater"` for right-tailed tests.

   b. If you do not enter a confidence level for the t-test, R will default to a confidence level of 95%.