Week 4 Cheat Sheet

Statistics and Data Analysis with R

Course Link: https://www.coursera.org/learn/statistics-and-data-analysis-with-r/

Charlie Nuttelman

Here, I provide the functions in R required to perform various calculations in Week 4 of the course. The headings represent the screencasts in which you will find those concepts and examples.

Testing the Assumption of Normality

This is not a screencast in itself, but many of the screencasts/topics below depend on normal or near-normal populations. In other words, we assume normality in many of the methods and techniques below. To test for the assumption of normality, we can do the following (note that the **ad.test** function requires a sample vector of at least size 8):

1. Load the **nortest** library: **library(nortest**) (if this library is not installed, install it using **install.packages("nortest")**.
2. Calculate the P-value of the Anderson-Darling (AD) statistic of the data in vector **x** using **ad.test(x)**.
3. To store the P-value of the AD statistic as a variable: `P <- ad.test(x)$p.value`
4. If the P-value of the AD statistic is *greater* than 0.05, then we can assume that the data in vector **x** come from a normally distributed population; if the P-value of the AD statistic is *not greater* than 0.05, then we can assume that the data in vector **x** do not come from a normally distributed population.

Confidence Interval on the Mean, Variance Known

In R, we can calculate $(1-\alpha)\cdot 100\%$ confidence intervals on the mean of a population when the variance is known using the following methods:

- Using mathematical formulas:
    - Given a vector of our data (`x`), a significance level (`alpha`), the known population standard deviation (`sigma`), and the population size (`n`), we can use mathematical formulas in R to calculate the confidence interval (`CI`). One such example is:

```
xbar <- mean(x)
z <- qnorm(alpha/2, lower.tail=FALSE)
CI <- xbar + c(-1,1)*z*sigma/sqrt(n)
CI
```

    - Another option is as follows:

```
error <- z*sigma/sqrt(n)
lower <- xbar-error
```

```
upper <- xbar+error
cat(" Upper limit =", upper, "\n","Lower limit
=",lower)
```

- Create a user-defined function that uses mathematical formulas, where `data` is our vector containing our sample data and `var` is the known population variance:

```
CI.z <- function(data, var, alpha=0.05) {
  n <- length(data)
  sigma <- sqrt(var)
  xbar <- mean(data)
  z <- qnorm(alpha/2, lower.tail=FALSE)
  error <- z*sigma/sqrt(n)
  lower <- xbar-error
  upper <- xbar+error
  cat(" Upper limit =", upper, "\n","Lower limit =",lower)
}
```

- **z.test** (TeachingDemos)
  - Given sample data in vector `x`, we can calculate a confidence interval on the population mean using `z.test(x,sd)`. For example, if the true population standard deviation is 40, we would use in R: `z.test(x,sd=40)` (note the explicit use of "sd=…")
  - The confidence interval is found in the output following the "`95 percent confidence interval`" line (note that other information here is not necessarily correct when we use the **z.test** function in this manner).
  - If you wish to use a different confidence interval, for example 90%, we can add "`conf.level=0.90`" as an optional argument to the **z.test** function: `z.test(x,sd=40,conf.level=0.90).`
  - To store the lower and upper bounds, respectively, of the confidence interval output by the **z.test** function, for example when the population standard deviation is known to be 40, we can use:
    - `low <- z.test(x,sd=40)$conf.int[1]`
    - `high <- z.test(x,sd=40)$conf.int[2]`
  - NOTE: If you have the BSDA library installed when you try to run the **z.test** function, it will not work properly. Unclick **BSDA** from the **Packages** pane in the lower right of RStudio and re-run the **z.test** function and it should work properly.
- **zsum.test** (BSDA) – useful when you only have the sample statistics and not the actual data
  - Given sample average and known population standard deviation (sigma), we can use the **zsum.test** function in the **BSDA** library to calculate a 95% confidence interval on the true population mean. For example, if the sample average is 127.5, the population standard deviation is known to be 40, and the sample size is 10, we could use in R: `zsum.test(mean.x=127.5, sigma.x=40, n.x=10, alternative="two.sided")`
  - The confidence interval is found in the output following the "`95 percent confidence interval`" line (note that other information here is not necessarily correct when we use the **zsum.test** function in this manner).
  - If you wish to use a different confidence interval, for example 90%, we can add "`conf.level=0.90`" as an optional argument to the **zsum.test** function:

```
zsum.test(mean.x=127.5, sigma.x=40, n.x=10,
alternative="two.sided",conf.level=0.90)
```

- To store the lower and upper bounds, respectively, of the confidence interval output by the **zsum.test** function, for example when the sample mean is 127.5, the population standard deviation is known to be 40, and the sample size is known to be 10, we can use:
  - `low <- zsum.test(mean.x=127.5, sigma.x=40, n.x=10, alternative="two.sided")$conf.int[1]`
  - `high <- zsum.test(mean.x=127.5, sigma.x=40, n.x=10, alternative="two.sided")$conf.int[2]`

Confidence Interval on the Mean, Variance Unknown

In R, we can calculate $(1-\alpha)\cdot 100\%$ confidence intervals on the mean of a population when the variance is unknown using the following methods:

- Using mathematical formulas:
  - Given a vector of our data (`x`) and a significance level (`alpha`), we can use mathematical formulas in R to calculate the confidence interval (`CI`). One such example is:

```
n <- length(x)
s <- sd(x)
xbar <- mean(x)
t <- qt(0.05/2,n-1,lower.tail=FALSE)
lower <- xbar-t*s/sqrt(n)
upper <- xbar+t*s/sqrt(n)
cat(" Upper limit=",upper,"\n","Lower limit=",lower)
```

  - You could also create a user-defined function as we did in the section above to calculate a confidence interval when the variance is unknown.
- **t.test** (stats)
  - Given sample data in vector `x`, we can calculate a confidence interval on the population mean using `t.test(x)`.
  - The confidence interval is found in the output following the "`95 percent confidence interval`" line (note that other information here is not necessarily correct when we use the **t.test** function in this manner).
  - If you wish to use a different confidence interval, for example 90%, we can add "`conf.level=0.90`" as an optional argument to the **t.test** function: `t.test(x,conf.level=0.90)`.
  - To store the lower and upper bounds, respectively, of the confidence interval output by the **t.test** function, we can use:
    - `low <- t.test(x)$conf.int[1]`
    - `high <- t.test(x)$conf.int[2]`
- **lm** (stats)
  - We can use the **lm** (linear model) function in the **stats** library to provide a confidence interval on the population mean when variance is unknown.

- Given sample data in vector `x`:

```
mod <- lm(x~1)
mod
CI <- predict(mod,interval="confidence")
CI[1,]
```

- If we want a different confidence level, the third of the four lines above can be modified to:

```
CI <- predict(mod,interval="confidence", level=0.9)
```

- **tsum.test** (BSDA) – useful when you only have the sample statistics and not the actual data
  - Given sample average and sample standard deviation, we can use the **tsum.test** function in the **BSDA** library to calculate a 95% confidence interval on the true population mean. For example, if the sample average is 416.33, the sample standard deviation is 17.83, and the sample size is 12, we could use in R: `tsum.test(mean.x=416.33, s.x=17.83, n.x=12, alternative="two.sided")`
  - The confidence interval is found in the output following the "`95 percent confidence interval`" line (note that other information here is not necessarily correct when we use the **tsum.test** function in this manner).
  - If you wish to use a different confidence interval, for example 90%, we can add "`conf.level=0.90`" as an optional argument to the **tsum.test** function: `tsum.test(mean.x=416.33, s.x=17.83, n.x=12, alternative="two.sided",conf.level=0.90)`
  - To store the lower and upper bounds, respectively, of the confidence interval output by the **tsum.test** function, for example when the sample mean is 416.33, the sample standard deviation is 17.83, and the sample size is 12, we can use:
    - `low <- tsum.test(mean.x=416.33, s.x=17.83, n.x=12, alternative="two.sided")$conf.int[1]`
    - `high <- tsum.test(mean.x=416.33, s.x=17.83, n.x=12, alternative="two.sided")$conf.int[2]`

Prediction Interval on a Future Observation

In R, we can a calculate $(1-\alpha)\cdot 100\%$ prediction interval on a single future observation from a population based on a sample vector **x** using the following methods:

- Using mathematical formulas
  - Variance known – Here, I'm using an example where α = 0.05 and the population standard deviation is 40:

```
alpha <- 0.05
xbar <- mean(x)
sigma <- 40
n <- length(x)
z <- qnorm(alpha/2, lower.tail=FALSE)
E <- z*sigma*sqrt(1+1/n)
PI <- xbar+c(-1,1)*E
PI
```

- Variance unknown – Here, I'm using an example where α = 0.05 and the population standard deviation is unknown:

```
alpha <- 0.05
xbar <- mean(x)
s <- sd(x)
n <- length(x)
t <- qt(alpha/2, n-1, lower.tail=FALSE)
E <- t*s*sqrt(1+1/n)
PI <- xbar+c(-1,1)*E
PI
```

  - To store the lower and upper bounds, we can just use:

```
low <- PI[1]
high <- PI[2]
```

- **lm** (stats) – this function can be used in combination with the **predict** function to calculate a prediction interval on a future observation when the variance is unknown:

```
mod <- lm(x~1)
PI <- predict(mod, interval="predict")
PI[1,]
```

  - We can use a different confidence level, for example 90%:

```
mod <- lm(x~1)
PI <- predict(mod, interval="predict",level=0.9)
PI[1,]
```

  - To store the lower and upper bounds, we can just use:

```
low <- PI[1,2]
high <- PI[2,3]
```

Hypothesis Tests on the Mean, Variance Known

In R, we can use the following methods to perform hypothesis tests on the mean when the variance is known:

- Using mathematical formulas
  - For a lower-tailed hypothesis test (H_0: $\mu = \mu_0$; H_1: $\mu < \mu_0$), we can use the following code in R to perform a hypothesis test, where **sigma** is the known standard deviation, **mu0** is μ_0 in the hypothesis test, **alpha** is the significance level (typically 0.05), and **x** is a vector containing our sample data:

```
z0 <- (mean(x)-mu0)/sigma*sqrt(length(x))
z <- qnorm(alpha)
P <- pnorm(z0)
P
```

    Note that it's not necessary to calculate **z**, which is the critical z-value for the test; I've done it here just for completeness. All we need is **P**, which is the P-value of the test. If **P** is less than **alpha**, then we can accept the alternate hypothesis. Otherwise, we reject the alternate hypothesis and fail to reject the null hypothesis.

- For an upper-tailed hypothesis test (H_0: $\mu = \mu_0$; H_1: $\mu > \mu_0$), we can use the following code in R to perform a hypothesis test, where **sigma** is the known standard deviation, **mu0** is μ_0 in the hypothesis test, **alpha** is the significance level (typically 0.05), and **x** is a vector containing our sample data:

```
z0 <- (mean(x)-mu0)/sigma*sqrt(length(x))
z <- qnorm(alpha,lower.tail=FALSE)
P <- pnorm(z0, lower.tail=FALSE)
P
```

  Note that it's not necessary to calculate **z**, which is the critical z-value for the test; I've done it here just for completeness. All we need is **P**, which is the P-value of the test. If **P** is less than **alpha**, then we can accept the alternate hypothesis. Otherwise, we reject the alternate hypothesis and fail to reject the null hypothesis.

- **z.test** (TeachingDemos)
  - We can use this function to perform a hypothesis test on a vector of sample data, **x**, when we know the population standard deviation. For example, if we'd like to perform an upper-tailed test, we can use the following (H_0: $\mu = 100$; H_1: $\mu > 100$) with known population standard deviation of 40:

```
z.test(x, mu=100, stdev=40, alternative="greater")
```

  - The output provides the test statistic (labeled simply as "z") and the P-value for the test, which the user can compare to a significance level (i.e., $\alpha = 0.05$).
  - For a lower-tailed test, we can simply replace **alternative="greater"** with **alternative="less"**. For a two-tailed test, we can use **alternative="two.sided"**.
  - NOTE: If you have the BSDA library installed when you try to run the **z.test** function, it will not work properly. Unclick **BSDA** from the **Packages** pane in the lower right of RStudio and re-run the **z.test** function and it should work properly.
- **zsum.test** (BSDA)
  - We can also use the **zsum.test** function from the BSDA library to perform hypothesis tests on the mean when the population variance is known. It is useful when you only have the sample statistics and not the actual data. For example, if we'd like to perform a lower-tailed test, we can use the following (H_0: $\mu = 30$; H_1: $\mu < 30$) knowing that the sample average is 27.5, the population standard deviation is 5.1, and the sample size is 13:

```
zsum.test(mean.x=27.5, sigma.x=5.1, n.x=13,
alternative="less", mu=30)
```

  - The output provides the test statistic (labeled simply as "z") and the P-value for the test, which the user can compare to a significance level (i.e., $\alpha = 0.05$).
  - For an upper-tailed test, we can simply replace **alternative="less"** with **alternative="greater"**. For a two-tailed test, we can use **alternative="two.sided"**.
  - Note that if you have the data, for example in vector **x**, you can also calculate the mean and size inside the **zsum.test** function, for example:

```
zsum.test(mean.x=mean(x), sigma.x=5.1, n.x=length(x),
alternative="less", mu=30)
```

Hypothesis Tests on the Mean, Variance Unknown

In R, we can use the following methods to perform hypothesis tests on the mean when the variance is unknown:

- Using mathematical formulas
    - For a lower-tailed hypothesis test ($H_0: \mu = \mu_0;\ H_1: \mu < \mu_0$), we can use the following code in R to perform a hypothesis test, where **mu0** is μ_0 in the hypothesis test, **alpha** is the significance level (typically 0.05), and **x** is a vector containing our sample data:

```
t0 <- (mean(x)-mu0)/sd(x)*sqrt(length(x))
t_crit <- qt(alpha, length(x)-1)
P <- pt(t0)
P
```

      Note that it's not necessary to calculate **t_crit**, which is the critical t-value for the test; I've done it here just for completeness. All we need is **P**, which is the P-value of the test. If **P** is less than **alpha**, then we can accept the alternate hypothesis. Otherwise, we reject the alternate hypothesis and fail to reject the null hypothesis.

    - For an upper-tailed hypothesis test ($H_0: \mu = \mu_0;\ H_1: \mu > \mu_0$), we can use the following code in R to perform a hypothesis test, where **mu0** is μ_0 in the hypothesis test, **alpha** is the significance level (typically 0.05), and **x** is a vector containing our sample data:

```
t0 <- (mean(x)-mu0)/sd(x)*sqrt(length(x))
t_crit <- qt(alpha, length(x)-1, lower.tail=FALSE)
P <- pt(t0,lower.tail=FALSE)
P
```

      Note that it's not necessary to calculate **t_crit**, which is the critical t-value for the test; I've done it here just for completeness. All we need is **P**, which is the P-value of the test. If **P** is less than **alpha**, then we can accept the alternate hypothesis. Otherwise, we reject the alternate hypothesis and fail to reject the null hypothesis.
- **t.test** (stats)
    - We can use this function to perform a hypothesis test on a vector of sample data, x, when we don't know the population variance. For example, if we'd like to perform a lower-tailed test, we can use the following ($H_0: \mu = 100;\ H_1: \mu < 100$):

```
t.test(x, mu=100, alternative="less")
```

    - The output provides the test statistic (labeled simply as "t") and the P-value for the test, which the user can compare to a significance level (i.e., $\alpha = 0.05$).
    - For a lower-tailed test, we can simply replace **alternative="greater"** with **alternative="less"**. For a two-tailed test, we can use **alternative="two.sided"**.
    - To store the P-value in the above example as a single variable, we can use:

```
P <- t.test(x, mu=100, alternative="less")$p.value
```

- **tsum.test** (BSDA)
    - We can also use the **tsum.test** function from the BSDA library to perform hypothesis tests on the mean when the population variance is unknown. It is useful when you only have the sample statistics and not the actual data. For example, if we'd like to perform a lower-tailed test, we can use the following ($H_0: \mu = 30;\ H_1: \mu < 30$) knowing that the

sample average is 27.5, the population standard deviation is 5.1, and the sample size is 13:

```
tsum.test(mean.x=27.5, sigma.x=5.1, n.x=13,
alternative="less", mu=30)
```

- The output provides the test statistic (labeled simply as "z") and the P-value for the test, which the user can compare to a significance level (i.e., $\alpha = 0.05$).
- For an upper-tailed test, we can simply replace **alternative="less"** with **alternative="greater"**. For a two-tailed test, we can use **alternative="two.sided"**.
- Note that if you have the data, for example in vector **x**, you can also calculate the mean, sample standard deviation, and size inside the **tsum.test** function, for example:

```
tsum.test(mean.x=mean(x), s.x=sd(x), n.x=length(x),
alternative="less", mu=30)
```

- To store the P-value of the example above in a single variable, we can use:

```
P <- tsum.test(mean.x=mean(x), s.x=sd(x),
n.x=length(x), alternative="less", mu=30)$p.value
```

Type II Error and Power of the Test

I have repeated here my information from the Weak 3 Cheat Sheet of "Statistics and Data Analysis with Excel, Part 2":

A Type II error is failing to accept the alternate hypothesis when it is true. The probability of making a Type II error is known as β, and it depends upon the shift in mean (or difference in mean) between the null and alternate hypotheses.

To calculate β, we must first calculate the critical sample average ($\bar{x}_c$). For an upper-tailed test, we would accept the alternate hypothesis if our sample average were greater than $\bar{x}_c$ and we would reject the alternate hypothesis if our sample average were less than $\bar{x}_c$. For a lower-tailed test, we would accept the alternate hypothesis if our sample average were less than $\bar{x}_c$ and we would reject the alternate hypothesis if our sample average were greater than $\bar{x}_c$.

Once we have identified $\bar{x}_c$, which depends on α, we can calculate β by calculating the area underneath the alternate hypothesis sampling distribution to the left of $\bar{x}_c$ for an upper-tailed test and to the right of $\bar{x}_c$ for a lower-tailed test:

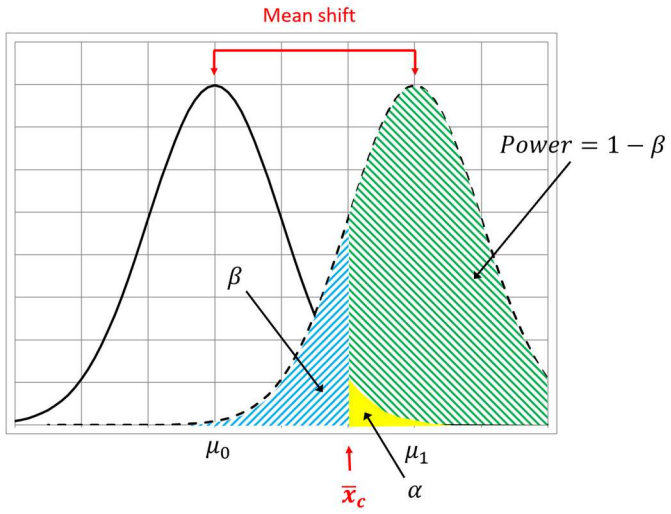
For an upper-tailed test: $\beta = P[\bar{X} < \bar{x}_c \mid \mu_1]$

For a lower-tailed test: $\beta = P[\bar{X} > \bar{x}_c \mid \mu_1]$

For both cases, "$\mid \mu_1$" indicates that we are operating under the alternate sampling distribution (we must know or assume a value for μ_1 in order to calculate a numerical value for β).

For an upper-tailed test (graphically):

H_0: $\mu = \mu_0$
H_1: $\mu > \mu_0$



We can calculate type II error (also known as β) and power of the test in R using the following options:

- Using mathematical formulas (in all the following bullet points, **alpha** is the type I error risk, **mu0** is μ_0, **sigma** is the known population standard deviation, **n** is the sample size, and **mu1** is μ_1):
  - For a lower-tailed test, variance known:
    ```
    xbarc <- qnorm(alpha,mean=mu0,sd=sigma/sqrt(n),
    lower.tail=TRUE)
    beta <-
    pnorm(xbarc,mean=mu1,sd=sigma/sqrt(n),lower.tail=FALSE)
    power <- 1-beta
    ```
  - For an upper-tailed test, variance known:
    ```
    xbarc <- qnorm(alpha,mean=mu0,sd=sigma/sqrt(n),
    lower.tail=FALSE)
    beta <-
    pnorm(xbarc,mean=mu1,sd=sigma/sqrt(n),lower.tail=TRUE)
    power <- 1-beta
    ```
  - NOTE: The above approach (using mathematical equations) cannot be performed using similar equations when the variance is unknown since the analysis is based on the noncentral T distribution. Instead, it's best to use the **pwr.t.test** function (see below).
- **pwr.norm.test** (pwr) – Use this when the variance is known
  ```
  pwr.norm.test(d=(mu1-mu0)/sigma, n=8, alternative="less")
  ```
  - Use **alternative="greater"** for upper-tailed tests
- **pwr.t.test** (pwr) – Use this when the variance is unknown
  ```
  pwr.t.test(n=length(x),d=(mu1-
  mu0)/s,alternative="less",type="one.sample")
  ```
  - Use **alternative="greater"** for upper-tailed tests
- To store the power from the **pwr.t.test** function as a single variable, we can use (extensions of the example above; similar for the **pwr.norm.test**):
  ```
  power <- pwr.t.test(n=n,d=(mu1-
  mu0)/s,alternative="greater",type="one.sample")$power
  ```

Choice of Sample Size

In R, we can use the following methods to calculate the minimum sample size required for a certain desired power of the test:

- Using mathematical formulas (for variance known case only); note that **beta** is the *desired* type II error (corresponds to a power of 0.8 in this example):

```
alpha <- 0.05
beta <- 0.1
z_alpha <- qnorm(alpha, lower.tail=FALSE)
z_beta <- qnorm(beta, lower.tail=FALSE)
delta <- mu1-mu0
n <- (z_alpha+z_beta)^2*sigma^2/delta^2
```

- **pwr.norm.test** (pwr) – Use this when the variance is known

```
pwr.norm.test(d=(mu1-mu0)/sigma, power=0.9, sig.level=0.05,
alternative="less")
```

- **pwr.t.test** (pwr) – Use this when the variance is unknown (this example is for desired power of 0.8):

```
d <- (mu0-mu1)/s
pwr.t.test(d=d, power=0.8, type="one.sample",
alternative="less")
```

  - Use **alternative="greater"** for upper-tailed tests
- In all cases, round the sample size (**n**) up to the nearest integer to be sure that you achieve the desired power level. We can do this using the **ceiling** function in R (see example that follows).
- NOTE: We can extract the sample size from **pwr.norm.test** and **pwr.t.test** function results by using the following (extensions of the examples above) and round the result up to the nearest integer:

```
n <- ceiling(pwr.norm.test(d=(mu1-mu0)/sigma, power=0.9,
sig.level=0.05, alternative="less")$n)
```

OR

```
n <- ceiling(pwr.t.test(d=d, power=0.8, type="one.sample",
alternative="less")$n)
```

Confidence Interval on the Variance

In R, we can use the following methods to calculate confidence intervals on the variance:

- Using mathematical formulas
  - Given our risk level (**alpha**) and our sample vector, **x**, we can determine a $(1-\alpha)\cdot 100\%$ confidence interval on the true population variance, σ^2, using the following lines of code in R:

```
s <- sd(x)
n <- length(x)
alpha <- 0.10
lower <- (n-1)*s^2/qchisq(alpha/2, n-1,
lower.tail=FALSE)
```

```
upper <- (n-1)*s^2/qchisq(alpha/2, n-1,
lower.tail=TRUE)
cat( "Upper limit =", upper, "\n", "Lower limit =",
lower)
```

  - There are many other ways to do this in R; the above is just one such example.
- **VarCI** and **VarTest** (DescTools)
  - The **VarCI** function only provides the confidence interval:
    ```
    VarCI(x)
    ```
  - The **VarTest** function also performs a hypothesis test on the variance, but you can ignore the rest of the output other than the confidence intervals in the output provided by the VarTest function if we use the function in the following manner:
    ```
    VarTest(x)
    ```
  - By default, the confidence level of the **VarCI** and **VarTest** functions is 0.95 (i.e., α = 0.05), but if you'd like to use a different significance level (for example, 90% shown here), you can use the following:
    ```
    VarCI(x, conf.level=0.9)
    VarTest(x, conf.level=0.9)
    ```
- **varTest** (EnvStats)
  - The **varTest** function (notice the lowercase "v") in the **EnvStats** library is nearly identical to the **VarTest** function of the **DescTools** library; the only difference is in the format of the output:
    ```
    varTest(x)
    ```
  - Note that, as with the **VarTest** function above, this function performs a hypothesis test on the variance – ignore the output if you are just interested in the confidence interval of the variance. We'll revisit this function below when we perform hypothesis tests on the variance.
  - The confidence level can be changed similarly to the **VarTest** function above.

Hypothesis Tests on the Variance

In R, we can use the following methods to perform hypothesis tests on the variance:

- Using mathematical formulas – useful when all you have are the sample statistics
  - For a lower-tailed test (H_0: $\sigma^2 = \sigma_0^2$; H_1: $\sigma^2 < \sigma_0^2$), where **s** is the sample standard deviation, **n** is the sample size, and **sigma0** is σ_0:
    ```
    chisq_0 <- (n-1)*s^2/sigma0^2
    P <- pchisq(chisq_0, n-1)
    ```
  - For an upper-tailed test (H_0: $\sigma^2 = \sigma_0^2$; H_1: $\sigma^2 > \sigma_0^2$), where **s** is the sample standard deviation, **n** is the sample size, and **sigma0** is σ_0:
    ```
    chisq_0 <- (n-1)*s^2/sigma0^2
    P <- pchisq(chisq_0, n-1, lower.tail=FALSE)
    ```
  - Like many of the hypothesis tests we perform, if **P** (the P-value) for either of the above is less than our significance level (α, which is typically 0.10 for tests on the variance), then we can accept the alternate hypothesis.
- **VarTest** (DescTools)

- For a lower-tailed test (H_0: $\sigma^2 = \sigma_0^2$; H_1: $\sigma^2 < \sigma_0^2$), where **x** is a vector of our sample data and **sigma0** is σ_0, we can use:
```
VarTest(x, alternative="less", sigma.squared=sigma0^2)
```
- For an upper-tailed test (H_0: $\sigma^2 = \sigma_0^2$; H_1: $\sigma^2 > \sigma_0^2$), where **x** is a vector of our sample data and **sigma0** is σ_0, we can use:
```
VarTest(x, alternative="greater",
sigma.squared=sigma0^2)
```
- As above, if **P** (the P-value) for either of the above is less than our significance level (α, which is typically 0.10 for tests on the variance), then we can accept the alternate hypothesis.
- To store the P-value as a single variable, we can use (using the lower-tailed example above):
```
P <- VarTest(x, alternative="less",
sigma.squared=sigma0^2)$p.value
```

- **varTest** (EnvStats)
  - The **varTest** function is used exactly the same as the **VarTest** function above; the only difference is in the format of the output in the console.

Hypothesis Tests on a Binomial Proportion

In R, we can use the following methods to perform hypothesis tests on a binomial proportion:

- **pbinom** (stats)
  - For a lower-tailed test (H_0: $p = p_0$; H_1: $p < p_0$), where **x** is the number of items in sample of size **n** that have a certain characteristic (or otherwise are successes) and **p** is the probability of success, we can calculate the P-value (**P**) using the following line of code in R:
```
P <- pbinom(x, n, p)
```
  - For an upper-tailed test (H_0: $p = p_0$; H_1: $p > p_0$), where **x** is the number of items in sample of size **n** that have a certain characteristic (or otherwise are successes) and **p** is the probability of success, we can calculate the P-value (**P**) using the following line of code in R:
```
P <- pbinom(x-1, n, p, lower.tail=FALSE)
```
  - As always, if **P** (the P-value) for either of the above is less than our significance level (α), then we can accept the alternate hypothesis. NOTE: For an upper-tailed test using the **pbinom** function, it is important that we subtract 1 from **x** and use **x-1** as the first argument.
- **binom.test** (stats)
  - We can also perform an exact binomial test in R using the **binom.test** function. For a lower-tailed test (H_0: $p = p_0$; H_1: $p < p_0$), where **x** is the number of items in sample of size **n** that have a certain characteristic (or otherwise are successes) and **p** is the probability of success, we can calculate the P-value using the following line of code in R:
```
binom.test(x, n, p, alternative="less")
```

- For an upper-tailed test (H_0: $p = p_0$; H_1: $p > p_0$), where **x** is the number of items in sample of size **n** that have a certain characteristic (or otherwise are successes) and **p** is the probability of success, we can calculate the P-value using the following line of code in R:

```
binom.test(x, n, p, alternative="greater")
```

- For both of the above options, the output in the console will provide a P-value. If the P-value is less than our significance level (α), then we can accept the alternate hypothesis.
- To store the P-value as a single variable from the **binom.test** function, we can use (for upper-tailed example):

```
P <- binom.test(x, n, p,
alternative="greater")$p.value
```