Week 3 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 3 of the course. The headings represent the screencasts in which you will find those concepts and examples.

Permutations and Combinations

A combination is defined as how many ways that **n** items can be placed into **r** spots. In R, we can use the following functions:

- **choose** (base) – number of combinations
  - **choose(n,r)** provides the number of combinations of **n** items chosen **r** at a time
- **combn** (utils) – lists all combinations
  - **combn(n,r)** lists all the combinations of **n** items chosen **r** at a time
  - **combn(v,r)** lists all the combinations of items in vector **v** chosen **r** at a time
- **combinations** (gtools) – lists all combinations, with and without repeats:
  - **combinations(n,r)** lists all combinations of **n** items chosen **r** at a time (similar to **combn**)
  - **combinations(n, r, repeats.allowed = TRUE)** lists the combinations of **n** items chosen **r** at a time *but repeats are allowed*

A permutation is defined as how many ways that **n** items can be placed into **r** spots but order matters (in other words, AB and BA are the same combination of A and B, but AB and BA are different permutations of A and B). In base R, there is no direct permutation function. Instead, we must use the following techniques and functions:

- **choose** (base) and **factorial** (base) – when multiplied together, we can obtain the number of permutations:
  - **choose(n,r)*factorial(r)** calculates the number of permutations of **n** items taken **r** at a time
- **permn** (combinat) – lists all permutations of **n** items taken **n** at a time (note that we cannot use this function to calculate the number of permutations of **n** items taken **r** at a time):
  - **permn(n)** lists all permutations of the numbers 1 through **n**
  - **permn(v)** lists all permutations of the items in vector **v** taken **n** at a time, where **n** is the length of vector **v**
- **permutations** (gtools) – lists all permutations of **n** items taken **r** at a time
  - **permutations(n,r)** lists all permutations of the numbers 1 through **n** taken **r** at a time
  - **permutations(n, r, repeats.allowed = TRUE)** lists all permutations of the numbers 1 through **n** taken **r** at a time *but repeats are allowed*
  - **permutations(n,r,v)** lists all permutations of **n** items taken **r** at a time from vector **v**.

- **permutations(n, r, v, repeats.allowed = TRUE)** lists all permutations of **n** items taken **r** at a time from vector **v** *but repeats are allowed.*
- In the two preceding bullet points, note that **n** must match the length of vector **v** (it doesn't yield an error – but the result is incorrect). It might be better to get in the habit of using **permutations(length(v),r,v)**, for example.

- **permute** (gtools) – randomly permutes the items in vector **v** into a new vector. Note that if you want to make a permanent, randomized change to vector **v**, you must reassign the result to vector **v**: **v <- permute(v)**

The Binomial Distribution

The binomial distribution is used to calculate the probability of obtaining **x** successes in **r** trials. Trials must meet the requirements of a Bernoulli trial (two possible outcomes, fixed probability of success for each trial, and the outcome of each trial is independent of the result of a previous trial).

In R, we can use the following functions to assist with binomial distribution calculations:

- **dbinom** (stats) – calculates the probability density (mass) function; equivalent to f(x)
  - **dbinom(x,n,p)** calculates the probability of exactly **x** successes in **n** trials given a probability of success of **p**
- **pbinom** (stats) – calculates the cumulative probability density function; equivalent to F(x)
  - **pbinom(x,n,p)** calculates the cumulative probability density up to and including **x** successes (i.e., at most **x** successes) out of **n** trials given a probability of success of **p**
  - **pbinom(x-1,n,p,lower.tail=FALSE)** calculates the upper-tailed cumulative probability of obtaining **x** or more successes out of **n** trials given a probability of success of **p**. Also equivalent to **1-pbinom(x-1,n,p)**.
- **qbinom** (stats) – inverse binomial distribution calculations (useful for hypothesis testing)
- **rbinom** (stats) – generates random data the follow a specified binomial distribution

The Geometric Distribution

The geometric distribution describes the probability of obtaining the first success on the $\mathbf{x}^{\mathbf{th}}$ trial. Trials must meet the requirements of a Bernoulli trial (see above).

In R, we can use the following functions to assist with geometric distribution calculations:

- **dgeom** (stats) – calculates the probability density (mass) function; equivalent to f(x)
  - **dgeom(x-1,p)** calculates the probability of obtaining the first success on exactly the $\mathbf{x}^{\mathbf{th}}$ trial (here, **x** is equal to the number of failures plus 1) given a probability of success of **p**
  - An alternative way to think about this is to use **dgeom(y,p)**, which calculates the probability of obtaining the first success after **y** total failures given a probability of success of **p**

- **pgeom** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pgeom(x-1,p)** calculates the cumulative probability density of obtaining the first success on at most the **x**th trial (here, **x** is equal to the number of failures plus 1) given a probability of success of **p**
    - An alternative way to think about this is to use **pgeom(y,p)**, which calculates the probability of obtaining the first success after at most **y** total failures given a probability of success of **p**
    - **pgeom(x-1,p,lower.tail=FALSE)** calculates the upper-tailed cumulative probability of obtaining the first success after **x** or more trials (i.e., at least **x** failures before the first success) given a probability of success of **p**. Also equivalent to **1-pgeom(x-1,p)**. In terms of failures (**y = x-1**), we can write this as **pgeom(y,p,lower.tail=FALSE)** or **1-pgeom(y,p)**.
- **qgeom** (stats) – inverse geometric distribution calculations (useful for hypothesis testing)
- **rgeom** (stats) – generates random data the follow a specified geometric distribution

The Negative Binomial Distribution

The negative binomial distribution is an extension of the geometric distribution; however, instead of determining the probability of obtaining the first success in a certain number of trials (this is what the geometric distribution does), the negative binomial distribution is used to calculate the probability of obtaining the rth success in x number of trials. Like the binomial and geometric distributions, trials must meet the requirements of a Bernoulli trial (see above).

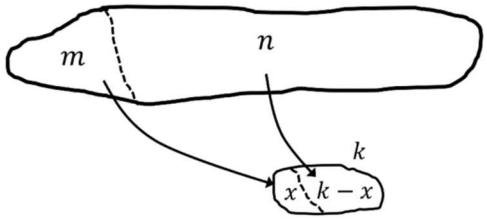
In R, we can use the following functions to assist with negative binomial distribution calculations:

- **dnbinom** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dnbinom(x,r,p)** calculates the probability that the **r**th success occurs on exactly the **x**th failure given a probability of success of **p**. Note that **x+r** is equal to the total number of trials.
- **pnbinom** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pnbinom(x,r,p)** calculates the cumulative probability density up to and including **x** failures (i.e., at most **x** failures) required to obtain the **r**th success given a probability of success of **p**. Note that **x+r** is equal to the total number of trials.
    - **pnbinom(x-1,r,p,lower.tail=FALSE)** calculates the upper-tailed cumulative probability of requiring **x** or more failures to obtain **r** successes given a probability of success of **p**. Also equivalent to **1-pnbinom(x-1,r,p)**.
- **qnbinom** (stats) – inverse negative binomial distribution calculations (useful for hypothesis testing)
- **rnbinom** (stats) – generates random data the follow a specified negative binomial distribution

The Hypergeometric Distribution

The hypergeometric distribution is used to describe probabilities associated with successes and failures in finite populations. This is also known as sampling without replacement, since removing items with or

without a certain characteristic affect the remaining proportion of the population that have or do not have that characteristic. Thus, trials involving the hypergeometric distribution do not follow the rules of Bernoulli trials.



The population contains **n+m** total items, **m** of which have a certain characteristic (I'll refer to this segment of the population as "defective", since this analysis is typically used with product quality assurance) and **n** of which are not defective. We take a small sample of size **k** from the finite population, and **x** is the number of defective items in that sample.

In R, we can use the following functions to assist with hypergeometric distribution calculations:

- **dhyper** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dhyper(x,m,n,k)** calculates the probability that, given the number of defective (**m**) and non-defective (**n**) items in the population, a sample of size **k** will contain exactly **x** defective items (and, consequently, **k-x** non-defective items).
- **phyper** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **phyper(x,m,n,k)** calculates the cumulative probability that, given the number of defective (**m**) and non-defective (**n**) items in the population, a sample of size **k** will contain at most **x** defective items (and, consequently, **k-x** non-defective items).
    - **phyper(x-1,m,n,k,lower.tail=FALSE)** calculates the upper-tailed cumulative probability that, given the number of defective (**m**) and non-defective (**n**) items in the population, a sample of size **k** will contain at least **x** defective items (and, consequently, at least **k-x** non-defective items). Also equivalent to **1-phyper(x-1,m,n,k)**.
- **qhyper** (stats) – inverse negative binomial distribution calculations (useful for hypothesis testing)
- **rhyper** (stats) – generates random data the follow a specified negative binomial distribution

The Poisson Distribution

The Poisson distribution is used to describe the probability of discrete events occurring over a continuous interval. The interval is typically time, length, area, or volume.

In R, we can use the following functions to assist with Poisson distribution calculations:

- **dpois** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dpois(x,mean)** calculates the probability that there will be x events occurring over an interval. IMPORTANTLY, **mean** in the **dpois** function is λT, where λ is the background

rate of occurrence (with units of events per interval size) and T is the interval of interest for a particular problem.

- **ppois** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **ppois(x,mean)** calculates the cumulative probability that there will be x events occurring over an interval. IMPORTANTLY, **mean** in the **dpois** function is λT, where λ is the background rate of occurrence (with units of events per interval size) and T is the interval of interest for a particular problem.
    - **ppois(x-1,mean,lower.tail=FALSE)** calculates the upper-tailed cumulative probability that there will be x events occurring over an interval. Also equivalent to **1-ppois(x-1,mean)**.
- **qpois** (stats) – inverse Poisson distribution calculations (useful for hypothesis testing)
- **rpois** (stats) – generates random data the follow a specified Poisson distribution

The Multinomial Distribution

The multinomial distribution is an extension of the binomial distribution and is useful for describing probabilities related to independent trials in which we have more than 2 possible outcomes. We can categorize the outcomes into *k* different classes, and each of the classes has a probability of occurrence.

In R, we can use the following two functions to assist with multinomial distribution calculations:

- **dmultinom** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dmultinom(x,size,pr)** calculates the probability that there will be **x** items chosen from a sample with size = **size** with corresponding probabilities **pr**. IMPORTANT: **x** and **pr** are vectors with corresponding elements, the sum of vector **x** must equal size, and the sum of vector **pr** must equal 1.
- **rmultinom** (stats) – generates random data the follow a specified multinomial distribution

The Uniform Distribution

The uniform distribution is a fairly simple continuous distribution that has constant density over its domain. In other words, the probability density is a straight line with zero slope. It is useful for calculating probabilities where the likelihood of a result is constant between two bounds. Many programming languages have a random number generator that generates a number between 0 and 1, and the probability of any number being generated between 0 and 1 is exactly the same. These number generators follow a uniform distribution.

In R, we can use the following functions to assist with uniform distribution calculations:

- **dunif** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dunif(x,min,max)** calculates the probability density at a value of **x** for the uniform distribution with minimum **min** and maximum **max**. Note that the area underneath the uniform distribution must be 1, and this function will output the same value regardless of **x** as long as **x** is between **min** and **max**. It would be rare to use the **dunif** function in typical statistics and probability problems, but nevertheless, it is presented here.

- **punif** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **punif(x,min,max)** calculates the cumulative probability density at a value of **x** for the uniform distribution with minimum **min** and maximum **max**.
    - **punif(x,min,max,lower.tail=FALSE)** calculates the upper-tailed cumulative probability, which is also equivalent to **1-punif(x,min,max)**.
- **qunif** (stats) – inverse uniform distribution calculations (useful for hypothesis testing)
- **runif** (stats) – generates random data the follow a specified uniform distribution

The Normal Distribution

Many real-world measurements can be described using the normal distribution (also known as the Gaussian distribution). A normally distributed variable can be described using a mean and standard deviation.

In R, we can use the following functions to assist with normal distribution calculations:

- **dnorm** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dnorm(x,mean,sd)** calculates the probability density at a value of **x** for the normal distribution with mean **mean** and standard deviation **sd**. It would be rare to use the **dnorm** function in typical statistics and probability problems, but nevertheless, it is presented here.
- **pnorm** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pnorm(x,mean,sd)** calculates the cumulative probability density at a value of **x** for the normal distribution with mean **mean** and standard deviation **sd**.
    - **pnorm(x,mean,sd,lower.tail=FALSE)** calculates the upper-tailed cumulative probability, which is also equivalent to **1-pnorm(x,mean,sd)**.
- **qnorm** (stats) – inverse normal distribution calculations (useful for hypothesis testing). See below for more details.
- **rnorm** (stats) – generates random data the follow a specified uniform distribution

Inverse Normal Distribution Calculations

The **pnorm** function (see above) is useful for calculating proportions (i.e., probabilities) underneath a specific normal distribution. It allows us to determine probabilities – for example, what percentage of the population lies between values **a** and **b**. Inverse normal distribution calculations allow us to start with a specified proportion or probability and determine what value(s) on the distribution correspond to that particular proportion or probability. For example, we may want to answer a question such as, "25% of the population lies below what value?" Or, "50% of the population lies between 3 and what value?"

We can use the **qnorm** function in R to perform these inverse normal distribution calculations.

- **qnorm** (stats) – inverse normal distribution calculations
    - **qnorm(p,mean,sd)** calculates the value of the normal distribution with mean **mean** and standard deviation **sd** that has **p** proportion or probability to the left of it.

- **qnorm(p,mean,sd,lower.tail=FALSE)** calculates the value of the normal distribution with mean **mean** and standard deviation **sd** that has **p** proportion or probability to the right of it.

Standardizing and Z-Values

Oftentimes, we wish to convert a normal random variable (X) to a standard normal variable (Z). This is useful for comparing distributions with different means and standard deviations and basically puts the two distributions on the same playing field. This process of converting a normal random variable to a standard normal variable with mean 0 and standard deviation equal to 1 is known as standardization. A z-value is essentially where on the standard normal distribution a specified random variable lies when standardized.

You can always use the simple standardization formula, if desired: $Z = \frac{X-\mu}{\sigma}$, where μ is the mean of the distribution and σ is the population standard deviation.

However, an alternative in R is as follows:

1. First, we can calculate the proportion (probability) to the left of X using the **pnorm** function: **p <- pnorm(X,mean,sd)**
2. Second, we can calculate the value on the standard normal distribution (this is known as the z-value) based on the **qnorm** function and **p**, calculated above: **z <- qnorm(p,0,1)**

We can also use the **qnorm** function to calculate "percentage points" (z_α) of the standard normal distribution:

- **qnorm(p)** calculates the z-value with **p** proportion of the distribution to the *left* of it based on the standard normal distribution. For example, to calculate $z_{0.05}$ using this version of the **qnorm** function, we can use **qnorm(0.95)**.
- **qt(p,lower.tail=FALSE)** calculates the z-value with **p** proportion of the distribution to the *right* of it based on the standard normal distribution. For example, to calculate $z_{0.05}$ using this version of the **qnorm** function, we can use **qnorm(0.05,lower.tail=FALSE)**.
- The normal distribution is symmetric, which means that z_α is equivalent to $-z_{1-\alpha}$.

The T Distribution

At this point in the course, we begin to address samples and sampling distributions (as opposed to population distributions). For the "variance unknown" case (when the true population variance is unknown *and* the sample size is small), we can standardize our sample statistics and use the T distribution to calculate probabilities. Analysis is very similar to the standard normal distribution, but the T distribution that we use is based on *degrees of freedom*. The T distribution is similar to the normal distribution but heavier (fatter) in the tails, which means there is more uncertainty when sample size is small. For simple probability problems, the degrees of freedom are typically **n-1**, where **n** is the sample size.

If we wish to solve problems related to the T distribution in R, we first need to standardize the sample average of a normally distributed variable to a "t-value", we can use the simple standardization formula: $T = \frac{\bar{x}-\mu}{s/\sqrt{n}}$, where $\bar{x}$ is the sample average, μ is the mean of the distribution, s is the sample standard deviation, and n is the sample size.

Next, once we have calculated a t-value (**t**), we can use the following functions to perform calculations related to the T distribution:

- **dt** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **dt(t,df)** calculates the probability density at a value of **t** for the T distribution based on **df** degrees of freedom. It would be rare to use the **dt** function in typical statistics and probability problems, but nevertheless, it is presented here.
- **pt** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pt(t,df)** calculates the cumulative probability density at a value of **t** for the T distribution based on **df** degrees of freedom.
    - **pt(t,df,lower.tail=FALSE)** calculates the upper-tailed cumulative probability at a value of **t** for the T distribution based on **df** degrees of freedom, which is also equivalent to **1-pt(t,df)**.
- **qt** (stats) – inverse T distribution calculations (useful for hypothesis testing).
    - **qt(p,df)** calculates the t-value with **p** proportion of the distribution to the ***left*** of it based on the T distribution with df degrees of freedom. For example, to calculate $t_{0.05,12}$ using this version of the **qt** function, we can use **qt(0.95,12)**.
    - **qt(p,df,lower.tail=FALSE)** calculates the t-value with **p** proportion of the distribution to the ***right*** of it based on the T distribution with df degrees of freedom. For example, to calculate $t_{0.05,12}$ using this version of the **qt** function, we can use **qt(0.05,12,lower.tail=FALSE)**.
    - The T distribution is symmetric, which means that $t_{\alpha,n-1}$ is equivalent to $-t_{1-\alpha,n-1}$.
- **rt** (stats) – generates random data the follow a specified T distribution

The Chi-Squared Distribution

The chi-squared distribution allows us to put together confidence intervals on and to perform hypothesis tests on sample variance.

In order to use the chi-squared distribution, we first need to "standardize" our sample variance (s^2) to a chi-squared variable:

$$\chi^2 = \frac{(n-1)s^2}{\sigma^2}$$

Next, we can calculate probabilities related to the chi-squared distribution in R using the following functions:

- **dchisq** (stats) – calculates the probability density (mass) function; equivalent to f(x)

    - **dchisq(x,df)** calculates the probability density at a value of **x** for the chi-squared distribution based on **df** degrees of freedom. It would be rare to use the **dchisq** function in typical statistics and probability problems, but nevertheless, it is presented here.
- **pchisq** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pchisq(x,df)** calculates the cumulative probability density at a value of **x** for the chi-squared distribution based on **df** degrees of freedom.
    - **pchisq(x,df,lower.tail=FALSE)** calculates the upper-tailed cumulative probability at a value of **x** for the chi-squared distribution based on **df** degrees of freedom, which is also equivalent to **1-pchisq(x,df)**.
- **qchisq** (stats) – inverse chi-squared distribution calculations (useful for hypothesis testing).
    - **qchisq(p,df)** calculates the chi-squared value with **p** proportion of the distribution to the *left* of it based on the chi-squared distribution with **df** degrees of freedom. For example, to calculate $\chi^2_{0.10,15}$ using this version of the **qchisq** function, we can use **qchisq(0.90,15)**, which is equivalent to **qchisq(0.10,15,lower.tail=FALSE)**.
    - **qchisq(p,df,lower.tail=FALSE)** calculates the chi-squared value with **p** proportion of the distribution to the *right* of it based on the chi-squared distribution with **df** degrees of freedom. For example, to calculate $\chi^2_{0.90,15}$ using this version of the **qchisq** function, we can use **qchisq(0.90,15,lower.tail=FALSE)**, which is equivalent to **qchisq(0.10,15)**.
    - The chi-squared distribution is asymmetric, which means that $\chi^2_{\alpha,n-1}$ is not equivalent to $-\chi^2_{1-\alpha,n-1}$.
- **rt** (stats) – generates random data the follow a specified chi-squared distribution

The F Distribution

The F distribution can be used to calculate probabilities related to the ratio of two chi-squared variables. This is useful when we perform hypothesis tests related to the comparison of variances of two populations. It is also useful in ANOVA (analysis of variance). In order to use the F distribution, we need numerator degrees of freedom (**df1**) and denominator degrees of freedom (**df2**).

In R, we can use the following functions to assist with F distribution calculations:

- **df** (stats) – calculates the probability density (mass) function; equivalent to f(x)
    - **df(x,df1,df2)** calculates the probability density at a value of **x** for the F distribution based on **df1** numerator degrees of freedom and **df2** denominator degrees of freedom. It would be rare to use the **df** function in typical statistics and probability problems, but nevertheless, it is presented here.
- **pf** (stats) – calculates the cumulative probability density function; equivalent to F(x)
    - **pf(x,df1,df2)** calculates the cumulative probability density at a value of **x** for the F distribution based on **df1** numerator degrees of freedom and **df2** denominator degrees of freedom.
    - **pf(x,df1,df2,lower.tail=FALSE)** calculates the upper-tailed cumulative probability at a value of **x** for the chi-squared distribution based on **df1** numerator degrees of freedom and **df2** denominator degrees of freedom. This is also equivalent to **1-pf(x,df1,df2)**.
- **qf** (stats) – inverse F distribution calculations (useful for hypothesis testing).

- **qf(p,df1,df2)** calculates the F-value with **p** proportion of the distribution to the *left* of it based on the F distribution with **df1** numerator degrees of freedom and **df2** denominator degrees of freedom. For example, to calculate $f_{0.05,12,15}$ using this version of the **qf** function, we can use **qf(0.95,12,15)**, which is equivalent to **qf(0.05,12,15,lower.tail=FALSE)**.
- **qf(p,df1,df2,lower.tail=FALSE)** calculates the F-value with **p** proportion of the distribution to the *right* of it based on the F distribution with **df1** numerator degrees of freedom and **df2** denominator degrees of freedom. For example, to calculate $f_{0.95,12,15}$ using this version of the **qf** function, we can use **qf(0.95,12,15,lower.tail=FALSE)**, which is equivalent to **qchisq(0.10,15)**.
- The F distribution is asymmetric, which means that $f_{\alpha,u,v}$ is not equivalent to $-f_{1-\alpha,u,v}$.

- **rf** (stats) – generates random data the follow a specified F distribution