Data Analysis and Statistical Inference by Dr. Mine Çetinkaya-Rundel

Notes taken during the course

2015-04-29

Unit 1

Variables

- Variables, numerical or categorical (quantitative or qualitative):
  - Define a **numerical** variable is an observed response that is a numerical value
  - If the variable is numerical, further classify it as **continuous** or **discrete** based on whether or not the variable can take on an infinite number of values or only non-negative whole numbers, respectively.
  - Define a categorical variable is a variable that can take on one of a limited, and usually fixed, number of possible values, thus assigning each individual to a particular group or “category.”
  - If the variable is categorical, determine if it is **ordinal** based on whether or not the levels have a natural ordering.
- Variables, associated or independent:
  - Define **associated** variables as variables that show some relationship with one another. Further categorize this relationship as positive or negative association, when possible.
  - Define variables that are not associated as independent.
- Variables, explanatory or response:
  - Identify the **explanatory** variable in a pair of variables as the variable suspected of affecting the other. The affected variable is called a **response** variable. However, note that labeling variables as explanatory and response does not guarantee that the relationship between the two is actually causal, even if there is an association identified between the two variables.

Studies

- Studies, observational or experimental
  - **Observational**: collect data in a way that doesn’t directly interfere with how the data arise; such studies only *observe* the data, only establish an association. Can be:
    - *retrospective*: uses past data
    - *prospective*: collects the data throughout the study
  - **Experimental**: randomly assign subject to treatments and establish casual connections; these studies lead *experiments*. These are the four principles of experimental design:
    - control any possible confounders
    - randomize into treatment and control groups
    - replicate by using a sufficiently large sample or repeating the experiment
    - block any variables that might influence the response:
      - E.g., design an investigation whether energy gels help you run faster (treatment: energy gel, control: no energy gel)
        - however, energy gels might affect pro and amateur athletes differently. and thus, we need to block for pro status:
          - divide the sample to pro and amateur
          - randomly assign pro and amateur athletes to treatment and control groups
          - pro and amateur athletes are equally represented in both groups
- Experimental studies make use of:
  - **Random sampling**
    - Which can be of these properties:
      - **Simple random sampling**: Each subject in the population is equally likely to be selected.
      - **Stratified sampling**: First divide the population into *homogenous* strata (subjects within each stratum are similar, across strata are different), then randomly sample from within each strata (E.g., divide men and women into two separate stratas and then randomly sampling from these stratas).
      - **Cluster sampling**: First divide the population into clusters (subjects within each cluster are *non-homogenous*, but clusters are similar to each other), then randomly sample a few clusters, and then randomly sample from within each cluster.
    - Or, can face the following bias:
      - **Convenience sample**: If individuals, who are easily accessible are more likely to be included in

the sample

            - **Non-response**: If only a [non-random] fraction of the randomly sampled people respond to a survey such that sample is no longer representative of the popuplation
            - **Volunatry response**: If the sample consists of people, who volunteer to respond because they have strong oppinions on the issue.
    - **Random assignment** (or random placement)
        - Which is an experimental technique for assigning subjects to different groups in an experiment (e.g., a treatment group versus a control group) using a chance procedure.
    - **Blinding**
        - Which can be
            - **Single**: If a subject is blinded
            - **Double**: If both tester and subject are blinded
- If there sources of bias in a given study, these variables are called **confounding**.

Results

- Results, can be generalized to the population or not, and whether the results suggest correlation or causation between the quantities studied:
    - If random sampling has been employed in data collection, the results should be generalizable to the target population.
    - If random assignment has been employed in study design, the results suggest causality.

Analysis

- Numerical variables
    - Pay attention to its shape, center, and spread, as well as any unusual observations
    - Note that there are three commonly used measures of center and spread:
        - Center: **mean** (the arithmetic average), **median** (the midpoint), **mode** (the most frequent observation)
            - If median is larger than mean, then left skewness is possible
            - If mean is larger than median, then right skewness is possible
            - In other words, the mean is right of the median under right skew, and left of the median under left skew
        - Spread: **standard deviation** (variability around the mean), **range** (max-min), **interquartile range** (middle 50% of the distribution)
    - Identify the shape of a distribution as *symmetric*, *right skewed*, or *left skewed*, and *unimodal*, *bimodoal*, *multimodal*, or *uniform*.
    - Use *histograms* and *box plots* to visualize the shape, center, and spread of numerical distributions, and intensity maps for visualizing the spatial distribution of the data
    - Define a **robust statistic** (e.g. median, IQR) as a statistic that is not heavily affected by skewness and extreme outliers, and determine when such statistics are more appropriate measures of center and spread compared to other similar statistics
    - Recognize when transformations (e.g. log) can make the distribution of data more symmetric, and hence easier to model
    - Use *scatterplots* for describing the relationship between two numerical variables, making sure to note the direction (positive or negative), form (linear or non-linear), and the strength of the relationship as well as any unusual observations that stand out (generally, we place the explanatory variable on the x-axis, and the response variable on the y-axis).
    - CORRELATION DOESN'T IMPLY CAUSATION!
- Categorical variables
    - Use *frequency tables* and bar plots to describe the distribution of one categorical variable
    - Use *contingency tables* and segmented bar plots or mosaic plots to assess the relationship between two categorical variables
    - Use *side-by-side box plots* for assessing the relationship between a numerical and a categorical variable
- Note that an observed difference in sample statistics suggesting dependence between variables may be due to random chance, and that we need to use hypothesis testing to determine if this observed difference

is too large to be attributed to random chance. Therefore, it’s essential to set up *null* and *alternative* hypotheses for testing for independence between variables, and evaluate the data’s support for these hypotheses using a simulation technique:

- Set a null and an alternative hypothesis
- Simulate the experiment assuming that the null hypothesis is true
- Thus, build a distribution of outcomes
- Evaluated the probability (**p-value**) of observing an outcome at least as extreme as the one observed in the original data
- And if this probability is sufficienty low, reject the null hypothesis in favor of the alternative

Unit 2

Outcomes and events

- Define the **experiment** as the situation involving probability that leads to results called **outcomes**.
- Define the **outcome** as the result of a single trial of an experiment
- Define the **event** as one or more outcomes of an experiment
- Define the **probability** of an outcome as the proportion of times the outcome would occur if we observed the random process that gives rise to it an infinite number of times

- Define **disjoint** (or **mutually exclusive**) **events** as events that cannot both happen at the same time: if A and B are disjoint, then `P(A and B) = 0`
- Distinguish between **disjoint** and **independent events**:
  - If A and B are *independent*, then having information on A does not tell us anything about B (and vice versa)
  - If A and B are *disjoint*, then knowing that A occurs tells us that B cannot occur (and vice versa)
  - *Disjoint* (mutually exclusive) events are always *dependent* since if one event occurs we know the other one cannot
- Define **complementary** outcomes as mutually exclusive outcomes of the same random process whose probabilities add up to 1: if A and B are complementary, then `P(A) + P(B) = 1`

- Distinguish between **union of events (A or B)** and **intersection of events (A and B)**
- Calculate the probability of *union of events* using the (general) addition rule:
  - If A and B *are not mutually exclusive*, then `P(A or B) = P(A) + P(B) − P(A and B)`
  - If A and B *are mutually exclusive*, `P (A or B) = P (A) + P (B)` (since for mutually exclusive events `P(A and B) = 0`)
- Calculate the probability of *intersection of events* using the multiplication rule:
  - If A and B are *independent*, then `P(A and B) = P(A) × P(B)`
  - If A and B are *dependent*, then `P(A and B) = P(A|B) × P(B)`

Probabilities of outcomes and events

- Distinguish between **marginal**, **joint** and **conditional** probabilities
  - *Marginal* probability: the probability of an event occurring (p(A)), it may be thought of as an unconditional probability. It is not conditioned on another event. Example: the probability that a card drawn is red (p(red) = 0.5). Another example: the probability that a card drawn is a 4 (p(four)=1/13)
  - *Joint* probability: p(A and B). The probability of event A and event B occurring. It is the probability of the intersection of two or more events. The probability of the intersection of A and B may be written p(A ∩ B). Example: the probability that a card is a four and red =p(four and red) = 2/52=1/26. (There are two red fours in a deck of 52, the 4 of hearts and the 4 of diamonds)
  - *Conditional* probability: p(A|B) is the probability of event A occurring, given that event B occurs. Example: given that you drew a red card, what's the probability that it's a four (p(four|red))=2/26=1/13. So out of the 26 red cards (given a red card), there are two fours so 2/26=1/13
- Understand Bayes' Theorem: `P(A|B) = P(A and B) / P(B)` (thus, `P(A and B) = P(A|B) * P(B)` (see above))
  - Or: *conditional* = *joint_of_both* / *marginal_of_this_condition*
  - Breast cancer test example

Normal distribution

- Define the **standardized (Z) score** of a data point as the number of SDs it is away from the mean: `Z = (X − μ) / σ`
  - Obviously, Z-score of the *mean* is always 0: `Z(μ) = 0`
- Use the **Z score**
  - If the distribution is normal, to determine the *percentile* score of a data point:

        - Use `pnorm(X, μ, σ)` to detect the percentile (i.e., the probability that any event from this outcome lies below event X)
        - Use `qnorm(P, μ, σ)` to detect the value of event X, which is the upper border for all the events from this outcome, which occur with the probability P
        - Note the `lower.tail` option
    - Regardless of the shape of the distribution, to assess whether or not the particular observation is considered to be unusual (more than 2 standard deviations away from the mean)
    - Depending on the shape of the distribution determine whether the *median* would have a negative (right skewed), positive (left skewed), or 0 (symmetrical) Z score keeping in mind that the mean always has a Z score of 0

Binomial distribution

- Determine if a random variable is **binomial** using the four conditions.
    - The trials are independent.
    - The *number of trials*, `n`, is fixed.
    - Each trial outcome can be classified as a success or failure.
    - The *probability of a success*, `p`, is the same for each trial.
- Build a binomial distribution: `rbinom(n, SIZE, p)` (compare to `rnorm(SIZE, μ, σ)`)
- Calculate the number of possible scenarios for obtaining `k` successes in `n`: `choose(n, k)`
- Calculate the probability of a given number of successes in a given number of trials: `pbinom(k-1, n, p, lower.tail = F)` or `sum(dbinom(k:n, n, p))`
- Calculate the expected number of successes in a given number of binomial trials `μ = n * p` and its standard deviation `σ = sqrt(n * p * (1 − p))` (see `bitono` function)
- With a sufficiently large number of trials (n * p ≥ 10 and n * (1 − p) ≥ 10), use the normal approximation to calculate binomial probabilities (converting the binomial distribution in question to the normal distribution via `bitono` function)

Unit 3

Sample statistics

- Define **sample statistic** as a *point estimate* for a population parameter, for example, the sample mean is used to estimate the population mean, and note that point estimate and sample statistic are synonymous
- Recognize that *point estimates* (such as the sample mean) will vary from one sample to another, and define this variability as **sampling variability** (sometimes also called sampling variation)
- Calculate the sampling variability of the mean, the **standard error**, as `SE = σ / sqrt(n)` where `σ` is the population standard deviation
    - Note that when the population standard deviation `σ` is not known (almost always), the standard error SE can be estimated using the sample standard deviation `s`, so that `SE = s / sqrt(n)`
- Distinguish *standard deviation* (`σ` or `s`) and *standard error* (`SE`): standard deviation measures the variability in the data, while standard error measures the variability in point estimates (aka sample statistics) from different samples of the same size and from the same population, i.e. measures the sampling variability

Confidence levels and Confidence intervals

- Define a **confidence interval** as the *plausible range of values* for a population parameter
    - The **confidence interval** is the *plus-or-minus figure* usually reported in newspaper or television opinion poll results. For example, if you use a confidence interval of 4 and 47% percent of your sample picks an answer you can be "sure" that if you had asked the question of the entire relevant population between 43% (47-4) and 51% (47+4) would have picked that answer.
- Define the **confidence level** as the *percentage of random samples* which yield confidence intervals that capture the true population parameter
    - The **confidence level** tells you *how sure you can be*. It is expressed as a percentage and represents how often the true percentage of the population who would pick an answer lies within the confidence interval. The 95% confidence level means you can be 95% certain; the 99% confidence level means you can be 99% certain. Most researchers use the 95% confidence level.
- Recognize that the Central Limit Theorem (CLT) is about the distribution of *point estimates*, and that given certain conditions, this distribution will be nearly normal
    - In the case of the mean the CLT tells us that
        - if the sample size is sufficiently large ($n \geq 30$ or larger if the data are considerably skewed - but less than 10%), or the population is known to have a normal distribution (when the population distribution is unknown, the condition skewness can be checked using a histogram or some other visualization of the distribution of the observed data in the sample)
        - and the observations in the sample are independent (either *randomly sampled* (in the case of observational studies) or *randomly assigned* (in the case of experiments))
        - then the distribution of the sample mean will be nearly normal, centered at the true population mean and with a spread of standard error: `x_bar ~ N (mean = μ, SE =  σ / sqrt(n)`
- Recognize that the nearly normal distribution of the point estimate (as suggested by the CLT) implies that a **confidence interval** can be calculated as `point_estimate ± z* * SE` (note that `z*` is always positive ; see `SI.confidence_intreval` function)
    - For means this is: `x_bar ± z* * σ / sqrt(n)`
- Define **margin of error** as the *distance* required to travel in either direction away from the point estimate when constructing a confidence interval, i.e. `z* * σ / sqrt(n)`

- Finally, interpret a **confidence interval** as "We are XX% confident that the true population parameter is in this interval", where XX% is the desired **confidence level**

Hypothesis testing

- Recognize that in *hypothesis testing* we evaluate two competing claims:
    - the *null hypothesis*, which represents a skeptical perspective or the status quo, and

    - the *alternative hypothesis*, which represents an alternative under consideration and is often represented by a range of possible parameter values
- Construction of hypotheses:
    - Always construct hypotheses about population parameters (e.g. population mean, `μ`) and not the sample statistics (e.g. sample mean, `x_bar`). Note that the population parameter is unknown while the sample statistic is measured using the observed data and hence there is no point in hypothesizing about it.
    - Define the null value as the value the parameter is set to equal in the null hypothesis.
    - Note that the alternative hypothesis might be one-sided (μ < or > the null value) or two-sided (μ ≠ the null value), and the choice depends on the research question
- Define a **p-value** as the conditional probability of obtaining a sample statistic at least as extreme as the one observed given that the null hypothesis is true:
  `p-value = P(observed or more extreme sample statistic | H0 true)`
- Calculate a **p-value** as the area under the normal curve beyond the observed sample mean (either in one tail or both, depending on the alternative hypothesis). Note that in doing so you can use a *Z-score*, where `Z = (point estimate - null value) / SE` or `Z = (x_bar - μ) / SE` (see `SI.z.score` and `SI.pvalue` functions)
    - E.g., median's Z-score is positive for the left skewed distribution and negative for the right skewed distribution
- Infer that if a **confidence interval** does not contain the null value the null hypothesis should be rejected in favor of the alternative
- Compare the **p-value** to the significance level to make a decision between the hypotheses:
    - If the p-value *is less than the significance level*, reject the null hypothesis since this means that obtaining a sample statistic at least as extreme as the observed data is extremely unlikely to happen just by chance, and conclude that the data provides evidence for the alternative hypothesis
    - If the p-value *is more the significance level*, fail to reject the null hypothesis since this means that obtaining a sample statistic at least as extreme as the observed data is quite likely to happen by chance, and conclude that the data does not provide evidence for the alternative hypothesis.
    - Note that we can never "accept" the null hypothesis since the hypothesis testing framework does not allow us to confirm it
    - Note that p-value is not the probability of the alternative hypothesis being true
- Note that the conclusion of a hypothesis test might be erroneous regardless of the decision we make
    - Define a **Type 1 error** as rejecting the null hypothesis when the null hypothesis is actually true
    - Define a **Type 2 error** as failing to reject the null hypothesis when the alternative hypothesis is actually true
        - Define **power** as the probability of correctly rejecting the null hypothesis (complement of Type 2 error)
- Note that the probability of making a *Type 1 error* is equivalent to the *significance level*, and therefore choose a significance level depending on the risks associated with Type 1 and Type 2 errors:
    - Use a smaller α if Type 1 error is relatively riskier
    - Use a larger α if Type 2 error is relatively riskier

> - Formulate the framework for statistical inference using hypothesis testing and nearly normal point estimates:
>   - Set up the hypotheses first in plain language and then using appropriate notation
>   - Identify the appropriate sample statistic that can be used as a point estimate for the parameter of interest
>   - Verify that the conditions for the CLT hold (if the conditions necessary for the CLT to hold are not met, note this and do not go forward with the analysis)
>   - Compute the SE, sketch the sampling distribution, and shade area(`s`) representing the `p-value`
>   - Using the sketch and the normal model, calculate the p-value and determine if the null hypothesis should be rejected or not, and state your conclusion in context of the data and the research question

Unit 4

Bootstrap

- The basic idea of **bootstrapping** is that inference about a population from sample data (sample → population) can be modeled by resampling the sample data and performing inference on (resample → sample). As the population is unknown, the true error in a sample statistic against its population value is unknowable. In *bootstrap-resamples*, the 'population' is in fact the sample, and this is known; hence the quality of inference from resample data → 'true' sample is measurable.
- More formally, the bootstrap works by treating inference of the true probability distribution A, given the original data, as being analogous to inference of the empirical distribution of Ã, given the resampled data. The accuracy of inferences regarding Ã using the resampled data can be assessed because we know Ã. If Ã is a reasonable approximation to A, then the quality of inference on A can in turn be inferred.
- To construct a *bootstrap distribution* see function `SI.boot`
- Construct bootstrap confidence intervals using one of the following methods (see function `SI.boot.confidence_interval`):
  - *Percentile method*: XX% confidence level is the middle XX% of the bootstrap distribution.
  - *Standard error method*: If the standard error of the bootstrap distribution is known, and the distribution is nearly normal, the bootstrap interval can also be calculated as `x_boot ± z * SE_boot`
  - Recognize that when the bootstrap distribution is extremely skewed and sparse, the bootstrap confidence interval may not be reliable

Paired data

- Define observations as **paired** if each observation in one dataset has a special correspondence or connection with exactly one observation in the other data set
- Carry out inference for paired data by first subtracting the paired observations from each other, and then treating the set of differences as a new numerical variable on which to do inference (such as a confidence interval or hypothesis test for the average difference).

Difference of two means

- Calculate the *standard error* of the *difference between means of two independent samples* as `SE = sqrt( (sd_1^2 / num_1) + (sd_2^2 / num_2) )` and use this standard error in hypothesis testing and confidence intervals comparing means of independent groups
- Recognize that a good interpretation of a confidence interval for the difference between two parameters includes a *comparative statement* (mentioning which group has the larger parameter; see function `by`)
- Recognize that a confidence interval for the difference between two parameters that doesn't include 0 is in agreement with a hypothesis test where the null hypothesis that sets the two parameters equal to each other is rejected

Student's T distribution

- Means of small samples (n is less than 30) follow the t distribution (instead of the normal, z, distribution)
- Note that the t-distribution has a single parameter, degrees of freedom, and as the degrees of freedom increases this distribution approaches the normal distribution (see functions `SI.t.score`, `SI.t.confidence_interval` and `SI.t.pvalue`)
- Use a t-statistic, with *degrees of freedom* `df = n-1` for inference for a population mean using data from a small sample: `CI: x_bar ± t_df * SE`, `HT: T_df = (x_bar − μ) / SE`, where `SE = sd / sqrt(n)` (see function `SI.t.confidence_interval` and `SI.t.pvalue`)
- Use a t-statistic, with *degrees of freedom* `df = min(n_1 − 1, n_2 − 1)` for inference for difference between means of two population means using data from two small samples, where `SE = sqrt( (sd_1^2 / n_1) + (sd_2^2 / n_2) )`
- Make note of the *pooled standard deviation* but use it in rare circumstances where the standard deviations

of the populations being compared are known to be very similar: `s_pooled = sqrt ( (s_1^2 * (n_1 - 1) + s_2^2 * (n_2 -1)) / (n_1 + n_2 -2) )`
- How to obtain a p-value for a t-test: `pt(T, df = n - 1)` (e.g. `pt(1.75, 19, lower.tail = F)`)
- How to calculate a critical t-score (t_df) for a confidence interval: `qt((1 - (1 - CI)/2), df = n - 1)`

ANOVA[1]

- Define **analysis of variance (ANOVA)** as a statistical inference method that is used to determine - by simultaneously considering many groups at once - if the variability in the sample means is so large that it seems unlikely to be from chance alone
- Recognize that the null hypothesis in ANOVA sets all means equal to each other, and the alternative hypothesis suggest that at least one mean is different: `H0: μ1=μ2=...=μk` and `HA: At least one mean is different`
- List the conditions necessary for performing ANOVA:
  - the *observations* should be *independent within and across groups*
  - the *data within each group* are *nearly normal*
  - the *variability across the groups* is *about equal*
  - use graphical diagnostics to check if these conditions are met (boxplots)
- Use `SI.anova` function
- Note that conducting many t-tests for differences between each pair of means leads to an increased Type 1 Error rate, and we use a corrected significance level (Bonferroni correction, `α⋆ = α/K`, where `K` is the number of comparisons being considered, `K = k * (k - 1) / 2`, where `k` is the number of groups; see `combn` function) to combat inflating this error rate
- Note that it is possible to reject the null hypothesis in ANOVA but not find significant differences between groups when doing pairwise comparisons (see `SI.anova.pairwise` function)

Unit 5

Proportions

- Define **population proportion** `p` (parameter) and **sample proportion** `p_bar` (point estimate)
- Calculate the **sampling variability of the proportion**, the standard error, as `SE = sqrt(p * (1-p) / n)`, where `p` is the population proportion
  - Note that when the *population proportion* `p` is not known (almost always), this can be estimated using the *sample proportion*, `p_bar`, `SE = sqrt(p_bar * (1 - p_bar) / n)`.
  - See function `SI.prop.standart_error`
- Recognize that the Central Limit Theorem (CLT) is about the distribution of point estimates, and that given certain conditions, this distribution will be nearly normal. In the case of the proportion the CLT tells us that if:
  - the observations in the sample are independent
  - the sample size is sufficiently large (checked using the success/failure condition: `n * p ≥ 10` and `n * (1 - p) ≥ 10`)
- Then the distribution of the sample proportion will be nearly normal, centered at the true population proportion and with a standard error described above: `p_bar ~ N(mean = p, SE =  sqrt(p * (1-p) / n)`
- Besides, note that if the CLT doesn't apply and the *sample proportion is low (close to 0) the sampling distribution will likely be right skewed*, if the *sample proportion is high (close to 1) the sampling distribution will likely be left skewed*

- Remember that confidence intervals are calculated as: `point estimate ± margin of error` (feel free to use usual function `SI.confidence_intreval`)
- Remember that est tstatistics are calculated as: `test statistic = (point estimate - null value) / standard error` (again, feel free to use usual function `SI.pvalue`, see examples)

- However, also not that the *standard error* calculation for the *confidence interval* and the *hypothesis test* are different when dealing with proportions, since in the hypothesis test we need to assume that the null hypothesis is true (remember: `p-value = P(observed or more extreme test statistic | H0 true)`)
  - For *confidence intervals* use `p_bar` (aka *observed sample proportion*) when calculating the standard error and checking the success/failure condition: `SE = sqrt(p_bar * (1 - p_bar) / n)`
  - For *hypothesis tests* use `p0` (aka *null value*) when calculating the standard error and checking the success/failure condition: `SE = sqrt(p0 * (1 - p0) / n)`
  - Note that such a discrepancy doesn't exist when conducting inference for means, since the mean doesn't factor into the calculation of the standard error, while the proportion does

Estimating the difference between two proportions

- Note that the calculation of the standard error of the distribution of the difference in two independent sample proportions is different for a confidence interval and a hypothesis test (still covered though by `SI.prop.standart_error` function):
  - confidence interval and hypothesis test when `H0: p_1 - p_2 = some value other than 0`: `SE(p_bar_1 - p_bar_2) = sqrt((p_bar_1* (1 - p_bar_1) / n_1) + (p_bar_2 * (1 - p_bar_2) / n_2))`
  - hypothesis test when `H0: p1 - p2 = 0`: `SE(p_bar_1 - p_bar_2) = sqrt((p_pool * (1 - p_pool) / n_1) + (p_pool * (1 - p_pool) / n_2))` where `p_pool` is the overall rate of success: `(number of successes in group 1 + number of successes in group 2) / n_1 + n_2` (note that in order to calculate pooled SE with `SI.prop.standart_error` we provide it with real numbers, not proportions, and set switch `pool` to `TRUE`; see examples)
- Obviously, the reason for the difference in calculations of standard error is the same as in the case of the single proportion: when the null hypothesis claims that the two population proportions are equal, we need to take that into consideration when calculating the standard error for the hypothesis test, and use a

common proportion for both samples

Chi-square goodness of fit (GOF) test

- Use a **chi-square test of goodness of fit** to evaluate if the distribution of levels of a single categorical variable follows a hypothesized distribution:
  - H0: The distribution of observed counts follows the hypothesized distribution, and any observed differences are due to chance
  - HA: The distribution of observed counts does not follow the hypothesized distribution
- Calculate the *expected counts for a given level (cell)* in a one-way table as the sample size times the hypothesized proportion for that level: `expected = n * ratio`
- Calculate the chi-square test statistic as `χ2 = sum( (observed count - expected count)^2 / expected count)` (use `SI.chisq` function)
- Note that the chi-square statistic is always positive, and follows a right skewed distribution with one parameter: degrees of freedom
- Note that the *degrees of freedom* for the *chi-square statistic for the goodness of fit test* is `df = k - 1`, where `k` is the number of cells
- List the conditions necessary for performing a chi-square test (both *goodness of fit* or *independence* (see below))
  - the observations should be independent
  - expected counts for each cell should be at least 5
  - degrees of freedom should be at least 2 (if not, use methods for evaluating proportions)
- Use `SI.chisq.pvalue` function to obtain p-value, given the chi-square statistic and the number of levels

Independence test

- When evaluating the independence of two categorical variables where at least one has more than two levels, use a **chi-square test of independence**:
  - H0: The two variables are independent
  - HA : The two variables are dependent
- Calculate expected counts in two-way tables as `E = (row total * column total) / grand total`
- Calculate the *degrees of freedom* for the *chi-square test of independence* as `df = (R - 1) * (C - 1)`, where R is the number of rows in a two-way table, and C is the number of columns
- Note that there is no such thing as a chi-square confidence interval for proportions, since in the case of a categorical variables with many levels, there isn't one parameter to estimate
- Use `SI.chisq.independence_test` function to obtain p-value for the independence test, given the chi-square statistic and the number of levels

Unit 6

Correlation and residuals

- Define the **explanatory variable** as the independent variable (predictor), and the **response variable** as the dependent variable (predicted) (explanatory variable is always graphed on x-axis (thus, eXplanatory))
- Plot the explanatory variable (x) on the x-axis and the response variable (y) on the y-axis, and fit a *linear regression model*: `y = β_0 + β_1 * x`, where where `β_0` is the **intercept**, and `β_1` is the **slope** (note that the point estimates (estimated from observed data) for `β_0` and `β_1` are `b_0` and `b_1`, respectively
- When describing the association between two numerical variables, evaluate
  - *direction*: positive (x↑,y↑), negative (x↓,y↑)
  - *form*: linear or not
  - *strength*: determined by the scatter around the underlying relationship
- Define **correlation** as the linear association between two numerical variables (note that a relationship that is nonlinear is simply called an association)
- Note that correlation coefficient (R, also called Pearson's R) has the following properties:
  - the *magnitude (absolute value)* of the correlation coefficient measures the strength of the linear association between two numerical variables
  - the *sign* of the correlation coefficient indicates the direction of association
  - the *correlation coefficient* is always between -1 and 1, -1 indicating perfect negative linear association, +1 indicating perfect positive linear association, and 0 indicating no linear relationship
    - the correlation coefficient is *unitless*
    - since the correlation coefficient is unitless, it is not affected by changes in the center or scale of either variable (such as unit conversions)
    - the correlation of X with Y is the same as of Y with X
    - the correlation coefficient is sensitive to outliers
- Recall that correlation does not imply causation!
- Define **residual** `e` as the difference between the *observed* `y` and *predicted* `y_bar` values of the response variable:`e_i = y_i - y_bar_i` (or use `resid` function in connection with `lm` function)

The least squares line

- Define the **least squares line** as the line that minimizes the sum of the squared residuals, and list conditions necessary for fitting such line:
  - linearity
  - nearly normal residuals
  - constant variability
- Define an **indicator variable** as a binary explanatory variable (with two levels)
- Calculate the *estimate for the slope* `b_1` as `b_1 = R * s_y / s_x`, where `R` is the correlation coefficient, `s_y` is the *standard deviation of the response variable*, and `s_x` is the *standard deviation of the explanatory variable* (or use `SI.corr.slope` function)
- Interpret the slope as:
  - when x is *numerical*: For each unit increase in `x`, we would expect `y` to be lower/higher on average by `|b_1|` units
  - when x is *categorical*: The value of the response variable is predicted to be `|b_1|` units higher/lower between the baseline level and the other level of the explanatory variable.
  - Note that whether the response variable increases or decreases is determined by the sign of `b_1`
- Note that the least squares line always passes through the *average of the response and explanatory variables* `(mean_x, mean_y)`
- Calculate the estimate for the intercept `b_0` as `b_0 = mean_y - b_1 * mean_x`, where `b_1` is the slope, `mean_y` is the *average of the response variable*, and `mean_x` is the *average of explanatory variable* (or use `SI.corr.intercept` function)
- Interpret the intercept as
  - when x is *numerical*: When `x = 0`, we would expect `y` to equal, on average, `b_0`

    - when x is *categorical*: The expected average value of the response variable for the reference level of the explanatory variable is `b_0`
- Predict the value of the response variable for a given value of the explanatory variable, `x`, by plugging in `x` in the linear model: `y = b_0 + b_1 * x`
    - Only predict for values of x* that are in the range of the observed data.
    - Do not extrapolate beyond the range of the data, unless you are confident that the linear pattern continues.
- Define `R^2` (aka R-squared) as the *percentage of the variability in the response variable explained by the explanatory variable*
    - For a good model, we would like this number to be as close to 100% as possible.
    - This value is calculated as the square of the correlation coefficient

Leverage points

- Define a **leverage point** as a point that lies away from the center of the data in the horizontal direction (i.e., an outlier, obviously)
- Define an **influential point** as a point that influences (changes) the slope of the regression line.
    - This is usually a *leverage point that is away from the trajectory of the rest of the data*
- Do not remove *outliers* from an analysis without good reason
- Be cautious about using a categorical explanatory variable when one of the levels has very few observations, as these may act as influential points

Inference for linear regression

- Determine whether an explanatory variable is a significant *predictor* for the response variable using the t-test and the associated p-value in the regression output
- Set the *null hypothesis testing for the significance of the predictor* as `H0: β_1 = 0`, and recognize that the standard software output yields the p-value for the two-sided alternative hypothesis (use `summary` function for the `lm` output)
    - Note that `β_1 = 0` means the regression line is horizontal, hence suggesting that there is no relationship between the explanatory and response variables
- Calculate the *T score* for the *hypothesis test for the slope* as `T_df = (b_1 - null value) /  SE_b1` with `df = n - 2` (note that the *T score* has `n - 2` *degrees of freedom* since we lose one degree of freedom for each parameter we estimate, and in this case we estimate the intercept and the slope)
- Note that a *hypothesis test for the intercept* is often irrelevant since it's usually out of the range of the data, and hence it is usually an extrapolation
- Calculate a *confidence interval for the slope* as `b_1 ± t_df * SE_b1`, where `df = n - 2` and `t_df` is the critical score associated with the given confidence level at the desired degrees of freedom, and the standard error of the slope estimate `SE_b1` can be found on the regression output (again, see summary of the `lm` output)

Unit 7

Multiple linear regression model

- Define the **multiple linear regression model** as `y_hat =β_0 + β_1 * x_1 + β_2 * x_2 + … + β_k * x_k` where there are `k` *predictors* (*explanatory variables*)
- Interpret the *estimate for the intercept* `β_0` as the expected value of `y` when all predictors are equal to 0, on average
- Interpret the *estimate for a slope* (say `β_1`) as "All else held constant, for each unit increase in `x_1`, we would expect `y` to be higher/lower on average by `β_1`"
- Define **collinearity** as a high correlation between two independent variables such that the two variables contribute redundant information to the model – which is something we want to avoid in multiple linear regression
- Note that `R^2` will increase with each explanatory variable added to the model, regardless of whether or not the added variable is a meaningful predictor of the response variable. Therefore we use **adjusted** `R^2`, which applies a penalty for the number of predictors included in the model, to better assess the strength of a multiple linear regression model: `R_adj^2 = 1 – (SSE / (n – k – 1)) / SST / (n – 1)`, where `n` is the number of cases and `k` is the number of predictors
- Note that `R_adj^2` will only increase if the added variable has a meaningful contribution to the amount of explained variability in `y`, i.e. if the gains from adding the variable exceeds the penalty

Model selection

- Define **model selection** as identifying the best model for predicting a given response variable
- Note that we usually prefer simpler (*parsimonious*) models over more complicated ones
- Define the *full model* as the model with all explanatory variables included as predictors
- The *significance of the model as a whole* is assessed using an F-test:
  - `H_0: β_1 = β_2 = … = β_k`
  - `H_A: At least one β_i ≠ 0`
  - Degrees of freedom: `df = n – k – 1`
  - Usually reported at the bottom of the regression output (`summary(model)`)
- Note that the *p-values* associated with each predictor are conditional on other variables being included in the model, so they can be used to assess if a given predictor is significant, given that all others *are*:
  - `H_0: β_1 = 0`, given all other variables are included in the model
  - `H_A: β_1 ≠ 0`, given all other variables are included in the model
  - These p-values are calculated based on a *t distribution* with `n – k – 1` degrees of freedom
  - The same degrees of freedom can be used to construct a confidence interval for the slope parameter of each predictor `b_i ± t * SE_bi`, where `t` is a *t statistic* computed with `n – k – 1` degrees of freedom
- Stepwise model selection (backward or forward) can be done based on *p-values* (drop variables that are *not significant*) or based on *adjusted* `R^2` (choose the model with higher *adjusted* `R^2`)
- The general idea behind *backward-selection* is to start with the full model and eliminate one variable at a time until the ideal model is reached:
  - *p-value method*:
    - 1. Start with the full model
    - 2. Drop the variable with the highest p-value and refit the model
    - 3. Repeat until all remaining variables are significant
  - *adjusted* `R^2` *method*:
    - 1. Start with the full model
    - 2. Refit all possible models omitting one variable at a time, and choose *the model with the highest adjusted* `R^2`
    - 3. Repeat until maximum possible adjusted `R^2` is reached
- The general idea behind *forward-selection* is to start with only one variable and adding one variable at a time

until the ideal model is reached

  - *p-value method*:
    - 1. Try all possible simple linear regression models predicting y using one explanatory variable at a time. Choose the model where the explanatory variable of choice has the lowest p-value
    - 2. Try all possible models adding one more explanatory variable at a time, and choose the model where the added explanatory variable has the lowest p-value
    - 3. Repeat until all added variables are significant
  - *adjusted* `R^2` *method*:
    - 1. Try all possible simple linear regression models predicting y using one explanatory variable at a time. Choose the model with the highest adjusted `R^2`
    - 2. Try all possible models adding one more explanatory variable at a time, and choose the model with the highest adjusted `R^2`
    - 3. Repeat until maximum possible adjusted `R^2` is reached
- Note that the *adjusted* `R^2` *method* is more computationally intensive, but it is more reliable, since it doesn't depend on an *arbitrary significance level*

Conditions for the multiple linear regression model

- Linear relationship between each (numerical) *explanatory variable* and the *response*
  - Check using scatterplots of y vs. each x (see `pairs.panel` function from `psych` library), and residuals plots of residuals vs. each x (looking for a complete random scatter around zero): `plot(model$residuals ~ initial_data$x)`
- Nearly normal residuals with mean 0
  - Check using a normal probability plot and histogram of residuals: `hist(model$residuals)`, `qqplot(model$residuals) ; qqline(model$residuals)`
- Constant variability of residuals
  - Check using residuals plots of residuals vs. `y_hat` aka the *predicted values of the response variable* (we want the residuals to be randomly scattered in a band with a constant width around zero): `plot(model$residuals ~ model$fitted)` (no "fan shape" is expected here)
  - It is also worthwhile the absolute value of residuals versus the predicted values to identify any unusual observations easily: `plot(abs(model$residuals) ~ model$fitted)` (no "triangle" is expected here)
  - Besides, may want to check residuals vs. each x (yet aforementioned methods allow to consider the entire model (with all explanatory variables) at once)
- Independence of residuals (and hence observations)
  - Check using a scatterplot of residuals vs. order of data collection (will reveal non-independence if data have time series structure): `plot(model$residuals)` (no pattern of any kind are expected)

Appendix

```
# SKEWED NORMAL DISTRIBUTION ####

# Generates a skewed normal distribution
SI.rnorm_skewed <- function(coef = 1, direction = "right", mu = 0, sigma = 1) {
    # generates a skewed normal distribution given the coef of skewness and its direction
    direction <- switch (direction,
                         right =  1,
                         left = -1)
    RNORM_SKEWED <- rnorm(5000, mu, sigma) + 3 * sigma
    RNORM_SKEWED <- RNORM_SKEWED[RNORM_SKEWED > 0 & RNORM_SKEWED < 6 * sigma]
    RNORM_SKEWED <- direction * (RNORM_SKEWED ^ (coef + 1))
    hist(RNORM_SKEWED)
    abline(v=mean(RNORM_SKEWED), col="red")
    abline(v=median(RNORM_SKEWED), col="blue")
    return(RNORM_SKEWED)
}

# BINOMIAL DISTRIBUTIONS ####

# Converts a Binomial Distribution to a Normal Distribution
SI.bitono <- function(n, p) {
    # converts a binomial distribution to a normal distribution
    # given its success probability and a number of trials
    mu <- n * p
    si <- sqrt(mu * (1 - p))
    vec <- c(mu, si)
    return(vec)
}

# STANDARD ERROR ####

# Caclulates the standard error for a normal distribution
SI.standart_error <- function(sigma, sample_size) {
    # caclulates the standard error given the variance and the size of a sample
    tmp <- sigma^2/sample_size
    if (length(sigma) > 1) {
        se <- sqrt(sum(tmp))
    } else {
        se <- sqrt(tmp)
    }
    return(se)
}

# Caclulates the standard error for the distribution of proportions
SI.prop.standart_error <- function(population_proportion, sample_size, pool = FALSE) {
    # caclulates the standard error given the proportion of a population and the size of a sampl
e
    if (pool) {
        pooled_proportion <- sum(population_proportion) / sum(sample_size)
        return( sqrt(
                (pooled_proportion * (1 - pooled_proportion) / sample_size[1]) +
                (pooled_proportion * (1 - pooled_proportion) / sample_size[2])
                ) )
    }
    tmp <- (population_proportion * (1 - population_proportion)) / sample_size
    if (length(population_proportion) > 1) {
        se <- sqrt(sum(tmp))
    } else {
        se <- sqrt(tmp)
    }
    return(se)
}

# CONFIDENCE INTERVALS ####

# Important Reminder:
#
#   Commonly used confidence levels (CLs) are 90%, 95%, 98%, and 99%
#   "...With the confidence level A we state that the answer lies in B..."
#

# Calculates the CI for a population
SI.confidence_intreval <- function(confidence_level, mu, standard_error) {
    # calculates the CI of the population given the CL and the mean and the standard error of a sample
    value <- (1 - confidence_level) / 2
    return(c(qnorm(value, mu, standard_error), qnorm(confidence_level + value, mu, standard_error)))
}

# Calculates the needed sample size
SI.sample_size <- function(margin_of_error, sigma, confidence_level) {
    # calculates the needed sample size, given the standard deviation of a population,
    # the desired margin of error and the confidence level of a projected sample
    z <- qnorm((1 - confidence_level) / 2)
    size <- ((z * sigma) / margin_of_error) ^ 2
    return(ceiling(size))
}

SI.prop.sample_size <- function(margin_of_error, population_proportion, confidence_level) {
    # calculates the needed sample size, given the standard deviation of a population,
    # the desired margin of error and the confidence level of a projected sample
    z <- qnorm((1 - confidence_level) / 2)
    return(ceiling(z^2 * population_proportion * (1 - population_proportion) / margin_of_error ^ 2))
}

# Important Reminder:
#
#   pnorm() takes values from a distribution as an input, and returns the probability with which that value
#           occurs in a given distribution
#   qnorm() - on the contrary - takes the probability as an input, and returns the value occurring with
#           that probability in a given distribution
#

# HYPOTHESIS TESTING ####
```

```r
# Calculates the z-score
SI.z.score <- function(test_value, mu, standard_error) {
    # calculates the z-score, given the proposed test value abd the mean and the standard error of a sample
    return((test_value - mu)/standard_error)
}

# Calculates the p-value
SI.pvalue <- function (null_hypothesis, standard_error, test_value_left = NULL, test_value_right = NULL) {
    # calculates the p-value, given the null hypothesis, the standard error and the test values
    if (is.null(test_value_left) && is.null(test_value_right)) {
        stop("No values to test provided")
    }
    if (!is.null(test_value_left) && is.null(test_value_right)) {
        return(pnorm((test_value_left - null_hypothesis)/standard_error))
    }
    if (is.null(test_value_left) && !is.null(test_value_right)) {
        return(pnorm((test_value_right - null_hypothesis)/standard_error, lower.tail = F))
    }
    if (!is.null(test_value_left) && !is.null(test_value_right)) {
        return( pnorm((test_value_left - null_hypothesis)/standard_error, lower.tail = T)
                + pnorm((test_value_right - null_hypothesis)/standard_error, lower.tail = F) )
    }
}

# Calculates p-value using standard deviation
SI.pvalue.sd <- function (null_hypothesis, sigma, sample_size,
                          test_value_left = NULL, test_value_right = NULL) {
    # calculates p-value, given the mean and the variance of a population and the size of a sample
    if (is.null(test_value_left) && is.null(test_value_right)) {
        stop("No values to test provided")
    }
    if (!is.null(test_value_left) && is.null(test_value_right)) {
        return(pnorm((test_value_left - null_hypothesis)/(sigma/sqrt(sample_size))))
    }
    if (is.null(test_value_left) && !is.null(test_value_right)) {
        return(pnorm((test_value_right - null_hypothesis)/(sigma/sqrt(sample_size)), lower.tail = F))
    }
    if (!is.null(test_value_left) && !is.null(test_value_right)) {
        return( pnorm((test_value_left - null_hypothesis)/(sigma/sqrt(sample_size)), lower.tail = T)
                + pnorm((test_value_right - null_hypothesis)/(sigma/sqrt(sample_size)), lower.tail = F) )
    }
}

# BOOTSTRAPPING ####

# Generates a bootstrapping distribution
SI.boot <- function(data, boot_size, statistic) {
    # generates a bootstrapping distribution of a statistic given the original data and the required size
    boot <- rep(NA, boot_size)
    for (i in 1:boot_size) {
        boot[i] <- match.fun(statistic)(sample(data, length(data), replace = T))
    }
    return(boot)
}

# Calculates the CI for a bootstrapping distribution
SI.boot.confidence_interval <- function(data, confidence_interval = 0.9, method = "percentile")
{
    # calculates the CI given the bootstrapping distribution and the required interval
    value <- (1 - confidence_interval) / 2
    if (method == "percentile") {
        quantile(data, c(value, 1-value))
    } else {
        if (method == "se") {
            mean <- mean(data)
            sd <- sd(data)
            v1 <- mean + qnorm(value) * sd
            v2 <- mean + qnorm(1-value) * sd
            rez <- c(v1, v2)
            names(rez) <- c(paste(value * 100, "%", sep=""), paste((1-value) * 100, "%", sep=""))
            return(rez)
        } else {
            stop("Invaid method")
        }
    }
}

# STUDENT'S T DISTRIBUTION ####

# Calculates the t-score
SI.t.score <- function(confidence_level = NULL, confidence_interval = NULL, sample_size) {
    # calculates the t-score, given the confidence level or the confidence interval & the size of a sample
    if (is.null(confidence_level) && is.null(confidence_interval)) {
        stop("Please, provide thr confidence level or the confidence interval")
    }
    if (is.null(confidence_interval)) {
        return( qt((1 - (1 - confidence_level) / 2), sample_size - 1) )
    }
    if (is.null(confidence_level)) {
        return( qt((1 - (1 - confidence_interval) / 2), sample_size - 1) )
    }

}

# Calculates the CI for a population
SI.t.confidence_intreval <- function(confidence_level, mean, standard_error, sample_size) {
    # calculates the CI of the population given the CL and the mean, the sd and the size of a sample
    tscore <- qt((1 - confidence_level) / 2, sample_size - 1)
    return( c(mean + tscore * standard_error, mean - tscore * standard_error ) )
}

# Calculates p-value
```

```
SI.t.pvalue <- function(null_hypothesis, standard_error, sample_size,
                        test_value_left = NULL, test_value_right = NULL) {
    # calculates the p-value, given the null hypothesis, the standard error and the test values
    if (is.null(test_value_left) && is.null(test_value_right)) {
        stop("No values to test provided")
    }
    df <- min(sample_size - 1)
    if (!is.null(test_value_left) && is.null(test_value_right)) {
        return(pt((test_value_left - null_hypothesis)/standard_error, df))
    }
    if (is.null(test_value_left) && !is.null(test_value_right)) {
        return(pt((test_value_right - null_hypothesis)/standard_error, df, lower.tail = F))
    }
    if (!is.null(test_value_left) && !is.null(test_value_right)) {
        return( pt((test_value_left - null_hypothesis)/standard_error, df, lower.tail = T)
                + pt((test_value_right - null_hypothesis)/standard_error, df, lower.tail = F) )
    }
}

# ANOVA ####

# Compute analysis of variance (or deviance) tables for one or more fitted model objects
SI.anova <- function(list_of_distributions) {
    # compute analysis of variance tables given a list of distributions

    # analyze the data
    lst <- list_of_distributions
    len <- length(lst)
    dat <- unlist(lst)
    lab <- c()
    # transform the data according to the analysis
    for (i in 1:len) {
        vec <- rep(labels(lst)[i], length(lst[[i]]))
        lab <- append(lab, vec)
    }
    # create a model
    fit <- lm(dat ~ lab)
    # supply the model to `anova` function
    return(anova(fit))
}

SI.anova_pairwise <- function(list_of_distributions, significance_level = 0.05) {
    lst <- list_of_distributions
    len <- length(lst)
    # number of comparisons: (number of groups) * (number of groups - 1) / 2
    num <- len * (len - 1) / 2
    # correction Bonferroni: (significance level) / (number of comparisons)
    BC <- significance_level / num
    # to calculate SE for pairwise comparisons we need to calculate MSE:
    mean_list <- lapply(lst, mean); mean_list
    # calculate the mean for all elements
    mean_all <- mean(unlist(lst)) ; mean_all
    # calculate SST
    st <- lapply(lst, function(x) x <- (x - mean_all)^2)
    # sapply applies a function (sum) to a list (st), returns its results as a vector, not as li
st (lapply)
```

```r
    sst <- sum(sapply(st, sum)) ; sst
    # calculate SSG
    sg <- lapply(mean_list, function(x) x <- (x - mean_all)^2)
    counts <- lapply(lst, length)
    ssg <- sum(unlist(sg) * unlist(counts))
    ssg
    # calculate SSE
    sse <- sst - ssg
    # calculate degrees of freedom
    dft <- length(unlist(lst)) - 1
    dfg <- length(lst) - 1
    dfe <- dft - dfg
    # calculate means squares
    msg <- ssg / dfg
    mse <- sse / dfe ; mse
    # get names for every pair of distributions
    names <- combn(c(1:len), 2, simplify = F) ; names
    names <- as.character(names) ; names
    names <- gsub(":", ", ", gsub("c|\\(|\\)", "", names)) ; names
    # get list of lengths for every pair of distributions
    len_pairs <- combn(sapply(lst, length), 2, simplify = F)
    # separate it
    len_pairs1 <- lapply(len_pairs, "[[", 1)
    len_pairs2 <- lapply(len_pairs, "[[", 2)
    # produce a vector of SEs for every pair of distributions
    ses <- sqrt(sapply(len_pairs1, function(x) mse/x) + sapply(len_pairs2, function(x) mse/x))
    # produce a vector of the corresponding mean differences
    mean_pairs <- combn(sapply(lst, mean), 2, simplify = F)
    mean_diffs <- sapply(lapply(mean_pairs, rev), diff)
    # produce a vector of T statistics
    t <- mean_diffs / ses ;
    # convert negative values to positive in place them to the right tail
    t <- abs(t)
    # calculate p-values for these pairs (for degrees of freedom we use dfe)
    p <- 2 * pt(t, dfe, lower.tail = F)
    # assign corresponding names
    names(p) <- names ; p
    # compare to Bonferroni correction
    fail <- p[p > BC]
    succ <- p[p < BC]
    print("The following list of distributions was provided")
    print(str(lst))
    if (!is.null(succ)) {
        print("Null hyphothesis successfully rejected for the pair(s): ")
        print(succ)
    }
    if (!is.null(succ)) {
        print("Failed to reject null hyphothesis for the pair(s): ")
        print(fail)
    }
}

# CHI-SQUARE ####

# Calculates the chi-square
SI.chisq <- function(observed, expected) {
    # calculates the chi-square given the vectors of observed and expected outcomes
    if (length(observed) != length(expected)) {
        stop("Check data, please")
    }
    return(sum( (observed - expected)^2 / expected ))
}

# Calculates the p-value for a chi-square distribution
SI.chisq.pvalue <- function (chisq, number_of_levels) {
    # calculates the p-value given the chi-square statistic and the number of categorical levels
    # goodness of fit: one categorical variable, more than two levels
    return(pchisq(chisq, number_of_levels - 1, lower.tail = F))
}

SI.chisq.independence_test <- function (chisq, number_of_levels) {
    df <- (number_of_levels[1] - 1) * (number_of_levels[2] - 1)
    return(pchisq(chisq, df, lower.tail = F))
}

# MULTIPLE LINEAR REGRESSION MODELS ####

# Fits separate models for separate categories of a categorical variable on a scatterplot
SI.model.cat_separated <- function(model, ...){
    # fits separate models for separate categories of a categorical variable on a scatterplot
    # the code provided by DASI Team
    if(class(model)!="lm"){
        warning("Model must be the output of the function lm()")
    }

    if(length(model$xlevels)!=1){
        warning("Model must contain exactly one categorical predictor")
    }

    if(length(model$coef)-length(model$xlevels[[1]])!=1){
        warning("Model must contain exactly one non-categorical predictor")
    }

    palette <- c("#E69F00", "#56B4E9", "#D55E00", "#009E73", "#CC79A7", "#F0E442", "#0072B2")

    baseIntercept <- model$coef[1]
    nLines <- length(model$xlevels[[1]])
    intercepts <- c(baseIntercept, rep(0, nLines-1))
    indicatorInd <- c(1, rep(0, nLines)) # used to find slope parameter by process of elimination

    for(i in 1:(nLines-1)){
        indicatorName <- paste(names(model$contrasts),model$xlevels[[1]][1+i], sep = "")
        intercepts[i+1] <- baseIntercept + model$coef[names(model$coef)==indicatorName]
        indicatorInd <- indicatorInd + (names(model$coef)==indicatorName)
    }

    slope <- model$coef[!indicatorInd]

    num_pred = which(names(model$model[,-1]) != names(model$xlevels)) + 1
    cat_pred = which(names(model$model[,-1]) == names(model$xlevels)) + 1
    model$model$COL = NA
    model$model$PCH = NA
    for(i in 1:nLines){
        model$model$COL[model$model[,cat_pred] == levels(model$model[,cat_pred])[i]] = adjustco
lor(palette[i],0.40)
        model$model$PCH[model$model[,cat_pred] == levels(model$model[,cat_pred])[i]] = i+14
    }

    plot(model$model[,1] ~ jitter(model$model[,num_pred]), col = model$model$COL, pch = model$m
odel$PCH,
         ylab = names(model$model)[1],
         xlab = names(model$model)[num_pred])

    for(j in 1:nLines){
        abline(intercepts[j], slope, col = palette[j], lwd = 2, ...)
    }

    if(slope > 0){legend_pos = "bottomright"}
    if(slope < 0){legend_pos = "topleft"}

    legend(legend_pos, col = palette[1:nLines], lty = 1, legend = levels(model$model[,cat_pred]
), lwd = 2)
}
```

Footnotes

1. For some additional information on ANOVA and `inference` function see here.↩