One- and two-sample tests

In statistics, a one-sample test and a two-sample test are both hypothesis tests used to compare sample data with a population or to compare two independent sample groups, respectively.

One-Sample Test:

A one-sample test is used when you have a single sample and want to determine whether it is significantly different from a known population or a specific value. The goal is to assess whether the sample is representative of the population or if there is a statistically significant difference between the sample and the population.

Two-Sample Test:

A two-sample test is employed when you have two independent samples and wish to determine whether they have different means or distributions. The goal is to assess whether there is a statistically significant difference between the two sample groups.

Here is an example concerning daily energy intake in kJ for 11 women (Altman, 1991, p. 183). First, the values are placed in a data vector:

In [1]:
```
daily_intake = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 87
```

Out[1]:
```
11-element Vector{Int64}:
 5260
 5470
 5640
 6180
 6390
 6515
 6805
 7515
 7515
 8230
 8770
```

Firstly I looked at some simple summary statistics, even though these were hardly necessary for such a small data set

The mean of a daily intake dataset refers to the average value of the data points in the dataset, representing the average daily intake of a particular variable or quantity.

In [2]:
```
mean_value = sum(daily_intake) / length(daily_intake)
```

Out[2]:
```
6753.636363636364
```

In statistics, variance is a measure of the variability or spread of a dataset. It quantifies how far each value in the dataset is from the mean (average) of the dataset. The variance is calculated as the average of the squared differences between each data point and the mean.

In [3]:
```
variance_value = sum((daily_intake .- mean_value).^2) / (length(daily_intake)
```

Out[3]:
```
1.3044454545454546e6
```

In statistics, the standard deviation is a measure of the amount of variation or dispersion in a dataset. It quantifies how spread out the values are from the mean (average) of the dataset. The standard deviation is calculated as the square root of the variance

In [1]:
```
using Statistics

daily_intake = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 87
std_deviation = std(daily_intake)
println("Standard Deviation: ", std_deviation)
```

```
Standard Deviation: 1142.1232221373727
```

In statistics, quantiles are values that divide a dataset into equal-sized intervals. They provide information about the relative position of individual data points within the dataset. Commonly used quantiles include the median (quantile 0.5), quartiles (quantiles 0.25 and 0.75), and the minimum (quantile 0) and maximum (quantile 1).

In [6]:
```
using Statistics
quantile_value = quantile(daily_intake, [0, 0.25, 0.5, 0.75, 1])
```

Out[6]:
```
5-element Vector{Float64}:
 5260.0
 5910.0
 6515.0
 7515.0
 8770.0
```

Investigate whether the women's energy intake deviates systematically from a recommended value of 7725 kJ.Thatdata come from a normal distribution, the object is to test whether thisdistribution might have mean μ = 7725. This was done with t.test asfollows

In [7]:
```
#using Pkg
#Pkg.add("StatsBase")
```

In [8]:
```
#using Pkg
#Pkg.add("HypothesisTests")
```

One-sample t test

In [9]:
```
using HypothesisTests

daily_intake = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 87
t_test_result = OneSampleTTest(daily_intake, 7725)
```

Out[9]:
```
One sample t-test
-----------------
Population details:
    parameter of interest:   Mean
    value under h_0:         7725
    point estimate:          6753.64
    95% confidence interval: (5986.0, 7521.0)

Test summary:
    outcome with 95% confidence: reject h_0
    two-sided p-value:           0.0181

Details:
    number of observations:   11
    t-statistic:              -2.8207540608310198
    degrees of freedom:       10
    empirical standard error: 344.3631083801271
```

I got the t statistic, the associated degrees of freedom, and the exact p-value.Quantiles the t-value can be found,seen that p < 0.05 and thus that (using the customary 5% level of significance) data deviate significantly from the hypothesis that the mean is 7725.This contains two important pieces of information: (a) the value we wanted to test whether the mean could be equal to (7725 kJ) and (b) that the test is two-sided (“not equal to”) This is a 95% confidence interval for the true mean; that is, the set of (hypothetical) mean values from which the data do not deviate significantly.It is based on inverting the t test by solving for the values of μ0 that cause to lie within its acceptance region. For a 95% confidence interval.the mean value μ under the null hypothesis (default is mu=0). In addition, you can specify that a one-sided test is desired against alternatives greater than μ by using alternative="greater" or alternatives less than μ using alternative="less". The third item that can be specified is the confidence level used for the confidence intervals;write conf.level=0.99 to get a 99% interval.

Wilcoxon signed-rank test

Wilcoxon test, the procedure is to subtract the theoretical μ0 and rank the differences according to their numerical value,ignoring the sign, and then calculated the sum of the positive or negativeranks. The point is that, assuming only that the distribution is symmetricaround μ0, the test statistic corresponds to selecting each number from 1 to n with probability 1/2 and calculating the sum. The distribution of thetest statistic can be calculated exactly, at least in principle. It becomes computationally excessive in large samples, but the distribution is then very well approximated by a normal distribution.

In [10]:
```
using HypothesisTests

daily_intake = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 87
test_result = SignTest(daily_intake .- 7725)
```

Out[10]:
```
Sign Test
---------
Population details:
    parameter of interest:   Median
    value under h_0:         0.0
    point estimate:          -1210.0
    95% confidence interval: (-2085.0, -210.0)

Test summary:
    outcome with 95% confidence: fail to reject h_0
    two-sided p-value:           0.0654

Details:
    number of observations:       11
    observations larger than 0.0: 2
```

In statistical hypothesis testing, the p-value is a measure that helps assess the strength of evidence against a null hypothesis. It quantifies the probability of observing a test statistic as extreme as, or more extreme than, the one calculated from the data, assuming that the null hypothesis is true.

In [11]:
```
using HypothesisTests

daily_intake = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 87
test_result = SignTest(daily_intake .- 7725)

p_value = pvalue(test_result)
```

Out[11]:
```
0.06542968750000006
```

The test statistic V is the sum of the positive ranks. In the example, the p-value is computed from the normal approximation because of the tie at 0654.

Two-sample t test

A two-sample t-test, also known as an independent samples t-test, is a statistical test used to compare the means of two independent groups. It assesses whether there is a significant difference between the means of the two groups.

In [12]:
```
#using Pkg
#Pkg.add("DataFrames")
```

Here used daily energy expenditure data the problem of comparing energy expenditures between lean and obese women.information is contained in two parallel columnsof a data frame. The factor stature contains the group and the numeric variable expend the energy expenditure in mega-Joules.

In [3]:
```
using DataFrames

# Create the DataFrame
energy = DataFrame(
    expend = [9.21, 7.53, 7.48, 8.08, 8.09, 10.15, 8.40, 10.88, 6.13, 7.90, 11
    stature = ["obese", "lean", "lean", "lean", "lean", "lean", "lean", "lean"
)
```

Out[3]: 22 rows × 2 columns

	expend	stature
	Float64	String
1	9.21	obese
2	7.53	lean
3	7.48	lean
4	8.08	lean
5	8.09	lean
6	10.15	lean
7	8.4	lean
8	10.88	lean
9	6.13	lean
10	7.9	lean
11	11.51	obese
12	12.79	obese
13	7.05	lean
14	11.85	obese
15	9.97	obese
16	7.48	lean
17	8.79	obese
18	9.69	obese
19	9.68	obese
20	7.58	lean
21	9.19	obese
22	8.11	lean

In [2]:
```
import Pkg
Pkg.add("Plots")
```

```
    Updating registry at `C:\Users\Lenovo\.julia\registries\General.toml`
   Resolving package versions...
  No Changes to `C:\Users\Lenovo\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\Lenovo\.julia\environments\v1.8\Manifest.toml`
```

In [18]:
```
#using Pkg
#Pkg.add("HypothesisTests")
```

In [19]:
```
using StatsBase
```

In [20]:
```
#Pkg.add("StatsModels")
```

Welch's t-test, also known as the unequal variances t-test or Welch's unequal variances t-test, is a statistical test used to compare the means of two independent groups when the variances of the two groups are not assumed to be equal. It is an extension of the traditional independent samples t-test, which assumes equal variances between the groups.

In [21]:
```
using DataFrames
using Statistics

# Create the DataFrame
energy = DataFrame(
    expend = [9.21, 7.53, 7.48, 8.08, 8.09, 10.15, 8.40, 10.88, 6.13, 7.90, 11
    stature = ["obese", "lean", "lean", "lean", "lean", "lean", "lean", "lean"
)

# Separate the "expend" values for each group
obese_expend = energy.expend[energy.stature .== "obese"]
lean_expend = energy.expend[energy.stature .== "lean"]

# Calculate sample means
mean_obese = mean(obese_expend)
mean_lean = mean(lean_expend)

# Calculate sample variances
var_obese = var(obese_expend)
var_lean = var(lean_expend)

# Calculate sample sizes
n_obese = length(obese_expend)
n_lean = length(lean_expend)

# Calculate Welch's t-statistic
t_stat = (mean_obese - mean_lean) / sqrt(var_obese/n_obese + var_lean/n_lean)

# Calculate degrees of freedom using Welch-Satterthwaite equation
dof = (var_obese/n_obese + var_lean/n_lean)^2 / ((var_obese^2)/(n_obese^2 * (n_



# Print the test result
println("Welch's t-test:")
println("t = ", t_stat)
println("df = ", dof)
```

```
Welch's t-test:
t = 3.8555035589737
df = 15.918736196767659
```

the one-sample test. The confidence interval is for the difference in means and does not contain 0, which is in accordance with the p-value indicating a significant difference at the 5% level. It isWelch's variant of the t test that is calculated by default. This is the test where you do not assume that the variance is the same in the two groups,which (among other things) results in the fractional degrees of freedom.Here we can found t = 3.8555035589737 df = 15.918736196767659

Comparison of variances

In [22]:
```
using Statistics
using Distributions

# Define the two groups
group_lean = energy.expend[energy.stature .== "lean"]
group_obese = energy.expend[energy.stature .== "obese"]

# Calculate the F-statistic and degrees of freedom
var_lean = var(group_lean)
var_obese = var(group_obese)
f_statistic = var_lean / var_obese
num_df = length(group_lean) - 1
denom_df = length(group_obese) - 1

# Calculate the p-value
p_value = ccdf(FDist(num_df, denom_df), f_statistic)

# Print the results
println("F Test to Compare Variances")
println("data: expend by stature")
println("F = $(f_statistic), num df = $num_df, denom df = $denom_df, p-value =
println("alternative hypothesis: true ratio of variances is not equal to 1")
```

```
F Test to Compare Variances
data: expend by stature
F = 0.7844459792357033, num df = 12, denom df = 8, p-value = 0.66012700731196
58
alternative hypothesis: true ratio of variances is not equal to 1

WARNING: using Distributions.dof in module Main conflicts with an existing id
entifier.
```

When comparing variances, there are several statistical tests that can be used to assess whether two or more groups have significantly different variances. The choice of test depends on the specific characteristics of the data and the research question at hand. Here I used F test

Two-sample Wilcoxon test

The Two-sample Wilcoxon test, also known as the Wilcoxon rank-sum test or the Mann-Whitney U test, is a nonparametric statistical test used to compare the distributions of two independent samples. It is commonly used when the data are not normally distributed or when the assumptions of parametric tests, such as the t-test, are not met.

In [23]:
```
using HypothesisTests

expend = [9.21, 7.53, 7.48, 8.08, 8.09, 10.15, 8.40, 10.88, 6.13, 7.90, 11.51,
stature = ["obese", "lean", "lean", "lean", "lean", "lean", "lean", "lean", "l

# Perform the Mann-Whitney U test
result = MannWhitneyUTest(expend[stature .== "obese"], expend[stature .== "lea

# Print the test result
println("Mann-Whitney U test")
println("data: expend by stature")
println("U = ", result)
println("p-value = ", result.pvalue)
println("alternative hypothesis: true location shift is not equal to 0")
```

```
Mann-Whitney U test
data: expend by stature
U = Approximate Mann-Whitney U test
-------------------------------
Population details:
    parameter of interest:   Location parameter (pseudomedian)
    value under h_0:         0
    point estimate:          1.79

Test summary:
    outcome with 95% confidence: reject h_0
    two-sided p-value:           0.0021

Details:
    number of observations in each group: [9, 13]
    Mann-Whitney-U statistic:             105.0
    rank sums:                            [150.0, 103.0]
    adjustment for ties:                  6.0
    normal approximation (μ, σ):          (46.5, 14.9708)


type ApproximateMannWhitneyUTest has no field pvalue

Stacktrace:
 [1] getproperty(x::ApproximateMannWhitneyUTest{Float64}, f::Symbol)
   @ Base .\Base.jl:38
 [2] top-level scope
   @ In[23]:13
```

In [24]:
```
#using Pkg
#Pkg.add("CSV")
#Pkg.add("DataFrames")
```

Daily Intake Dataset

In [25]:
```
using DataFrames

# Define the data
pre = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 8770]
post = [3910, 4220, 3885, 5160, 5645, 4680, 5265, 5975, 6790, 6900, 7335]

# Create the DataFrame
df = DataFrame(pre=pre, post=post)
```

Out[25]: 11 rows × 2 columns

	pre	post
	Int64	Int64
1	5260	3910
2	5470	4220
3	5640	3885
4	6180	5160
5	6390	5645
6	6515	4680
7	6805	5265
8	7515	5975
9	7515	6790
10	8230	6900
11	8770	7335

The point is that the same 11 women are measured twice, so it makes sense to look at individual differences

In [26]:
```
using DataFrames

# Define the data
pre = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 8770]
post = [3910, 4220, 3885, 5160, 5645, 4680, 5265, 5975, 6790, 6900, 7335]

# Calculate the differences between post and pre measurements
differences = post .- pre

# Create a DataFrame with the differences
df_differences = DataFrame(differences=differences)
```

Out[26]: 11 rows × 1 columns

	differences
	Int64
1	-1350
2	-1250
3	-1755
4	-1020
5	-745
6	-1835
7	-1540
8	-1540
9	-725
10	-1330
11	-1435

It is immediately seen that they are all negative. All the women have a lower energy intake postmenstrually than premenstrually. The paired t test is obtained as follows:

one sample t test

In [27]:
```
pre = [5260, 5470, 5640, 6180, 6390, 6515, 6805, 7515, 7515, 8230, 8770]
post = [3910, 4220, 3885, 5160, 5645, 4680, 5265, 5975, 6790, 6900, 7335]

differences = pre .- post
result = OneSampleTTest(differences)
```

Out[27]:
```
One sample t-test
-----------------
Population details:
    parameter of interest:   Mean
    value under h_0:         0
    point estimate:          1320.45
    95% confidence interval: (1074.0, 1567.0)

Test summary:
    outcome with 95% confidence: reject h_0
    two-sided p-value:           <1e-06

Details:
    number of observations:   11
    t-statistic:              11.941392877647603
    degrees of freedom:       10
    empirical standard error: 110.57793332687574
```

There is not much new to say about the output; it is virtually identical to that of a one-sample t test on the elementwise differences.

Regression and correlation

Simple linear regression:-Simple linear regression is a statistical technique used to model the relationship between two variables: a dependent variable (also known as the response variable or the outcome) and an independent variable (also known as the predictor variable or the input). It assumes a linear relationship between the two variables, meaning that the relationship can be represented by a straight line.

The linear regression model is given by yi = a + bxi + ei

In [28]:
```
short_velocity, blood_glucose=blood_glucose)
```

Out[28]: 24 rows × 2 columns

	short_velocity	blood_glucose
	Float64	Float64
1	1.76	15.3
2	1.34	10.8
3	1.27	8.1
4	1.47	19.5
5	1.27	7.2
6	1.49	5.3
7	1.31	9.3
8	1.09	11.1
9	1.18	7.5
10	1.22	12.2
11	1.25	6.7
12	1.19	5.2
13	1.95	19.0
14	1.28	15.1
15	1.52	6.7
16	NaN	8.6
17	1.12	4.2
18	1.37	10.3
19	1.19	12.5
20	1.05	16.1
21	1.32	13.3
22	1.03	4.9
23	1.12	8.8
24	1.7	9.5

In [29]:
```
#import Pkg
#Pkg.add("GLM")
```

In [30]:
```
#import Pkg
#Pkg.add("Lathe")
```

Train-Test Split

"Splitting the data" and "training the data" are two important steps in the process of building and evaluating predictive models, including linear regression models

In [31]:
```
# Train test split
using Lathe.preprocess: TrainTestSplit
train, test = TrainTestSplit(df,.75)
```

Out[31]:
```
(18×2 DataFrame
 Row │ short_velocity │ blood_glucose │
     │ Float64        │ Float64       │
─────┼────────────────┼───────────────┼
   1 │ 1.76           │ 15.3          │
   2 │ 1.27           │ 8.1           │
   3 │ 1.27           │ 7.2           │
   4 │ 1.49           │ 5.3           │
   5 │ 1.31           │ 9.3           │
   6 │ 1.09           │ 11.1          │
   7 │ 1.18           │ 7.5           │
   8 │ 1.22           │ 12.2          │
   9 │ 1.25           │ 6.7           │
  10 │ 1.19           │ 5.2           │
  11 │ 1.28           │ 15.1          │
  12 │ 1.12           │ 4.2           │
  13 │ 1.19           │ 12.5          │
  14 │ 1.05           │ 16.1          │
  15 │ 1.32           │ 13.3          │
  16 │ 1.03           │ 4.9           │
  17 │ 1.12           │ 8.8           │
  18 │ 1.7            │ 9.5           │, 6×2 DataFrame
 Row │ short_velocity │ blood_glucose │
     │ Float64        │ Float64       │
─────┼────────────────┼───────────────┼
   1 │ 1.34           │ 10.8          │
   2 │ 1.47           │ 19.5          │
   3 │ 1.95           │ 19.0          │
   4 │ 1.52           │ 6.7           │
   5 │ NaN            │ 8.6           │
   6 │ 1.37           │ 10.3          │)
```

Model Building

Linear regression analysis is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, meaning that the relationship can be represented by a straight line

In [45]:
```
using DataFrames
using StatsModels, GLM

# Define the vectors
short_velocity = [1.76, 1.34, 1.27, 1.47, 1.27, 1.49, 1.31, 1.09, 1.18, 1.22,
blood_glucose = [15.3, 10.8, 8.1, 19.5, 7.2, 5.3, 9.3, 11.1, 7.5, 12.2, 6.7, 5

# Create the data frame
df = DataFrame(short_velocity=short_velocity, blood_glucose=blood_glucose)

# Perform linear regression
model = lm(@formula(short_velocity ~ blood_glucose), df)

# Get the model summary
model
```

```
WARNING: using GLM.dof in module Main conflicts with an existing identifier.
```

Out[45]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GL
M.DensePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float6
4}, Vector{Int64}}}}, Matrix{Float64}}

short_velocity ~ 1 + blood_glucose

Coefficients:
──────────────────────────────────────────────────────────────────────────────
                Coef.  Std. Error    t  Pr(>|t|)  Lower 95%  Upper 95%
──────────────────────────────────────────────────────────────────────────────
(Intercept)       NaN         NaN  NaN       NaN        NaN        NaN
blood_glucose     NaN         NaN  NaN       NaN        NaN        NaN
──────────────────────────────────────────────────────────────────────────────
```

In [34]:
```
#using Pkg
#Pkg.add("RCall")
#ENV["R_HOME"]="C:/Users/Lenovo/ANACON~1/envs/rstudio/lib/R"
#Pkg.build("RCall")
```

In [35]:
```
using RCall
```

In [36]:
```
R"""
library(microbenchmark)
x=rnorm(1000000)
microbenchmark(sum(x))
"""
```

```
┌ Warning: RCall.jl: Warning: package 'microbenchmark' was built under R vers
ion 3.6.3
└ @ RCall C:\Users\Lenovo\.julia\packages\RCall\LWzAQ\src\io.jl:172
```

Out[36]:
```
RObject{VecSxp}
Unit: milliseconds
   expr    min      lq     mean  median      uq    max neval
 sum(x) 2.1797 2.18505 2.219479 2.18985 2.19625 2.6053   100
```

In [37]:
```
x1 = @rget x;
```

In [38]:
```
@time sum(x1)
```

```
  0.000717 seconds (1 allocation: 16 bytes)
```

Out[38]:
```
-865.0217002098822
```

0.914 mili seconds

In [46]:
```
2.18815/0.717
```

Out[46]:
```
3.051813110181311
```

In [ ]: