Statistics WS 23/24

by Max Semdner

Table of contents

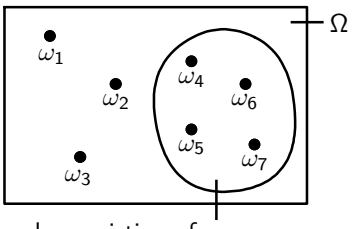
Descriptive Statistics

1 Observation, Population, Sample

Observation: When collecting data, each individual piece of data is called an *observation*. Observations (plural) refers to the collection of all data points. It's often denoted with ω.

Population: Collection of all possible observations. It's usually denoted with Ω.

Sample: Selection of observations $\omega_1, \ldots, \omega_n$, which is a subset of the population Ω.



Sample consisting of $\omega_4, \omega_5, \omega_6, \omega_7$

2 Variables

Variables: A variable X represents a specific feature of an observation. Multiple characteristics are each captured in a separate variable $X_i, 1, \ldots, p$. These variables can be collectively represented as $X = (X_1, X_2, \ldots, X_p)$, which is a way of grouping all the features together. For each observation made, these variables then take specific, measurable values, and these are denoted as $(x_1, x_2, \ldots, x_p)$, corresponding the values of $X_1, X_2, \ldots, X_3$.

Example: You want to study the health of a group of people. Each observation is one person. You are interested in the features weight (X_1), age (X_2), blood pressure (X_3). For each observation you consider $X = (X_1, X_2, X_3)$. The observed perrson ω_1 may have the values $(75\,\text{kg}, 26, 120/80)$, the observed person ω_2 may the values $(100\,\text{kg}, 51, 130/85)$, ...

Variable Types: Variables can be of two different types, either *qualitative or quantitative*, or *discrete or continous*.

Qualitative: Variables which values take values that cannot be orded in a logical or natural way (example: eye color, political party, ...)

Quantitative: Are measured in terms of numbers. They can be ordered in a logical or natural way (example: size of shoes, prices, ...).

$\Rightarrow$ In some cases qualitative variables can be numbers too, but they still can't be ordered (example: Matrikel Number, Male = 0 and Female = 1).

Discrete: They can take only a finite number of values (example: eye color, country, ...).

Continous: They can take on any value in a certain range (example: height, weight, ...).

$\Rightarrow$ All qualitative variables are discrete. Quantitative variables can be either discrete or continous.

3 Scale

Scale: Different variables can contain different amount of information, which is considered the scale. There are 5 types of scale: *nominal-, ordinal-,* $\underbrace{\textit{interval-, ratio-, absolute}}_{\textit{metric scale}}$ *scale*.

Nominal Scale: Values that cannot be ordered (example: male-female, eye color, ...).

Ordinal Scale: Values that can be ordered, but the difference between the values cannot be interpreted in a meaningful way (example: education level, star ratings on products, ...).

Metric Scale: Values that can be ordred and the differences between the values can be interpreted in a meaningful way (example: height of a person, temperature, ...). The metric scale can be divided in to three sub-scales, interval-, ratio and absolute scale. Therefore, we use the three subscales and not simply the general metric scale.

> **Interval Scale:** The difference between the values can be interpreted, but not the ratio (example: The difference between 2°C and 6°C is 4°C but, it doesnt mean 6°C is 3 times as cold.)
>
> **Ratio Scale:** The difference and ratio between the values can be interpreted, because we have an identifiable absolute zero point.
>
> **Absolute Scale:** The same as ratio scale but we don't have "natural" units such as km/h, °C. Instead the values are simply 1, 2, 3, ... (example: Number of semester studied).

4 Tidying Data

Why is the dataset not tidy?

Rules of a tidy dataset: variables are in columns, observations are in rows, and values are in cells.

1. Each variable forms a column.

2. Each observation forms a row.

3. Each type of observational unit forms a table.

Typical problems: One variable might be spread across multiple columns. One observation might be scattered across multiple rows

gather()

gather() is used to combine multiple columns in one key-value pair column.

col1	var1	var2	var3
...	5	2	7
...	2	12	3

→

col1	vars	vals
...	var1	5
...	var2	2
...	var3	7
...	var1	2
...	var2	12
...	var3	3

R-Code

```
gather(data = tibble_name,
         key = ‘‘vars’’,
       value = ‘‘vals’’,
       var1, var2, var3
)
```

spread()

col1	col2	col3
X	val1	1
X	val2	5.0
Y	val1	3
Y	val2	2.4

→

col1	val1	val2
X	1	5.0
Y	3	2.4

R-Code

1	se
2	
3	
4	
5	

seperate()

col1	col2
...	1/5.0
...	7/2.3

→

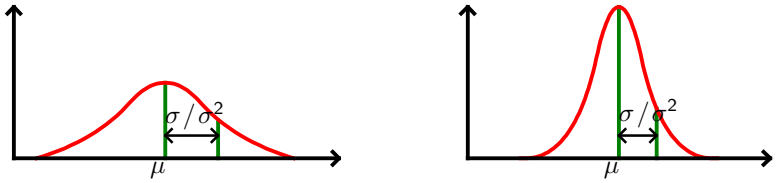
col1	new_col_2	new_col_3
...	1	5.0
...	7	2.3

R-Code

```
seperate(data = tibble_name,
           col = ‘‘col2’’,
           into = c(‘‘new_col2’’, ‘‘new_col3’’), # values in col2 are split into new_col_2 and new_col_3
           sep = ’’/’’,
       convert = TRUE
)
```

5 Standard Deviation and Variance

Standard Deviation and Variance describe how close the scores are to the middle of the distribution. *(Note: The Standard Deviation and Variance is calculated slightly different depending if we consider the entire population or just a sample)*



Standard Deviation (s/σ): Describes how dispersed the data is in relation to the sample mean ($\bar{x}$). Low, or small, standard deviation indicates data are clustered tightly around the mean, and high, or large, standard deviation indicates data are more spread out.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad n = \text{sample size}, \bar{x} = \text{sample mean}$$

R-Code
```
1 sd(x=c(...)) # c() is the vector with the sample data
```

Variance (s^2/σ^2):

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad n = \text{sample size}, \bar{x} = \text{sample mean}$$

R-Code
```
1 var(x=c(...)) # c() is the vector with the sample data
```

Bivariate Data:

Covariance (s_{xy}): The covariance tells in which direction the regression is going (0 means no linear relationship)

$$s_{xy} = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i \cdot y_i - n \cdot \bar{x} \cdot \bar{y}\right), \quad n = \text{sample size}$$

R-Code
```
1 cov(x=..., y=...)
```

Coefficient of Correlation (r_{xy}): The coefficient of correlation also tells us the strength of their relation and is unit independent.

$$r_{xy} = \frac{s_{xy}}{s_x \cdot s_y}$$

R-Code
```
1 cor(x=..., y=...)
```

Coefficient of Determination: Coefficient of Correlation squared. How much % of the data can be explained by the regression line.

R-Code
```
1 cor(x=..., y=...)^2 # result is how muchd data can be explained by regression line
```

6 Plots

Box-Plots

R-Code

```
boxplot(y ~ x,     # y = data to be plotted, x = for each different x value a plot
        main = ''title'',
        ylab = ''y-axis description'',
       names = c(''name1'', ''name2'')
)
```

or

R-Code

```
boxplot(y1, y2     # plots two boxplots
        main = ''title'',
        ylab = ''y-axis description'',
       names = c(''name1'', ''name2'')
)
```

Example: Interpretation Boxplot

Thick Line: The Line is the median of the data. Since the median of the left boxplot is lower it may suggest that the treatment may was effective.

Box (IQR): The Box represents where the middle 50% of the data lies. The left box is slightly higher indicating more variability in the pain rating.

Scatterplot and Linear Regression

R-Code

```
plot(data$x, data$y     # here data is a tibble
    main = ''title'',
    xlab = ''x-axis label'', ylab = ''y-axis label'',
)
reg <- lm(data$y ~ data$x) # here data is a tibble -- important always y ~ x
abline(a = reg$coefficient[1], b = reg$coefficient[2], col = "red")
```

7 Important R-Functions and More

Logical Operations: AND: &, OR: |

Generate Sequence

R-Code

```r
seq(from = 1, to = 10, by = 1) # 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Generate Random Values of Specific Length

R-Code

```r
sample(x = c("f", "m"), size = 10, replace = TRUE)  # 10 random values either f or m
sample(x = 20:35, size = 10, replace = TRUE)        # 10 random values between 20 and 35
```

Switch Case

R-Code

```r
tbl_name %>% rowwise()
sample(x = 20:35, size = 10, replace = TRUE)        # 10 random values between 20 and 35
```

Create Tibble

R-Code

```r
tbl_name <- tibble(
        id = ...,
        age = ...,
        sex = ...,
        score = ...
)
```

Get Tibble Length

R-Code

```r
length(tbl_name$id) # returns length of any column and thus the tibble length
```

Get Column Names

R-Code

```r
names(tbl_name)
```

Filter Column for a Value

R-Code

```r
tbl_name %>% filter(sex = "m")
tbl_name %>% filter(sex = "m", grade == c(1, 4, 6)) # or
tbl_name %>% filter(sex = "m" & grade == c(1, 4, 6))
```

Select Specific Columns

R-Code

```r
tbl_name %>% select(id, age)
```

Add One or More Columns to Tibble

R-Code

```r
tbl_name %>% mutate(new_col_1 = ...,
                    new_col_2 = ...
)
```

Add One or More Columns with New Values From Rows

R-Code

```
1 tbl_name %>% rowwise() %>% mutate(id_plus_age = id+age)
```

Before:

id	age	gender
1	3	m
2	4	f

After:

id	age	gender	id_plus_age
1	3	m	1+3
2	4	f	2+4

Add Column Based on Criteria (Switch-Case/Case-When)

R-Code

```
1 tbl_name %>% mutate(grade = case_when(
2     score >= 90 ~ 1,
3     score >= 70 ~ 2,
4     score >= 50 ~ 3,
5     score >= 30 ~ 4,
6     score >= 10 ~ 5,
7     score >= 0  ~ 6,
8 ))
```

Order/Arrange Values Descending/Ascending

Default: Ascending

R-Code

```
1 tbl_name %>% arrange(grade)        # 1, 2, 3, 4, 5, 6
2 tbl_name %>% arrange(desc(grade))  # 6, 5, 4, 3, 2, 1
3 tbl_name %>% arrange(sex)          # f, m (Alphabetically)
4 tbl_name %>% arrange(desc(sex))    # m, f
```

Select and Arrange

R-Code

```
1 tbl_name %>% select(sex, grade) %>% arrange(sex)
```

Summarise Values

Summaries creates a new datagrame. It's often used together with `mean()`, `median()`, `min()`, `max()`, `sd()`, ...

R-Code

```
1 tbl_name %>% group_by(sex) %>% summarise(grade_mean = mean(grade)) %>% ungroup()
```

Get Only Unique Values

R-Code

```
1 library(nycflights13)
2 flights %>% filter(carrier == c("UA", "AA", "DL") & month == 5) %>%
3             select(carrier, flight) %>%
4             arrange(carrier, desc(flight)) %>%
5             unique()
```

Print Top 10 Values

R-Code

```
1 tbl_name %>% top_n(n = 10)
```

8 Independence and Dependence

Independence: Two events are independent if the occurence of one event does not effect the probability of the occurrence of the other event.

$P(A \cap B) = P(A) \cdot P(B)$

Dependent: Two events are dependent if the occurence of one event does affect the probability of the occurrence of the other event (in some cases conditional probabilities are used to express dependence).

$P(A \cap B) \neq P(A) \cdot P(B)$

9 Random Experiment, Sample Space, Events, Probability Function and Probability Space

Random Experiment: An experiment that can be repeated any number of times, but the outcome cannot be predicted with certainty, until the completion of the experiment (example: tossing a coin, rolling a die, ...).

Sample Space: In probability the sample space Ω is the set of all possible outcomes in a random experiment (example: tossing two coins coin with 1 for head and 0 for tail $\rightarrow \Omega = \{(0,0),(0,1),(1,0),(1,1)\}$).

Events: Subsets of the sample space of an experiment. It can be said an event is something that either happens or doesn't happen. (example: tossing two coins with 1 for head and 0 for tail. Event $A = \{(1,1)\}$ occurs when both coins show head).

Probability Function: A probability function assigns probabilities to events (example: tossing two coins with 1 for head and 0 for tail. Event $A = \{(1,1)\}$ occurs when both coins show head, the probability function maps to A, the probability of the event to occur, which is $P(A) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$).

Probability Space: A probability space is a triple (Ω, A, P) consisting of the sample space Ω, which is a set of all possible outcomes, An event space, which is a set of all events we want to consider and the probability function P, which assigns each event in the event space it's probability (a number between 0 and 1).

10 Law of Large Numbers:

The Law of Large Numbers states that as the number of trials in an experiment increases ("mit zunehmender Anzahl von Versuchen in einem Experiment"), the average of the results obtained from these trials comes closer to the expected value. In other words, the more often an experiment is repeated, the closer the observed frequency of an event will approach its true probability.

11 Basic Rules

$P(A^c) = 1-P(A)$		The event A does not occur
$P(A \setminus B) = P(A)-P(A\cap B)$		The event A occurs but not B occurs
$P(A\cup B) = P(A)+P(B)-P(A\cap B)$		One or both of the events A and B occur
$P(A\cup B\cup C) = P(A)+P(B)+P(C)-$ $P(A\cap B)-P(A\cap C)-P(B\cap C)+$ $P(A\cap B\cap C)$		One, two or all (at least one) of the events occur
$P((A\cup B\cup C))^c = 1-P(A\cup B\cup C)$		None of the the three events occur
$P((A\cap B^c\cap C^c)\cup P(A^c\cap B\cap C^c)\cup P(A^c\cap B^c\cap C)) =$ $P(A)+P(A\cap B)+P(A\cap C)+$ $P(B)+P(A\cap B)+P(B\cap C)+$ $P(C)+P(A\cap C)+P(B\cap C)+3\cdot P(A\cap B\cap C)$		Exactly one of the three events occur
$P((A\cap B\cap C^c)\cup(A\cap C\cap B^c)\cup(B\cap C\cap A^c)) =$ $P(A\cap B)-P(A\cap B\cap C)+$ $P(A\cap C)-P(A\cap B\cap C)+$ $P(B\cap C)-P(A\cap B\cap C)$		Exactly two of the three events occur
$P(A\cap B) =$		Both the events A and B occur
$P(A^c\cup B^c) = P((A\cap B)^c)=1-P(A\cap B)$		Either event A does not occur, or event B does not occur, or both events do not occur
$P(A^c\cap B^c) = P((A\cup B)^c)=1-P(A\cup B)$		Neither event A nor event B occur
$P(A^c\cap B) = P(B\setminus(A\cap B))=P(B)-P(A\cap B)$		Event B happening without A happening at the same time
$P(A\cap B^c) = P(A\setminus(A\cap B))=P(A)-P(A\cap B)$		Event A happening without B happening at the same time
$P(A\cup B^c) = 1-(P(B)-P(A\cap B))$		Event A happening, B not happening, or both not happening.
$A\subset B \Rightarrow P(A)\leq P(B)$		If A is a subset of B, then the probability of event A is happening is less or equal to the probability of event B
Condtional Probabilites		
$P(B\|A) = \frac{P(A\cap B)}{P(A)}$		Event B occurs, under the condition that event A occured before
$\mathrm{P(A\|B)} = \frac{P(A\cap B)}{P(B)}$		Event A occurs, under the condition that event B occured before
$P(A^c\|B) = \frac{P(A^c\cap B)}{P(B)}$ $= \frac{P(B\setminus P(A\cap B))}{P(B)}=\frac{P(B)-P(A\cap B)}{P(B)}$		Event A does not occur under the condition that event B occured before
$P(A\|B^c) = \frac{P(A\cap B^c)}{P(B^c)}$ $= \frac{P(A\setminus P(A\cap B))}{1-P(B)}=\frac{P(A)-P(A\cap B)}{1-P(B)}$		Event A does occur under the condition that event B did not occured before
$P(A^c\|B^c) = \frac{P(A^c\cap B^c)}{P(B^c)}$ $= \frac{P((A\cup B)^c)}{1-P(B)}=\frac{1-P(A\cup B)}{1-P(B)}$		Event A does not occur under the condition that event B did not occur before

12 Conditional Probabilities

Conditional Probability means, that an event A occurs with the condition that another event B occured before:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Condition Identifiers: There are certain indicators, to identify what the condition is, in an exercise: who, with, given that, if ... then, suppose that, assuming that, on the condition that, in the case where, with the information that, ...

Multiplication Rule: The rule can be used if only $P(A|B)$ and $P(B)$ is known, to calculate $P(A \cap B)$. It's just $P(A|B) = \frac{P(A \cap B)}{P(B)}$ but rearranged.

$$P(A \cap B) = P(A|B)P(B)$$
$$or/\Leftrightarrow$$
$$P(B \cap A) = P(B|A)P(A)$$

Law of Total Probability: The total probability of can outcome B that can be realized via serveral distinct events A_i.

$$P(B) = \sum_n P(B \cap A_n) = \sum_n P(B|A_n)P(A_n) = P(B|A_1)P(A_1) + \cdots + P(B|A_n)P(A_n)$$

Bayes' Rule: Calculate the individual probabilities A_k.

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(B)} = \frac{P(B|A_k)P(A_k)}{\sum_n P(B|A_n)P(A_n)}$$

(In)Dependence: When the occurence of one event B influences the probability of occurence of another event A they are dependent. If not, they are independent.

- If $P(A \cap B) = P(A)P(B)$ then A and B are independent, if not they are dependent.

Positive/Negative Correlated: If A and B are dependent we check how they are correlated. If they are positively correlated, the occurence of one event, increased the probability of occurrence of the other event. If they are negatively correlated, the occurence of one event, decreased the probability of occurrence of the other event.

- If $P(A|B) > P(B)$ then A and B are postively correlated

- If $P(A|B) < P(B)$ then A and B are negatively correlated

13 Random Variables and Distributions

Random Variables

Random Variables: A Variable (actually a function) maps outcomes to any value (outcome $\rightarrow$ value). This is used to shorten/simplify the expression of probabilities. Instead of explicitly writing out the probability of each possible result, the values of the random variables can simply be used (example: rolling two fair dies. If we want to know the probability of an outcome we would usually write $P(\text{sum of dies is} \leq 4)$ or $P(\text{sum of dies is} = 8)$. When using a random variable X, we could assign values to the outcomes. So X would for example represent all the possible sums: $X = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. If we want to know the probability of an outcome now, we can just write $P(X \leq 4)$ or $P(X = 8)$).

Types: *Discrete Random Variables* and *Continous Random Variables*.

(In)Dependence: The random variables X, Y are called independent, if $P(X \leq x, Y \leq y) = P(X \leq x) \cdot P(Y \leq y)$, for all $x, y \in R$

Discrete Random Variables

Discrete Random Variables: Can take on countable values (example: $0, 1, 2, \ldots, 10$).

Expectation: $E(X) = \sum_{\text{all } x} x \cdot P(X = x)$ (Expectation reflect the arithmetic mean of the distribution of the population)

Variance: $\text{Var}(X) = E([X - E(X)]^2) = E(X^2) - (E(X))^2$

Squared Deviation: $\sqrt{\text{Var}(X)}$

Continous Random Variables

Continous Random Variables: Can take on a range of values (example: temperature).

Distributions

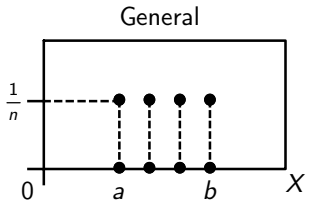
Distributions: Distributions describe the way in which the values of set of a random variable are spread. Since there are two types of random variables (discrete and continous), there are also two types of distributions (discrete and continous). Within both discrete and continuous distributions, there are various types. The way how the distributions are described differs: Discrete distributions are described using a *Discrete Probability Density Function* (Also called Probability Mass Function). Continous distributions are described using a *Probability Density Function*. The (Discrete) Probability Density Function of an Distribution depends on the type.

Discrete Distributions

Types of Discrete Distributions: Discrete Uniform-, Binomial-, Hypergeometric-, Multinomial-, Generalised Hypergeometric-, Geometric Distribution

Discrete Probability Density Function: The discrete probability density function describes the probabilities associated with each possible outcome of a discrete random variable (If we say want to find the distribution or the density of a random variable we need to find the discrete probability density function).

Discrete Uniform Distribution: An element is chosen at random from a infinite set S. All outcomes are *equally likely*: $f(x) = \frac{1}{|S|}$ (example: tossing a fair coin or rolling a fair die).



Binomial- and Hypergeometric Distribution: If there are only two types of outcomes (e.g success (1) or failure (0), rolling a 5 or not rolling a 5, drawing a red ball in an urn or not a red ball) and from one possible outcome we want to know the probability if it occurs/is drawn k times, it's always either a binomial distribution or a hypergeometric. It's a binomial if the experiment is done with replacement and hypergeometric if it's without replacement.

Binomial Distribution (with replacement): Each trial is independent of others, meaning the outcome of one trial does not affect the outcome of another. The experiment is with replacement, which means that after each trial, the selected item (success or failure) is put back into the population, and the population remains unchanged (and thus it's proability to occur).

$n =$ Number of Elements drawn in total/Number of times Experiment repeated/Trails, $k =$ Number of times one outcome occurs

$$P(k \text{ successes in } n \text{ trails}) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

R-Code

```
dbinom(x=k, size=n, prob=p)
# or
choose(n,k)*p^k*(1-p)^(n-k)
```

Hypergeometric Distribution (without replacement): Each trial are dependent because the sampling is done without replacement. This means that after each trial, the selected item is not returned to the population (and thus it's probability to occur changes).

$N =$ Total Number of Elements in Population, $M =$ Number of Elements of one Type/Outcome,

$n =$ Number of Elements drawn in total/Number of times Experiment repeated, $k =$ Number of Elements drawn/occur from M

$$P(k \text{ successes in } n \text{ traills}) = P(X = k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

R-Code

```
dhyper(x=k, m=M, n=N-M, k=n)
# or
(choose(M,k)*choose(N-M,n-k))/choose(N,n)
```

Multinomial- and Generalised Hypergeometric Distribution: Here there are more than two outcomes. Both distributions are used to determine the probability of each outcome occurring a specific number of times in a given number of trials. The Multinomial is an extension of the binomial distribution, so it's with replacement and the generalised hypergeometric distribution is without replacement.

Multinomial Distribution (with replacement):

$n =$ Total Number of Elements, $p_1, p_2, \dots p_k =$ Probabilities for each outcome, $x_1, x_2, \dots, x_k =$ Number of Times each Outcome occur

$$P(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) = \frac{n!}{x_1! \cdot x_2! \cdot \dots \cdot x_k!} \cdot p_1^{x_1} \cdot p_2^{x_2} \cdot \dots \cdot p_k^{x_k}$$

R-Code

```
dmultinom(x=c(x1,x2,...,xk), prob=c(p1,p2,...,pk))
```

Generalised Hypergeometric Distribution (without replacement):

$n =$ Total Number of Elements, $k =$ Number of Elements drawn/Trails,

$m_1, m_2, \dots, m_k =$ Number of Elements for each Outcome, $x_1, x_2, \dots, x_k =$ Number of Times each Outcome occurs

$$P(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) = \frac{\binom{m_1}{x_1} \cdot \binom{m_2}{x_2} \cdot \dots \cdot \binom{m_k}{x_k}}{\binom{n}{k}}$$

R-Code

```
(choose(m1,x1)*choose(m2,x2)*...*choose(mk,xk))/choose(n,k)
```

Geometric Distribution (with replacement): At a specific trail or drawing a event is occuring. The probability of occurence for the event doesn't change, with the trails.

$k =$ The trail when the event first occurs, $p =$ Probability that the event occurs

$$P(X = k) = (1-p)^{k-1} \cdot p$$

R-Code

```
dgeom(x=k-1, prob=p) # x = number of failures before occurence,
# or
(1-p)^(k-1)*p
```

(Without replacement): At a specific trail or drawing a event is occuring. The probability of occurence for the event does change, with the trails. (*NOTE: The name of this distribution wasn't given in the lecture!*)

$$P(X = k) = \frac{\binom{K}{k-1} \cdot \binom{N-K}{0}}{\binom{N}{k-1}} \cdot \frac{N-K}{K-(N-1)}$$

Continous Distributions

Types of Continous Distributions:

Uniform Distribution:

Normal Distribution $N(\mu, \sigma)$: A symmetric, bell shaped Described by the two parameters

Probability Density Function $f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Maximum: $x = \mu$

Symmetric: $f(\mu + x) = f(\mu - x)$

Inflection Points: $\mu \pm \sigma$

$E(X) = \mu, \text{Var}(X) = \sigma^2$

Standard Normal Distribution $N(\mu = 0, \sigma = 1)$: A Normal Distribution where $\mu = 0$ and $\sigma = 1$.

Probability Density Function $\Phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

Inferential Statistics

14 Estimators

Mean

If we don't know the actual value μ (mean) of an population, we can calculate an estimation of this value based on a random sample $(X_1, X_2, \ldots, X_n)$. This estimation is denoted with $\bar{X}$ (sample mean).

$$\bar{X}_{(n)}(X) = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad n = \text{sample size}$$

Variance

μ is known: If μ is known we used μ to calculate the estimation of the variance

$$W_n^2(X) = \frac{1}{n}\sum_{i=1}^{n} (X_i - \mu)^2$$

μ is unkown: If we don't know μ we use the estimated value $\bar{X}$ of μ.

$$S_n^2(X) = \frac{1}{n-1}\sum_{i=1}^{n} (X_i - \bar{X}_{(n)}(X))^2$$

15 Confidence Intervals

Confidence Interval for μ: Standard Deviation (σ) or Variance (σ^2) known

$$\left[\bar{X}_{(n)} - u_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}, \bar{X}_{(n)} + u_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}\right]$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
x_mean <- mean(x) # mean of samples
sigma <- σ/sqrt(σ^2)   # standard deviation given. If only the variance is given then use sqrt(sigma)
alpha <- 1-(1-α) # 1-α is often 0.95/0.99/..., thus 1-(1-α)/α then 0.05/0.01/...
u <- qnorm(1-alpha/2, mean = 0, sd = 1)
lb <- x_mean-u*(sigma/sqrt(n))
ub <- x_mean+u*(sigma/sqrt(n))
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$$n = \left(\frac{-u_{1-\alpha/2} \cdot \sigma}{\text{lb} - \bar{X}_{(n)}}\right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
sigma <- ...    # given
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
n <- ((-u*sigma)/(lb-x_mean))^2
```

Upper Bound (ub)

$$n = \left(\frac{u_{1-\alpha/2} \cdot \sigma}{\text{ub} - \bar{X}_{(n)}}\right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
sigma <- ...    # given
ub <- ...       # given
x_mean <- ...   # given or calculated
n <- ((u*sigma)/(ub-x_mean))^2
```

$$\bar{X}_{(n)} = \text{lb} + u_{1-\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
sigma <- ...    # given
lb <- ...       # given
n <- ...        # given/calculated from sample
x_mean <- lb+u*sigma/sqrt(n)
```

$$\bar{X}_{(n)} = \text{ub} - u_{1-\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
sigma <- ...    # given
ub <- ...       # given
n <- ...        # given/calculated from sample
x_mean <- ub-u*sigma/sqrt(n)
```

$$\sigma = \frac{(-\text{lb} + \bar{X}_{(n)}) \cdot \sqrt{n}}{u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...        # given/calculated from sample
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
sigma <- ((-lb+x_mean)*sqrt(n))/(u)
```

$$\sigma = \frac{(\text{ub} - \bar{X}_{(n)}) \cdot \sqrt{n}}{u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...        # given/calculated from sample
ub <- ...       # given
x_mean <- ...   # given/calculated from sample
sigma <- ((ub-x_mean)*sqrt(n))/(u)
```

$$u_{1-\alpha/2} = \frac{(-\text{lb} + \bar{X}_{(n)}) \cdot \sqrt{n}}{\sigma}$$

R-Code

```
sigma <-...     # given
n <- ...        # given/calculated from sample
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
u <- ((-lb+x_mean)*sqrt(n))/(sigma)
# conf level: 1-(2*(1-pnorm(u)))
#       alpha: (2*(1-pnorm(u)))
```

$$u_{1-\alpha/2} = \frac{(\text{ub} - \bar{X}_{(n)}) \cdot \sqrt{n}}{\sigma}$$

R-Code

```
sigma <-...     # given
n <- ...        # given/calculated from sample
ub <- ...       # given
x_mean <- ...   # given/calculated from sample
u <- ((-lb+x_mean)*sqrt(n))/(sigma)
# conf level: 1-(2*(1-pnorm(u)))
#       alpha: (2*(1-pnorm(u)))
```

Length of the Interval: $\text{length} = \text{ub} - \text{lb}$

$$n = \left(\frac{2 \cdot u_{1-\alpha/2} \cdot \sigma}{\text{length}} \right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
sigma <- ...        # given
length <- ...       # given
n <- (2*u*sigma/length)^2
```

$$\sigma = \frac{\text{length} \cdot \sqrt{n}}{2 \cdot u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...        # given/calculated from sample
length <- ...   # given
sigma <- (length*sqrt(n)/(2*u))
```

$$u_{1-\alpha/2} = \frac{\text{length} \cdot \sqrt{n}}{2 \cdot \sigma}$$

R-Code

```
n <- ...        # given/calculated from sample
sigma <-...     # given
length <- ...   # given
u <- (length*sqrt(n))/(2*sigma)
```

Mean from bounds: $\text{mean} = \frac{\text{lb} + \text{ub}}{2}$

R-Code

```
x_mean <- (lb+ub)/2
```

Confidence Interval for μ: Standard Deviation (σ) unknown

$$\left[\bar{X} - t_{n-1;1-\frac{\alpha}{2}} \cdot \frac{S_{(n)}}{\sqrt{n}}, \bar{X} + t_{n-1;1-\frac{\alpha}{2}} \cdot \frac{S_{(n)}}{\sqrt{n}}\right]$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
x_mean <- mean(x) # mean of samples
s <- sd(x)        # calculate standard deviation based on samples
alpha <- 1-(1-α) # 1-α is often 0.95/0.99/..., thus 1-(1-α)/α then 0.05/0.01/...
t <- qt(1-alpha/2, n-1)
lb <- x_mean-t*(s/sqrt(n))
ub <- x_mean+t*(s/sqrt(n))
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$$n = \left(\frac{-u_{1-\alpha/2} \cdot S_{(n)}}{\text{lb} - \bar{X}_{(n)}}\right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
x_sd <- ...     # given/calculated from sample
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
n <- ((-u*x_sd)/(lb-x_mean))^2
```

Upper Bound (ub)

$$n = \left(\frac{u_{1-\alpha/2} \cdot S_{(n)}}{\text{ub} - \bar{X}_{(n)}}\right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
x_sd <- ...     # given/calculated from sample
ub <- ...       # given
x_mean <- ...   # given or calculated
n <- ((u*x_sd)/(ub-x_mean))^2
```

$$\bar{X}_{(n)} = \text{lb} + u_{1-\alpha/2} \cdot \frac{S_{(n)}}{\sqrt{n}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
x_sd <- ...     # given/calculated from sample
lb <- ...       # given
n <- ...        # given/calculated from sample
x_mean <- lb+u*sigma/sqrt(n)
```

$$\bar{X}_{(n)} = \text{ub} - u_{1-\alpha/2} \cdot \frac{S_{(n)}}{\sqrt{n}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
x_sd <- ...     # given/calculated from sample
ub <- ...       # given
n <- ...        # given/calculated from sample
x_mean <- ub-u*sigma/sqrt(n)
```

$$S_{(n)} = \frac{(-\text{lb} + \bar{X}_{(n)}) \cdot \sqrt{n}}{u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...        # given/calculated from sample
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
x_sd <- ((-lb+x_mean)*sqrt(n))/(u)
```

$$S_{(n)} = \frac{(\text{ub} - \bar{X}_{(n)}) \cdot \sqrt{n}}{u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...        # given/calculated from sample
ub <- ...       # given
x_mean <- ...   # given/calculated from sample
x_sd <- ((ub-x_mean)*sqrt(n))/(u)
```

$$u_{1-\alpha/2} = \frac{(-\text{lb} + \bar{X}_{(n)}) \cdot \sqrt{n}}{S_{(n)}}$$

R-Code

```
x_sd <- ...     # given/calculated from sample
n <- ...        # given/calculated from sample
lb <- ...       # given
x_mean <- ...   # given/calculated from sample
u <- ((-lb+x_mean)*sqrt(n))/(sigma)
# conf level: (1-(2*(1-pt(u)))
#       alpha: (2*(1-pt(u))
```

$$u_{1-\alpha/2} = \frac{(\text{ub} - \bar{X}_{(n)}) \cdot \sqrt{n}}{S_{(n)}}$$

R-Code

```
x_sd <- ...     # given/calculated from sample
n <- ...        # given/calculated from sample
ub <- ...       # given
x_mean <- ...   # given/calculated from sample
sigma <- ((ub+x_mean)*sqrt(n))/(u)
# conf level: (1-(2*(1-pt(u)))
#       alpha: (2*(1-pt(u))
```

Length of the Interval: $\text{length} = \text{ub} - \text{lb}$

$$n = \left(\frac{2 \cdot u_{1-\alpha/2} \cdot S_{(n)}}{\text{length}} \right)^2$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
x_sd <- ...          # given/calculated from sample
length <- ...        # given
n <- (2*u*sigma/length)^2
```

$$S_{(n)} = \frac{\text{length} \cdot \sqrt{n}}{2 \cdot u_{1-\alpha/2}}$$

R-Code

```
u <- qnorm(1-alpha/2, 0, 1)
n <- ...          # given/calculated from sample
length <- ...     # given
x_sd <- (length*sqrt(n)/(2*u))
```

$$u_{1-\alpha/2} = \frac{2 \cdot \text{length} \cdot \sqrt{n}}{S_{(n)}}$$

R-Code

```
n <- ...          # given/calculated from sample
x_sd <- ...       # given/calculated from sample
length <- ...     # given
u <- (2*length*sqrt(n))/sigma
```

Mean from bounds: $\text{mean} = \frac{\text{lb} + \text{ub}}{2}$

R-Code

```
x_mean <- (lb+ub)/2
```

Confidence Interval for Standard Deviation σ: Mean μ_0 known

$$\left[\sqrt{\frac{Q_{(n)}}{\chi^2_{n;1-\frac{\alpha}{2}}}}, \sqrt{\frac{Q_{(n)}}{\chi^2_{n;\frac{\alpha}{2}}}}\right] \text{and with } Q_{(n)} = \sum_{i=1}^{n} (X_i - \mu_0)^2$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
mu <- μ0          # μ0 is given
alpha <- 1-(1-α) # 1-α is often 0.95/0.99/..., thus 1-(1-α)/α then 0.05/0.01/...
Qn <- sum((x-mu)^2)
chi1 <- qchisq(alpha/2,n)
chi2 <- qchisq(1-alpha/2,n)
lb <- sqrt(Qn/chi2)
ub <- sqrt(Qn/chi1)
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$\chi^2_{n;1-\alpha/2} = \frac{Q_{(n)}}{\text{lb}}$

R-Code

```
Qn <- sum((x-mu)^2)  # given/calculated
lb <- ...            # given
chi.sq <- (Qn/lb)
```

$Q_{(n)} = \text{lb}^2 \cdot \chi^2_{n;1-\alpha/2}$

R-Code

```
chi.sq <- qchisq(1-alpha/2,n) # given/calculated
lb <- ... # given
Qn <- chi.sq*lb
```

Upper Bound (ub)

$\chi^2_{n;\alpha/2} = \frac{Q_{(n)}}{\text{ub}^2}$

R-Code

```
Qn <- sum((x-mu)^2)  # given/calculated
ub <- ...            # given
chi.sq <- (Qn/ub)
```

$Q_{(n)} = \text{ub}^2 \cdot \chi^2_{n;\alpha/2}$

R-Code

```
chi.sq <- qchisq(alpha/2,n) # given/calculated
ub <- ... # given
Qn <- chi.sq*ub
```

Length of the Interval: $\text{length} = \text{ub} - \text{lb}$

Mean from bounds: $\text{mean} = \frac{\text{lb} + \text{ub}}{2}$

R-Code

```
x_mean <- (lb+ub)/2
```

Confidence Interval for Standard Deviation σ: Mean unkown

$$\left[\sqrt{\frac{(n-1)S^2_{(n)}}{\chi^2_{n-1;1-\frac{\alpha}{2}}}}, \sqrt{\frac{(n-1)S^2_{(n)}}{\chi^2_{n-1;\frac{\alpha}{2}}}}\right]$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
s <- sd(x)        # Since μ0 is not given the standard deviation needs be calculated from the sample
alpha <- 1-(1−α) # 1−α is often 0.95/0.99/..., thus 1−(1−α)/α then 0.05/0.01/...
chi1 <- qchisq(alpha/2,  n-1)
chi2 <- qchisq(1-alpha/2,n-1)
lb <- sqrt(((n-1)*s^2)/chi2)
ub <- sqrt(((n-1)*s^2)/chi1)
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$\chi^2_{n;1-\alpha/2} = \frac{(n-1)\cdot S^2}{\text{lb}^2}$

R-Code

```
n <- length(...)  # given/calculated
s <- sd(...)      # given/calculated
lb <- ...         # given
chi.sq <- ((n-1)*s^2)/lb^2
```

Upper Bound (ub)

$\chi^2_{n;\alpha/2} = \frac{(n-1)\cdot S^2}{\text{ub}^2}$

R-Code

```
n <- length(...)  # given/calculated
s <- sd(...)      # given/calculated
ub <- ...         # given
chi.sq <- ((n-1)*s^2)/ub^2
```

$n = \frac{\text{lb}^2 + \chi^2_{n;1-\alpha/2}}{S^2}$

R-Code

```
chi.sq <- ... # given
lb <- ...     # given
s <- ...      # given
n <- (lb^2+chi.sq)/s
```

$n = \frac{\text{ub}^2 + \chi^2_{n;\alpha/2}}{S^2}$

R-Code

```
chi.sq <- ... # given
ub <- ...     # given
s <- ...      # given
n <- (ub^2+chi.sq)/s
```

$S^2 = \frac{\text{lb}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)} / S = \sqrt{\frac{\text{lb}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)}}$

R-Code

```
chi.sq <- ... # given/calculated
lb <- ...     # given
s <- ...      # given
S.sq <- (lb^2+chi.sq)/(n-1)
S <- sqrt((lb^2+chi.sq)/(n-1))
```

$S^2 = \frac{\text{ub}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)} / S = \sqrt{\frac{\text{ub}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)}}$

R-Code

```
chi.sq <- ... # given/calculated
ub <- ...     # given
s <- ...      # given
S.sq <- (ub^2+chi.sq)/(n-1)
S <- sqrt((ub^2+chi.sq)/(n-1))
```

Confidence Interval for Variance σ^2: Mean μ_0 known

$$\left[\frac{Q_{(n)}}{\chi^2_{n;1-\frac{\alpha}{2}}}, \frac{Q_{(n)}}{\chi^2_{n;\frac{\alpha}{2}}}\right] \text{ with } Q_{(n)} = \sum_{i=1}^{n} (X_i - \mu_0)^2$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
mu <- μ0          # μ0 is given
alpha <- 1-(1−α) # 1−α is often 0.95/0.99/..., thus 1−(1−α)/α then 0.05/0.01/...
Qn <- sum((x-mu)^2)
chi1 <- qchisq(alpha/2,n)
chi2 <- qchisq(1-alpha/2,n)
lb <- Qn/chi2
ub <- Qn/chi1
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$\chi^2_{n;1-\alpha/2} = \frac{Q_{(n)}}{\text{lb}}$

R-Code

```
Qn <- sum((x-mu)^2)  # given/calculated
lb <-...             # given
chi.sq <- (Qn/lb)
```

$Q_{(n)} = \text{lb}^2 \cdot \chi^2_{n;1-\alpha/2}$

R-Code

```
chi.sq <- qchisq(1-alpha/2,n) # given/calculated
lb <- ... # given
Qn <- chi.sq*lb
```

Upper Bound (ub)

$\chi^2_{n;\alpha/2} = \frac{Q_{(n)}}{\text{ub}^2}$

R-Code

```
Qn <- sum((x-mu)^2)  # given/calculated
ub <-...             # given
chi.sq <- (Qn/ub)
```

$Q_{(n)} = \text{ub}^2 \cdot \chi^2_{n;\alpha/2}$

R-Code

```
chi.sq <- qchisq(alpha/2,n) # given/calculated
ub <- ... # given
Qn <- chi.sq*ub
```

Confidence Interval for Variance σ^2: Mean unkown

$$\left[\frac{(n-1)S^2_{(n)}}{\chi^2_{n-1;1-\frac{\alpha}{2}}}, \frac{(n-1)S^2_{(n)}}{\chi^2_{n-1;\frac{\alpha}{2}}}\right]$$

R-Code

```
x <- c(...)       # sample values
n <- length(x)    # number of samples
s <- sd(x)        # Since μ0 is not given the standard deviation is not given
alpha <- 1-(1-α) # 1-α is often 0.95/0.99/..., thus 1-(1-α)/α then 0.05/0.01/...
chi1 <- qchisq(alpha/2,  n-1)
chi2 <- qchisq(1-alpha/2,n-1)
lb <- ((n-1)*s^2)/chi2
ub <- ((n-1)*s^2)/chi1
cat("[", lb, ", ", up, "]")
```

Lower Bound (lb)

$$\chi^2_{n;1-\alpha/2} = \frac{(n-1)\cdot S^2}{\text{lb}}$$

R-Code

```
n <- length(...)  # given/calculated
s <- sd(...)      # given/calculated
lb <- ...         # given
chi.sq <- ((n-1)*s^2)/lb
```

Upper Bound (ub)

$$\chi^2_{n;\alpha/2} = \frac{(n-1)\cdot S^2}{\text{ub}}$$

R-Code

```
n <- length(...)  # given/calculated
s <- sd(...)      # given/calculated
ub <- ...         # given
chi.sq <- ((n-1)*s^2)/ub
```

$$n = \frac{\text{lb} + \chi^2_{n;1-\alpha/2}}{S^2}$$

R-Code

```
chi.sq <- ... # given
lb <- ...     # given
s <- ...      # given
n <- (lb+chi.sq)/s
```

$$n = \frac{\text{ub}^2 + \chi^2_{n;\alpha/2}}{S^2}$$

R-Code

```
chi.sq <- ... # given
ub <- ...     # given
s <- ...      # given
n <- (ub+chi.sq)/s
```

$$S^2 = \frac{\text{lb} + \chi^2_{n;1-\alpha/2}}{(n-1)} / S = \sqrt{\frac{\text{lb} + \chi^2_{n;1-\alpha/2}}{(n-1)}}$$

R-Code

```
chi.sq <- ... # given/calculated
lb <- ...     # given
s <- ...      # given
S.sq <- (lb^2+chi.sq)/(n-1)
S <- sqrt((lb^2+chi.sq)/(n-1))
```

$$S^2 = \frac{\text{ub}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)} / S = \sqrt{\frac{\text{ub}^2 + \chi^2_{n;1-\alpha/2}}{(n-1)}}$$

R-Code

```
chi.sq <- ... # given/calculated
ub <- ...     # given
s <- ...      # given
S.sq <- (ub^2+chi.sq)/(n-1)
S <- sqrt((ub^2+chi.sq)/(n-1))
```

16 Hypothesis Testing

Hint 1: If it's not specified wheter to do an approximiation or an exact calculation use the R-Function (exact calculation) in the exam because it's faster.

Hint 2: If no alpha/α is given, then just use $\alpha = 0.05$.

H_0, H_1, Type I Error, Type II Error and Significance Level

Null Hypothesis (H_0): Hypothesis indicating the absence of the effect you are testing.

Alternative Hypothesis (H_1): Hypothesis you are trying to provide evidence for.

Type I Error: H_0 is true but is rejected

Type II Error: H_0 is false but not rejected

	H_0 is true	H_0 is not true
H_0 is not rejected	Correct decision	Type II error
H_0 is rejected	Type I error	Correct decision

Significance Level (α): $P(H_1|H_0) = \alpha$ is the probability of rejecting H_0 (accepting H_1) even if H_0 is actually true (the α is the same as in the confidence intervals and defines the intervals. If the value specified by the null hypothesis is in the interval, H_0 is not rejected, if it's outside the interval H_0 can be rejected).

β: $P(H_0|H_1) = \beta$ is the probability of rejecting H_1 (accepting H_0) even if H_1 is actually true.

The smaller the risk of one type of error, the higher the risk of the other.

Hypothesis Testing Objective: Is there statistical evidence to reject the null hypothesis (H_0) in favor of the alternate hypothesis (H_1) (Hypothesis Test = Statistical Decision).

One- and Two-Tailed Tests

Types of Hypothesis Tests: *One-Tailed Tests* and *Two-Tailed Tests*

Null Hypothesis (H_0)	Alternative Hypothesis (H_1)	Type
$\theta \leq \theta_0$	$\theta > \theta_0$	One-Tailed Test
$\theta \geq \theta_0$	$\theta < \theta_0$	One-Tailed Test
$\theta = \theta_0$	$\theta \neq \theta_0$	Two-Tailed Test

16.1 One-Sample Hypothesis Tests

Normal Model

Add Code: z.test, t.test, sigma.test

Z/Gauß-Test: $N(\mu, \sigma_0^2)$ with μ unkown and σ_0 known

Test Statistic: $\frac{\bar{X}_{(n)} - \mu}{\sigma_0}\sqrt{n} \sim N(0,1)$

Decision Rule: $\frac{\bar{x} - \mu_0}{\sigma_0}\sqrt{n} \in R \Rightarrow$ reject H_0

H_0	Rejection Region R
$\mu = \mu_0$	$(-\infty, -u_{1-\frac{\alpha}{2}}) \cup (u_{1-\frac{\alpha}{2}}, \infty)$
$\mu \leq \mu_0$	$(u_{1-\alpha}, \infty)$
$\mu \geq \mu_0$	$(-\infty, -u_{1-\alpha})$

t-Test: $N(\mu, \sigma^2)$ with μ unkown and σ_0 unkown

Test Statistic: $\frac{\bar{X}_{(n)} - \mu}{s_{(n)}}\sqrt{n} \sim t_{n-1}$ with $S^2_{(n)} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_{(n)})^2$ which is the sample variance

Decision Rule: $\frac{\bar{x} - \mu_0}{s_{(n)}}\sqrt{n} \in R \Rightarrow$ reject H_0

H_0	Rejection Region R
$\mu = \mu_0$	$(-\infty, -t_{n-1,1-\frac{\alpha}{2}}) \cup (t_{n-1,1-\frac{\alpha}{2}}, \infty)$
$\mu \leq \mu_0$	$(t_{n-1,1-\alpha}, \infty)$
$\mu \geq \mu_0$	$(-\infty, -t_{n-1,1-\alpha})$

$N(\mu, \sigma^2)$ with μ and σ unkown

Test Statistic: $\frac{(n-1) - S^2_{(n)}}{\sigma^2} \sim \chi^2_{n-1}$ with $S^2_{(n)} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_{(n)})^2$

Decision Rule: $\frac{(n-1)s^2_{(n)}}{\sigma_0^2} \in R \Rightarrow$ reject H_0

H_0	Rejection Region R
$\sigma^2 = \sigma_0^2$	$(-\infty, -t_{n-1,1-\frac{\alpha}{2}}) \cup \left(\chi^2_{n-1,1-\frac{\alpha}{2}}, \infty\right)$
$\sigma^2 \leq \sigma_0^2$	$(\chi^2_{n-1,1-\alpha}, \infty)$
$\sigma^2 \geq \sigma_0^2$	$(0, \chi^2_{n-1,\alpha})$

17 Two-Sampled

Hint 1: If it's not specified wheter to do an approximiation or an exact calculation use the R-Function (exact calculation) in the exam because it's faster.

Hint 2: If no alpha/α is given, then just use $\alpha = 0.05$.

Normal Model

Two-sample Gauß Test: μ_1, μ_2 unkown, σ_1, σ_2 known

The two groups are independent

Teststatistic: $T = \dfrac{\bar{X}_{(n_1)} - \bar{Y}_{(n2)} - \overbrace{(\mu_1 - \mu_2)}^{0}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_1^2}{n_2}}} \sim N(0,1)$

Hypotheses and Decision Rule:

	Null Hypotheses	Alternative Hypotheses	Decision Rule – reject H_0 if ...
a)	H_0: $\mu_1 = \mu_2$	H_1: $\mu_1 \neq \mu_2$	$\lvert T \rvert > u_{1-\frac{\alpha}{2}}$
b)	H_0: $\mu_1 \geq \mu_2$	H_1: $\mu_1 < \mu_2$	$T < u_\alpha$
c)	H_0: $\mu_1 \leq \mu_2$	H_1: $\mu_1 > \mu_2$	$T > u_{1-\alpha}$

Code:

R-Code:

```
X <- c(...)
Y <- c(...)
X.mean <- mean(X)
Y.mean <- mean(Y)
X.n <- length(X)
Y.n <- length(Y)
X.sigma <- ... # given in exercise
Y.sigma <- ... # given in exercise
test.stat <- (X.mean - Y.mean)/sqrt((X.sigma^2/X.n)+(Y.sigma^2/Y.n)) # approximation
```

R-Code: a) H_0: $\mu_1 = \mu_2$

```
# H0: mu1 = mu2; H1: mu1 != mu2
alpha <- ... # given in exercise
u <- qnorm(1-alpha/2)
test.stat > u    # TRUE => reject H0; FALSE => don't reject H
p.value <- pnorm(test.stat, 0, 1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: b) H_0: $\mu_1 \geq \mu_2$

```
# H0: mu1 >= mu2; H1: mu1 < mu2
alpha <- ... # given in exercise
u <- qnorm(1-alpha, 0, 1)
test.stat > u    # TRUE => reject H0; FALSE => don't reject H
p.value <- pnorm(test.stat, 0, 1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: c) H_0: $\mu_1 \leq \mu_2$

```
# H0: mu1 <= mu2; H1: mu1 > mu2
alpha <- ... # given in exercise
u <- qnorm(1-alpha, 0, 1)
test.stat > u    # TRUE => reject H0; FALSE => don't reject H0
p.value <- -pnorm(test.stat, 0, 1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

Two-sample t-Test: μ_1, μ_2 unkown, $\sigma_1 = \sigma_2$ but unknown

The two groups are independent

Pooled Sample Variance: $S_p^2 = \frac{(n_1-1)S_{X,n_1}^2 + (n_2-1)S_{Y,n_2}^2}{n_1+n_2-2}$

Teststatistic: $T = \frac{\bar{X}_{(n_1)} - \bar{Y}_{(n_2)} - \overbrace{(\mu_1-\mu_2)}^{0}}{S_p\sqrt{\frac{n_1+n_2}{n_1 \cdot n_2}}} \sim t_{n_1+n_2-2}$

Hypotheses and Decision Rule:

	Null Hypotheses	Alternative Hypotheses	Decision Rule – reject H_0 if ...
a)	H_0: $\mu_1 = \mu_2$	H_1: $\mu_1 \neq \mu_2$	$\lvert T \rvert > t_{n_1+n_2-2,1-\frac{\alpha}{2}}$
b)	H_0: $\mu_1 \geq \mu_2$	H_1: $\mu_1 < \mu_2$	$T < t_{n_1+n_2-2,\alpha}$
c)	H_0: $\mu_1 \leq \mu_2$	H_1: $\mu_1 > \mu_2$	$T > t_{n_1+n_2-2,1-\alpha}$

Code:

R-Code:

```
X <- c(...);          Y <- c(...)
X.mean <- mean(X);    Y.mean <- mean(Y)
X.n <- length(X);     Y.n <- length(Y)
X.sd <- sd(X);        Y.sd <- sd(Y) # maybe both already given in the exercise
S.p <- (((X.n-1)*X.sd^2)+(Y.n-1)*Y.sd^2)/(X.n+Y.n-2)
alpha <- ... # given in exercise
# --- Approximation ---
test.stat <- (X.mean - Y.mean)/(S.p*sqrt((X.n+Y.n)/(X.n+Y.n)))
# --- Exact ---
exact <- t.test(x = X, y = Y, alternative = "...", var.equal = TRUE, conf.level = 1-alpha)
```

R-Code: a) H_0: $\mu_1 = \mu_2$

```
# H0: mu1 = mu2; H1: mu1 != mu2
# --- Check Approximation ---
t <- qt(1-alpha/2, df = X.n+Y.n-2)
test.stat > t    # TRUE => reject H0; FALSE => don't reject H
p.value <- 2*pt(test.stat, df = X.n+Y.n-2) # ???
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "two.sided", var.equal = TRUE,
            conf.level = 1-alpha)$statistic
exact > t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "two.sided", var.equal = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: b) H_0: $\mu_1 \geq \mu_2$

```
# H0: mu1 >= mu2; H1: mu1 < mu2
# --- Check Approximation ---
t <- qt(alpha, df = X.n+Y.n-2)
test.stat < t    # TRUE => reject H0; FALSE => don't reject H
p.value <- pt(test.stat, df = X.n+Y.n-2)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "less", var.equal = TRUE,
            conf.level = 1-alpha)$statistic
exact < t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "less", var.equal = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: c) H_0: $\mu_1 \leq \mu_2$

```r
# H0: mu1 <= mu2; H1: mu1 > mu2
# --- Check Approximation ---
t <- qt(1-alpha, df = X.n+Y.n-2)
test.stat > t    # TRUE => reject H0; FALSE => don't reject H
p.value <- 1-pt(test.stat, df = X.n+Y.n-2))
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "greater", var.equal = TRUE,
            conf.level = 1-alpha)$statistic
exact > t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "greater", var.equal = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

Welsh Test: μ_1, μ_2 unkown, $\sigma_1 \neq \sigma_2$ but unknown

The two groups are independent

Teststatistic: $T = \dfrac{\bar{X}_{(n_1)} - \bar{Y}_{(n_2)} - \overbrace{(\mu_1 - \mu_2)}^{0}}{\sqrt{\frac{S^2_{X,n_1}}{n_1} + \frac{S^2_{Y,n_2}}{n_2}}} \sim t_\nu, \qquad$ with $\nu = \dfrac{\left(\frac{S^2_{X,n_1}}{n_1} + \frac{S^2_{Y,n_2}}{n_2}\right)^2}{\frac{(S^2_{X,n_1}/n_1)^2}{n_1 - 1} + \frac{(S^2_{Y,n_2}/n_2)^2}{n_2 - 1}}$

Hypotheses and Decision Rule:

	Null Hypotheses	Alternative Hypotheses	Decision Rule – reject H_0 if ...
a)	H_0: $\mu_1 = \mu_2$	H_1: $\mu_1 \neq \mu_2$	$\lvert T \rvert > t_{\nu,1-\frac{\alpha}{2}}$
b)	H_0: $\mu_1 \geq \mu_2$	H_1: $\mu_1 < \mu_2$	$T < t_{\nu,\alpha}$
c)	H_0: $\mu_1 \leq \mu_2$	H_1: $\mu_1 > \mu_2$	$T > t_{\nu,1-\alpha}$

Code:

R-Code:

```
X <- c(...);          Y <- c(...)
X.mean <- mean(X);    Y.mean <- mean(Y)
X.n <- length(X);     Y.n <- length(Y)
X.sd <- sd(X);        Y.sd <- sd(Y) # maybe both already given in the exercise
alpha <- ... # given in exercise
# --- Approximation ---
test.stat <- (X.mean - Y.mean)/(sqrt((X.sd^2/X.n)+(Y.sd^2/Y.n)))
# --- Exact ---
exact <- t.test(x = X, y = Y, alternative = "...", var.equal = FALSE, conf.level = 1-alpha)
```

R-Code: a) H_0: $\mu_1 = \mu_2$

```
# H0: mu1 = mu2; H1: mu1 != mu2
# --- Check Approximation ---
t <- qt(1-alpha/2, df = v)
test.stat > t    # TRUE => reject H0; FALSE => don't reject H
p.value <- 2*pt(test.stat, df = v) # ???
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "two.sided", var.equal = FALSE,
            conf.level = 1-alpha)$statistic
exact > t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "two.sided", var.equal = FALSE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: b) H_0: $\mu_1 \geq \mu_2$

```
# H0: mu1 >= mu2; H1: mu1 < mu2
# --- Check Approximation ---
t <- qt(alpha, df = v)
test.stat < t    # TRUE => reject H0; FALSE => don't reject H
p.value <- pt(test.stat, df = v)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "less", var.equal = FALSE,
            conf.level = 1-alpha)$statistic
exact < t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "less", var.equal = FALSE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: c) H_0: $\mu_1 \leq \mu_2$

```r
# H0: mu1 <= mu2; H1: mu1 > mu2
# --- Check Approximation ---
t <- qt(1-alpha, df = v)
test.stat > t    # TRUE => reject H0; FALSE => don't reject H
p.value <- 1-pt(test.stat, df = v))
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "greater", var.equal = FALSE,
            conf.level = 1-alpha)$statistic
exact > t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "greater", var.equal = FALSE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

Two-Paired t-Test: σ unkown

The two groups are the same

Paired Difference: $\bar{X} = \text{mean}(X - Y)$

Teststatistic: $T = \sqrt{n} \cdot \dfrac{\bar{X}_{(n)}}{\sqrt{s^2_{(n)}}} \sim t_{n-1}$

Hypotheses and Decision Rule:

	Null Hypotheses	Alternative Hypotheses	Decision Rule – reject H_0 if ...
a)	H_0: $\mu = 0$	H_1: $\mu \neq 0$	$\lvert T \rvert > t_{n-1,1-\frac{\alpha}{2}}$
b)	H_0: $\mu \geq 0$	H_1: $\mu < 0$	$T < -t_{n-1,1-\alpha}$
c)	H_0: $\mu_1 \leq 0$	H_1: $\mu > 0$	$T > t_{n-1,1-\alpha}$

Code:

R-Code:

```r
X <- c(...); Y <- c(...)
diff.mean <- mean(X-Y)
n <- length(X)
diff.sd <- sd(X-Y)
alpha <- ... # given in exercise
# --- Approximation ---
test.stat <- sqrt(n)*(diff.mean)/sqrt(diff.sd^2)
# --- Exact ---
exact <- t.test(x = X, y = Y, alternative = "...", mu = 0, paired = TRUE, conf.level = 1-alpha)
```

R-Code: a) H_0: $\mu = 0$

```r
# H0: mu1 = mu2; H1: mu1 != mu2
# --- Check Approximation ---
t <- qt(1-alpha/2, df = n-1)
test.stat > t    # TRUE => reject H0; FALSE => don't reject H
p.value <- 2*pt(test.stat, df = n-1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "two.sided", mu = 0, paired = TRUE,
            conf.level = 1-alpha)$statistic
exact > t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "two.sided", mu = 0, paired = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: b) H_0: $\mu \geq 0$

```r
# H0: mu1 >= mu2; H1: mu1 < mu2
# --- Check Approximation ---
t <- -qt(alpha, df = n-1)
test.stat < t    # TRUE => reject H0; FALSE => don't reject H
p.value <- pt(test.stat, df = n-1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "less", mu = 0, paired = TRUE,
            conf.level = 1-alpha)$statistic
exact < t        # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "less", mu = 0, paired = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: c) H_0: $\mu \leq 0$

```r
# H0: mu1 <= mu2; H1: mu1 > mu2
# --- Check Approximation ---
t <- qt(1-alpha, df = n-1)
test.stat > t     # TRUE => reject H0; FALSE => don't reject H
p.value <- 1-pt(test.stat, df = n-1))
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
t <- t.test(x = X, y = Y, alternative = "greater", mu = 0, paired = TRUE,
            conf.level = 1-alpha)$statistic
exact > t         # TRUE => reject H0; FALSE => don't reject H
p.value <- t.test(x = X, y = Y, alternative = "greater", mu = 0, paired = TRUE,
                  conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

F-Test

In the exercise there are often two types A and B

The two groups are independent

- Check wheter two variances are the equal or not equal:
  - If variances are not equal (rejected): t.test with `var.equal = FLASE`
  - If variances are equal (not rejected): t.test with `var.equal = TRUE`

Teststatistic: $T = \frac{S^2_{(n_1)}}{S^2_{(n_2)}} \sim F_{n_1-1,n_2-1}$

Hypotheses and Decision Rule:

	Null Hypotheses	Alternative Hypotheses	Decision Rule – reject H_0 if ...
a)	$H_0: \sigma_1 = \sigma_2$	$H_1: \sigma_1 \neq \sigma_2$	$T < F_{n_1-1,n_2-1,\frac{\alpha}{2}}$ or $T > F_{n_1-1,n_2-1,1-\frac{\alpha}{2}}$
b)	$H_0: \sigma_1 \geq \sigma_2$	$H_1: \sigma_1 < \sigma_2$	$T < F_{n_1-1,n_2-1,\alpha}$
c)	$H_0: \sigma_1 \leq \sigma_2$	$H_1: \sigma_1 > \sigma_2$	$T > F_{n_1-1,n_2-1,1-\alpha}$

Code:

R-Code:

```
X <- c(...); Y <- c(...)
X.sd <- sd(X); Y.sd <- sd(Y)
X.n <- length(X); Y.n <- length(Y)
alpha <- ... # given in exercise
# --- Approximation ---
test.stat <- (X.sd^2)/(Y.sd^2)
# --- Exact ---
exact <- var.test(x = X, y = Y, alternative = "...", conf.level = 1-alpha)
```

R-Code: a) $H_0: \sigma_1 = \sigma_2$

```
# H0: sigma1 = sigma2; H1: sigma1 != sigma2
# --- Check Approximation ---
F1 <- qf(alpha/2, df1 = X.n-1, df2 = Y.n-1)
F2 <- qf(1-alpha/2, df1 = X.n-1, df2 = Y.n-1)
test.stat < F1 | test.stat > F2 # TRUE => reject H0; FALSE => don't reject H
p.value <- 2*pf(test.stat, df1 = X.n-1, df2 = Y.n-1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
F <- var.test(x = X, y = Y, alternative = "two.sided", conf.level = 1-alpha)$statistic
exact > F        # TRUE => reject H0; FALSE => don't reject H
p.value <- var.test(x = X, y = Y, alternative = "two.sided", conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: b) $H_0: \sigma_1 \geq \sigma_2$

```
# H0: sigma1 >= sigma2; H1: sigma1 < sigma2
# --- Check Approximation ---
F <- qf(alpha, df1 = X.n-1, df2 = Y.n-1)
test.stat < F    # TRUE => reject H0; FALSE => don't reject H
p.value <- pf(test.stat, df1 = X.n-1, df2 = Y.n-1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
F <- var.test(x = X, y = Y, alternative = "less", conf.level = 1-alpha)$statistic
exact < F        # TRUE => reject H0; FALSE => don't reject H
p.value <- var.test(x = X, y = Y, alternative = "less", conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```

R-Code: c) $H_0: \sigma_1 \leq \sigma_2$

```r
# H0: sigma1 <= sigma2; H1: sigma1 > sigma2
# --- Check Approximation ---
F <- qf(alpha, df1 = X.n-1, df2 = Y.n-1)
test.stat > F    # TRUE => reject H0; FALSE => don't reject H
p.value <- pf(test.stat, df1 = X.n-1, df2 = Y.n-1)
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0

# --- Check Exact ---
F <- var.test(x = X, y = Y, alternative = "greater", conf.level = 1-alpha)$statistic
exact > F        # TRUE => reject H0; FALSE => don't reject H
p.value <- var.test(x = X, y = Y, alternative = "greater", conf.level = 1-alpha)$p.value
p.value < alpha  # TRUE => reject H0; FALSE => don't reject H0
```