```
sin(2*pi)
cos(3/4)
tan(pi)
log(100, base=10)
log(15); log(15, base=exp(1))
log(1/7, base=7)
exp(3)
64^(1/3)

vector = 1:10
vector
sum(vector)
vec = seq(1, 10, 1)
vec
sum(vec)

x = seq(2, 20, 2)
x
correct = x*x==x^2
sum(!correct) # always matches
rev(x)
sum(x)
length(x)
sqrt(sum(x^2))

y = seq(5, 10, length.out=13)
y
length(y)

z1 = rep(c(1,2), 5)
# IMPORTANT
z2 = rep(c(1,2), each=5)
z1; z2
z1*z2 # piecewise multiplication
t(z1)%*%z2
z1%*%t(z2)

a = c(1,3,6,2,7,4, 0)
min(a)
# which is about indices
which.min(a) # the first one
which(a==min(a)) # all indices
which(a<=4)
sum(a)
sum(a^2)
length(a)
a[3]
a = a + 4
```

```
a
#IMPORTANT
b = a[-4] #exclude the 4th element, -c(4,5,...) for more elements
b
c = a + b
c
d = a[a>4] #filtering
d

A = matrix(c(2,1,1,3,-1,1,0,2,-1), ncol=3)
A
t(A)
det(A)
#IMPORTANT trace
sum(diag(A))
A*A
A%*%A
A %*% diag(A)
solve(A) # find the inverse
#IMPORTANT
A_inverse = solve(A)
round(A %*% A_inverse)
a = A[,3] # select col
b = A[2,] # select row
b
t(a) %*% b
a %*% t(b)

x = 1:10
y = c(3, 5, 4, 8, 6, 9, 11, 10, 13, 12)
plot(x, y)
df = data.frame(x, y)
df
plot(df, xlab="X", ylab="Y")
plot(rbind(x, y)) # these are just two strange points (dim too high)
plot(cbind(x, y)) # similar to data.frame
cbind(x, y)

x = seq(-3, 4, 0.1)
f = x^2+3*x-5
plot(x, f, type="l")

# A = cbind(c(2,1,1),c(3,-1,1),c(0,2,-1))
# also rbind()
# b=as.matrix(b)
dim(df)
length(df)
```

```
# task 1
flights = read.csv("flights.csv", sep=";")
flights
install.packages("arm")
library("arm")
# a)
class(flights)
# class(flights[,1]) "integer"
#b)
par(mfrow=c(2,3))
colours = c("green", "yellow", "blue", "red", "orange", "purple")
for(year in 1:6){
  title = substr(names(flights)[year], 2, 5)
  hist(flights[,year], main=paste0("Year ", title), xlab="Number of monthly passengers",
ylab="Frequency", col=colours[year])
}
#c)
mean(flights[,1])
for(year in 1:6){
  cat(paste0("Statistical characteristics for year ", names(flights)[year], "\n"))
  cat(paste0("Mean: ", mean(flights[,year]), "\n"))
  cat(paste0("Median: ", median(flights[,year]), "\n"))
  i = 0
  for(num in quantile(flights[,year])){
    cat(paste0(25*i, "%: ", num, "\n"))
    i = i + 1
  }
  cat(paste0("Standard deviation: ", sd(flights[,year]), "\n"))
  cat(paste0("Variability index: ", sd(flights[,year])/mean(flights[,year]), "\n"))
}

for(year in 1:6){ # bad approach, don't sharey!!!
  data = flights[,year]
  boxplot(flights[,year])
}
par(mfrow=c(1,1))
boxplot(flights) # much better

# task 2
notes = read.csv("notes.csv", sep=";", dec=",")
notes
table(na.omit(notes["group.M1"]))
#c)
for(group in 1:length(notes)){
```

```
  group_data = na.omit(notes[,group])
  group_name = names(notes)[group]

  freq_table = table(group_data)
  #IMPORTANT: change the table/freqtable name
  names(dimnames(freq_table)) = group_name

  print(freq_table)
}
#d)
library("arm")
discrete.histogram(notes[,1])
par(mfrow=c(2,2))
for(group in 1:length(notes)){
  title = names(notes)[group]
  discrete.histogram(notes[, group], main=title)
}
#e)
for(group in 1:length(notes)){
  title = names(notes)[group]
  data = na.omit(notes[,group])
  cat(paste0("Statistics for ", title, ":\n"))
  cat(paste0("Mean: ", mean(data), "\n"))
  cat(paste0("Median: ", median(data), "\n"))
  cat(paste0("Standard deviation: ", sd(data), "\n"))
  cat(paste0("Variability index: ", sd(data)/mean(data), "\n\n"))
}
#f)
boxplot(notes) # uses all available data in each column separately
boxplot(na.omit(notes)) # removes any row that has at least one NA anywhere in the
dataframe
#!IMPORTANT!
# on the test use na.omit (to keep them balanced)

#g)
for(group in 1:length(notes)){
  print(paste0("Group ", names(notes)[group]))
  print(table(notes[,group]))
}
par(mfrow=c(2,2))
for(group in 1:length(notes)){
  title = names(notes)[group]
  #IMPORTANT pie() requires table()
  pie(table(notes[,group]), main=title)
}

# task 3
straws = read.csv("strawberries.csv", sep=";")
```

```
straws

#b)
# interval frequency table
for(year in 1:length(straws)){
  #intFreq = cut(straws[,year], breaks=seq(70,140, 10))
  # for each year, drop NANs separately
  data = na.omit(straws[,year])
  intFreq = cut(data, breaks=seq(min(data),max(data), length=7))
  print(table(intFreq))
}
#c)
par(mfrow=c(1,2))
for(year in 1:length(straws)){
  title = names(straws)[year]
  #data = straws[,year]
  data = na.omit(straws[,year])
  #IMPORTANT nice for hist breaks!!!
  br = seq(min(data), max(data), length=floor(sqrt(length(data))))
  hist(data, freq=F, main=title, xlab="Yields of strawberries", breaks=br)
}
#d)
for(year in 1:length(straws)){
  title = names(straws)[year]
  data = na.omit(straws[,year])
  cat(paste0("Statistics for ", title, ":\n"))
  cat(paste0("Mean: ", mean(data), "\n"))
  cat(paste0("Median: ", median(data), "\n"))
  cat(paste0("Standard deviation: ", sd(data), "\n"))
  cat(paste0("Variability index: ", sd(data)/mean(data), "\n\n"))
}
#e)
par(mfrow=c(1,1))
# omit all of them at the same time to preserve balance
boxplot(na.omit(straws))
#f)
non_na_data = na.omit(straws)
minimum=min(non_na_data[,1], non_na_data[,2])
maximum=max(non_na_data[,1], non_na_data[,2])
par(mfrow=c(1,2))
for(year in 1:length(non_na_data)){
  freqTable = table(cut(straws[,year], breaks=seq(minimum, maximum, length.out=7)))
  # length() may be tricky with 2D entities, sth dim() is needed
  #k = floor(sqrt(dim(non_na_data)[1]))
  #freqTable = table(cut(straws[,year], breaks=seq(minimum, maximum, length.out=k)))
  title = names(straws)[year]
  pie(freqTable, main=title)
}
```

```
# may be useful:
dim(notes)
length(na.omit(notes[,1]))
colSums(!is.na(notes))




n = 10
p = 0.6
probs = dbinom(0:n, n, p)
plot(probs, type="l")

x = 0:n
x>=2
which(x>=2)
sum(probs[x>=2])

exp(1)
y = pexp(x, 0.25) # cumulative sum
plot(x, y, type="l")
plot(x, dexp(x, 0.25), type="l") # prob at a given point

plot(0:180, pbinom(0:180, 180, 0.5))

# P(X>30.45)=1-F(30.45)
mu=30; sig=0.2
1-pnorm(30.45, mu, sig)

# TASKS

# task 1
# rbinom() counts successes
n=30;p=0.3
rbinom(1, n, p) #typically <10
p=0.5
rbinom(1, n, p) # ~15
p=0.7
rbinom(1, n, p) # typically >20

#task 2
library("arm")
n=30; p=0.3
successes = rbinom(20, n, p)
#hist(successes, prob=T, breaks=seq(0,30,1))
title = paste0("p = ", p)
```

```
#IMPORTANT: so called line graph in questions
discrete.histogram(successes, main=title)

par(mfrow=c(2,3))
n=30
for(p in seq(0.3, 0.7, 0.1)){
  successes = rbinom(20, n, p)
  title = paste0("p = ", p)
  discrete.histogram(successes, freq=F, main=title, xlim=c(0,30))
}

# task 3
# S - number of wells contaminated
# S ~ bin(n=5, p=0.3)
# p - probability well is contaminated
# n - sample size (5)
n=5; p=0.3
#a)
# P(S==3)
dbinom(3, n, p)
#b) P(S>=3)
1 - pbinom(2, n, p)
#c) P(S<3)
pbinom(2, n, p)

# OR (works only for discrete cases (bin distribution))
S = 0:5
probs = dbinom(S, n, p)
probs
prob[S==3]
sum(probs[S>=3])
sum(probs[S<3])

# task 4
# p - probability that a fluorescent bulb burns for at least 500 hours
# B - the number of bulbs which burn for at least 500 hours
# B ~ bin(n=8, p=0.9)
# n - sample size (8)
p=0.9; n = 8
B = 0:8
probs = dbinom(B, n, p)
#a) P(B==8)
probs[B==8]
#b) P(B==7)
probs[B==7]
#c) P(B>5)
sum(probs[B>5])
#d) E(B)
```

```
sum(probs*B) # n*p
#e) SD(B)
variance = sum(probs*B^2) - sum(probs*B)^2
sqrt(variance) # sqrt(n*p*(1-p))
#!IMPORTANT!
# std=sqrt(n*p*(1-p)) for bin distribution!

# task 5
# X - number of days a power cell is functioning
# X ~ exp(lambda=0.01)
par(mfrow=c(1,1))
lambda = 0.01
n = 2
x=0:500
curve(dexp(x, lambda), xlim=c(0,500))
# <= and  < don't matter here as it's continuous
# !IMPORTANT! DISTINCT BETWEEN CONTINUOUS AND DISCRETE DIST
# WHETHER TO SUBTRACT 1 OR NOT
#a) P(X>=200) = 1 - P(X<200) = 1 - F(200)
1 - pexp(200, lambda)
#b) P(X<=100) = F(100)
pexp(100, lambda)
#c) P(X<500) = F(500)
pexp(500, lambda)

# task 6
# X - magnitude of earthquake recorded in a region of North America
# X ~ exp(lambda=1/2.4)
mu = 2.4
#IMPORTANT lambda from mean
lambda = 1/mu
#a) P(X>3) = 1 - P(X<=3) = 1 - F(3)
1 - pexp(3, lambda)
#b) P(2<=X<=3) = P(X<=3) - P(X<2) = F(3) - F(2)
pexp(3, lambda) - pexp(2, lambda)

f = function(x){lambda*exp(-lambda*x)}
integrate(f, 3, Inf)
integrate(f, 2, 3)

# for continuous random variables, the expectation is defined as:
# E(X) = integral(x * f(x))
f2 = function(x){lambda*exp(-lambda*x)*x}
integrate(f2, 0, Inf)
# OR
integrate(function(x){x*f(x)}, 0, Inf)

# task 7
```

```
# X - measured resistance of the wires produced by company A
# X ~ N(mu=0.13, sig=0.005)
mu = 0.13
sig = 0.005
x = seq(0.1, 0.16, 0.01)
curve(dnorm(x, mu, sig), xlim=c(0.1, 0.16))
# P(0.12<X<0.14) = P(X<0.14) - P(X<=0.12) = F(0.14) - F(0.12)
pnorm(0.14, mu, sig) - pnorm(0.12, mu, sig)

# task 8
# X - paint-drying time
# X ~ N(120, 15)
# mu - mean of paint-drying time (120 minutes)
# sig - standard deviation of paint-drying time (15 minutes)
mu=120; sig=15
x = 60:180
curve(dnorm(x, mu, sig), xlim=c(60,180))
# P(110<=X<=135) = P(X<=135) - P(X<110) = F(135) - F(110)
pnorm(135, mu, sig) - pnorm(110, mu, sig)

# decimal coding
# X ~ N(mu=2, sig=0.25)
mu=2; sig=0.25
x = seq(1, 3, 1/60)
curve(dnorm(x, mu, sig), xlim=c(1, 3))
pnorm(2.25, mu, sig) - pnorm(11/6, mu, sig)

# task 9
# X - maximum vehicle speed
# X ~ N(mu=46.8, sig=1.75)
mu=46.8; sig=1.75
#a) P(X<=50) = F(50)
pnorm(50, mu, sig)
#b) P(X>=48) = 1 - P(X<48) = 1 - F(48)
1 - pnorm(48, mu, sig)

# task 10
# X ~ bin(n=20, p=0.2)
n=20; p=0.2
nums = rbinom(500, n, p)
#a)
par(mfrow=c(1,3))
library("arm")
discrete.histogram(nums, main="Prob hist of generated data")
plot(0:20, dbinom(0:20, n, p), type="h", freq=F, main="binomial distribution")

mu = n * p
x = 0:20
```

```
probs = dbinom(x, n, p)
variance = sum(probs*x^2) - sum(probs*x)^2
sig = sqrt(variance)
mu; sig
# or sig=sqrt(n*p*(1-p))
# X ~(app) N(mu=4, sig=1.788854)
curve(dnorm(x, mu, sig), xlim=c(0, 20), main="normal distribution")
#b)
par(mfrow=c(1,1))
hist(nums, breaks=0:20, freq=F)
curve(dnorm(x, mu, sig), xlim=c(0,20), col="red", add=T)
#c) just play around

# task 11
# X - the number of students who applied for financial aid
# X ~ bin(n=100, p=0.25)
n=100; p=0.25

# P(X<=15) = F(15)
# exact probability
pbinom(15, n, p)
mu = n*p
sig = sqrt(n*p*(1-p))
mu; sig
# X ~(app) N(mu=25, sig=4.330127)
# approximate probability
pnorm(15, mu, sig)

# useful for checking for mistakes: (p & d)
#x=0:100
#curve(pbinom(x, n, p), xlim=c(0,100))

# task 12
# standard normal distribution - default params
# X - individual observation from standard normal distribution
# X ~ N(mu=0, sig=1)
vectormean = rep(0,0)
for(i in 1:200){
  sample = rnorm(30)
  sample_mean = mean(sample)
  vectormean[i] = sample_mean
}
hist(vectormean, freq=F)
# !IMPORTANT!
x=seq(min(vectormean), max(vectormean), length=100)
mu=0
# !IMPORTANT!
sigma=1/sqrt(30)
```

```
# The vectormean values ~ N(mu=0, sig=1/sqrt(30))
curve(dnorm(x,mu,sigma), add=T, col="red")

# task 13
# X - number of successes in 30 Bernoulli trials
# X ~ bin(n=30, p=0.4)
n=30; p=0.4
vector = rep(0,0)
sample_size = 10
for(i in 1:200){
  sample = rbinom(sample_size, n, p)
  vector[i] = mean(sample)
}
hist(vector, freq=F)
mu = n*p
sig = sqrt(n*p*(1-p)) / sqrt(sample_size)
mu; sig
# Xbar ~(app) N(mu=12, sig=0.8485281)
x=seq(min(vector), max(vector), length=100)
curve(dnorm(x, mu, sig), col="red", add=T)

# task 14
# n - sample size (50)
# X - resistance of a resistor
# X ~ N(mu=200, sig=10)
mu=200; sig=10; n=50
# Y - average resistance of 50 resistors
# Y ~ N(mu, sig/sqrt(n))
#a) P(199<=Y<=202) = P(Y<=202) - P(Y<199) = F(202) - F(199)
pnorm(202, mu, sig/sqrt(n)) - pnorm(199, mu, sig/sqrt(n))

# Z - total resistance of 50 resistors
# Z ~ N(n*mu, sig*sqrt(n))
#b) P(Z<=10020) = F(10020)
pnorm(10020, mu*n, sig*sqrt(n))

# task 15
# X - blood cholesterol level of a worker
# X ~ ?(mu=202, sig=14)
# n - sample size (64)
mu=202; sig=14; n=64
# Xbar - average blood cholesterol level of a worker
# Xbar ~(app) N(mu, sig/sqrt(n)) # can approximate by Central Limit Theorem (large sample,
n>=30)

# P(198<=Xbar<=206) = P(Xbar<=206) - P(Xbar<198) = F(206) - F(198)
pnorm(206, mu, sig/sqrt(n)) - pnorm(198, mu, sig/sqrt(n))
```

```
# task 16
# X - strength of a thread
# X ~ ?(mu=0.5, sig=0.2)
# n - number of threads
mu=0.5; sig=0.2; n=100
# Xtotal - strength of a rope (summed strength of n threads)
# Xtotal ~(app) N(mu*n, sig*sqrt(n)) # can approximate by Central Limit Theorem (large sample, n>=30)

# P(Xtotal>=47) = 1 - P(Xtotal<47) = 1 - F(47)
1 - pnorm(47, mu*n, sig*sqrt(n))




# q - Quantile functions (INVERSE of p-functions)
# qnorm(0.9) - find x such that P(X<=x)=0.9

# qt(0.9, 24)
# This means the t-distribution was estimated using a sample of size 25

qnorm(0.9)     # Returns ~1.28 because P(Z ≤ 1.28) = 0.90
qt(0.9, 24)    # Returns ~1.32 because P(T ≤ 1.32) = 0.90 (t-distribution, df=24)
qt(0.9, 7)     # Returns ~1.41 because P(T ≤ 1.41) = 0.90 (t-distribution, df=7)

# Answer: "What value corresponds to this cumulative probability?

# TASKS
rm(list=ls())
data = read.csv("data_est.csv", sep=";", dec=".")
data

# task 1
weights = na.omit(data$diamonds) #returns a vector
#weights = na.omit(data[,"diamonds"]) #returns a dataframe
#weights = na.omit(data["diamonds"]) #returns a vector
weights[1:length(weights)]

#a)
# population: all synthetic diamonds produces
# sample: 12 synthetic diamonds
# random variable: the weight of a synthetic diamond
#b) the point estimate of mean weight
mean(weights)
#c) estimate the mean weight with 95% confidence
```

```
xbar = mean(weights)
alpha = 0.05
n = length(weights)
sig = sd(weights)
L = xbar - qt(1-alpha/2, n-1)*sig/sqrt(n)
U = xbar + qt(1-alpha/2, n-1)*sig/sqrt(n)
c(L, U)
# !IMPORTANT! round L down, and U up, lengths mathematically
# we are 95% confident that the interval 0.498 to 0.57 covers the true mean
# weight of the population of synthetic diamonds produced
x = t.test(weights, conf.level=1-alpha)
x$conf.int
U-L; x$conf.int[2] - x$conf.int[1]
#d) estimate the mean weight with 98% confidence
xbar = mean(weights)
sig = sd(weights)
alpha = 0.02
L = xbar - qt(1-alpha/2, n-1)*sig/sqrt(n)
U = xbar + qt(1-alpha/2, n-1)*sig/sqrt(n)
U-L
# for 95% confidence the interval length is 0.071
# for 98% confidence the interval length is 0.087
# for 99% confidence the interval length is 0.1
#e)
variance = var(weights)
std = sqrt(variance)
variance; std
#f)
alpha = 0.05
n = length(weights)
std = sd(weights)
# !IMPORTANT! the formula on the slides is for VARIANCE
L = sqrt((n-1)*std^2/qchisq(1-alpha/2, n-1))
U = sqrt((n-1)*std^2/qchisq(alpha/2, n-1))
c(L, U)
# we are 95% confident that the interval 0.039 to 0.095 covers the true standard deviation
# of weight of the population of synthetic diamonds produced

# task 2
milk = na.omit(data$milk)

#a)
# population - milk of all nursing women
# sample - milk of 20 nursing women
# measurement - the amount of PCB in the milk of nursing mothers
# better:
# measurement - the amount of PCB in the milk of a single nursing mother
#b)
```

```
mean(milk)
#c)
var(milk)
sd(milk)
#d) estimate the average amount of PCB in the milk with 95% confidence
alpha = 0.05
xbar = mean(milk)
n = length(milk)
std = sd(milk)
# !IMPORTANT! don't mix up qt() and qchisq() !!! (ALSO ^2 IS JUST THE NOTATION, NOT
THE SQUARE)
L = xbar - qt(1-alpha/2, n-1)*std/sqrt(n)
U = xbar + qt(1-alpha/2, n-1)*std/sqrt(n)
c(max(L, 0), U)
# we are 95% confident that the interval 3.42 to 8.18 covers the true population mean
#e) estimate the variance and standard deviation of the
# amount of PCB in the milk with 95% confidence
alpha = 0.05
n = length(milk)
std = sd(milk)
L = (n-1)*std^2/qchisq(1-alpha/2, n-1)
U = (n-1)*std^2/qchisq(alpha/2, n-1)
# interval for variance:
c(L, U)
# interval which covers standard deviation
c(sqrt(L), sqrt(U))
# we are 95% confident that the interval 14.951 to 55.151 covers the true  population
variance and that the
# interval 3.866 to 7.427 covers the true population standard deviation

# task 3
cigarettes = na.omit(data$cigarettes)
# nicotine content ~ N(mu=?, sig=0.7)
sig = 0.7
# population - nicotine content of all newly marketed cigarettes
#a) estimate the population mean with 95% confidence
alpha = 0.05
xbar = mean(cigarettes)
n = length(cigarettes)
L = xbar - qnorm(1-alpha/2)*sig/sqrt(n)
U = xbar + qnorm(1-alpha/2)*sig/sqrt(n)
c(L, U)
# we are 95% confident that the interval 1.455 to 2.164 covers the true population mean
#install.packages("BSDA")
library(BSDA)
x = z.test(cigarettes, sigma.x=sig, conf.level=0.95)
x$conf.int
```

```
#b) calculate the smallest sample needed for the 95% confidence interval to be not longer
than 0.3 milligrams
# len = Xbar+qnorm(1-alpha/2)*sig/sqrt(n)-Xbar+qnorm(1-alpha/2)*sig/sqrt(n)
# len = 2*qnorm(1-alpha/2)*sig/sqrt(n)
# sqrt(n) = 2*qnorm(1-alpha/2)*sig/len
# min_n = (2*qnorm(1-alpha/2)*sig/len)^2
alpha = 0.05
n = length(cigarettes)
min_n = (2*qnorm(1-alpha/2)*sig/0.3)^2
ceiling(min_n)
# a sample of size greater than or equal to 84
#c) compute the estimate of standard deviation and compare it with population standard
deviation
std = sd(cigarettes)
std
# the estimate of standard deviation is noticeably smaller than population standard deviation

# task 4
signals = na.omit(data$signal)
# X - intensity of a signal recorded at location B
# X ~ N(mu=?, sig=3)
# n - number of times a signal is sent (10)
sig = 3
mu = 0
n = 10
# Xbar - average intensity of a signal among 10 "copies" of it
#IMPORTANT look it average/total
# Xbar ~ N(?, sig/sqrt(n))

# calculate the point estimate of mu
Xbar = mean(signals)
Xbar
# estimate the true signal intensity at location A with 95% confidence
alpha = 0.05
#IMPORTANT these formulas already account for /sqrt(n) for means
L = Xbar - qnorm(1-alpha/2)*sig/sqrt(n)
U = Xbar + qnorm(1-alpha/2)*sig/sqrt(n)
c(L, U)
# we are 95% confident that the interval 17.44 to 21.16 covers the true
# signal intensity at location A

# task 5
# X - time span of a phone call made during midday
# X ~ ?(mu=4.7, sig=2.2)
# n - sample size (1200)
# population - lengths of all phone calls made during midday using the company's service
# sample - 1200 phone calls' lengths selected from population
mu=4.7; sig=2.2; n=1200
```

```
# Xbar - average span of a phone call made during midday among the sample
#IMPORTANT: if don't know distribution, avg/total ~(app) normal + Central Limit Theorem
# Xbar ~(app) N(mu, sig/sqrt(n)) # can approximate by Central Limit Theorem because the sample if large (n>=30)

# estimate the population mean with 95% confidence
alpha = 0.05
L = mu - qt(1-alpha/2, n-1)*sig/sqrt(n)
U = mu + qt(1-alpha/2, n-1)*sig/sqrt(n)
c(L, U)
# we are 95% confident that the interval 4.575 to 4.825 covers the true population mean
alpha = 0.05
L = sqrt((n-1)*sig^2/qchisq(1-alpha/2, n-1))
U = sqrt((n-1)*sig^2/qchisq(alpha/2, n-1))
c(L, U)
# we are 95% confident that the interval 2.115 to 2.292 covers the true population standard deviation

# task 6
transistors = na.omit(data$lifetime)
n = length(transistors)
# population: lifetime of all General Electric transistors
# sample: 36 General Electric transistors selected from population
# X - a lifetime of a General Electric transistor
# Xbar - mean lifetime of General Electric transistors among the sample
avg = mean(transistors)
std = sd(transistors)
avg; std
# !IMPORTANT! how to write sample estimates to distribution:
# X ~ ?(mu=avg, sig=std) # using sample estimates for population parameters
# Xbar ~(app) N(avg, std/sqrt(n)) # can approximate by Central Limit Theorem because the sample is large (n>=30)

# !IMPORTANT! don't divide by sqrt(n), it's in the formula
#S = std/sqrt(n)
alpha = 0.05
L = avg - qt(1-alpha/2, n-1)*std/sqrt(n)
U = avg + qt(1-alpha/2, n-1)*std/sqrt(n)
c(L, U)
# we are 95% confident that the interval 1162.505 to 1231.375 covers the true population mean

# task 7
# X - setting time of cement mixture
# X ~ N(mu=?, sig=5)
# n - sample size required to obtain a given precision
sig = 5
```

```
# !IMPORTANT! ERROR IS THE MAX ACCEPTABLE DEVIATION FROM XBAR, NOT U-L!!
# error = qnorm(1-alpha/2)*sig/sqrt(n)
# sqrt(n) = qnorm(1-alpha/2)*sig/error
# n = (qnorm(1-alpha/2)*sig/error)^2
alpha = 0.05
n = (qnorm(1-alpha/2)*sig/1)^2
# !IMPORTANT!
# "musisz miec wiecej samples zeby zminimalizowac error"
ceiling(n)
# the sample size should be at least 97

# task 8
# X - weight of salmon grown at a commercial hatchery
# Xbar - average weight of salmon grown at a commercial hatchery
sig=0.3
error=0.1
# error = qnorm(1-alpha/2)*sig/sqrt(n)
# n = (qnorm(1-alpha/2)*sig/error)^2
# if we want to be 90% certain:
alpha=0.1
n = (qnorm(1-alpha/2)*sig/error)^2
ceiling(n)
# we are 90% confident that that our estimate of the mean weight of salmon is
# correct within +-0.1 pounds if sample size is at least 25
# if we want to be 99% sure:
alpha=0.01
n = (qnorm(1-alpha/2)*sig/error)^2
ceiling(n)
# we are 99% confident that that our estimate of the mean weight of salmon is
# correct within +-0.1 pounds if sample size is at least 60

# task 9
# population - all computers of a particular brand and model
# sample - 100 computers selected from population
# n - sample size (100)
n = 100
x = 3
p_hat = x/n
# X - number of times computers were down
# X ~ bin(100, 3/n) # using sample estimates for population parameters
alpha = 0.05

L = max(p_hat - qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n), 0)
U = p_hat + qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
c(L, U)
# we are 95% confident that interval 0 to 0.0635 covers the true population proportion of
unscheduled downtime
binom.test(x, n, conf.level=1-alpha)$conf.int
```

```
prop.test(x, n, conf.level=1-alpha)$conf.int

# task 10
# population - all cans in a brewery
# sample - 100 cans selected from brewery
# p_hat - sample proportion of underfilled cans
# n - sample size (100)
# x - number of underfilled cans found in a sample
x = 4; n = 100; alpha = 0.05
p_hat = x/n
# estimate the true population proportion of underfilled cans
L = p_hat - qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
U = p_hat + qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
c(L, U)
# we are 95% confident that interval 0.0015 to 0.0785 covers the true population proportion
of underfilled cans

# task 11
# population - all upholstery-installer teams in an automobile assembly plant, where each
team is either arranging materials besides work stations or not
# sample - 120 teams randomly selected from population
# n - sample size (120)
# x - number of teams arranging materials beside their work station (24)
# p_hat - sample proportion  of teams arranging materials besides work stations
n=120; x=24
p_hat = x/n
alpha = 0.1
# estimate the true population proportion of teams arranging materials besides work stations
L = p_hat - qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
U = p_hat + qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
c(L, U)
# we are 90% confident that interval from 0.1399 to 0.2601 covers the true population
proportion of teams arranging materials besides work stations

# task 12
# population - all construction workers , where each worker is either employed or
unemployed
# sample - 1000 workers randomly selected from population
# n - sample size(1000)
# p_hat - sample proportion of workers who are unemployed
n = 1000; x = 122
alpha = 0.1
p_hat = x/n
# estimate the true population proportion of workers who are unemployed
L = p_hat - qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
U = p_hat + qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
c(L, U)
```

```
# we are 90% confident that interval 0.1049 to 0.1391 covers the true population proportion of workers who are unemployed

# task 13
alpha = 0.02
error = 0.05
# error - estimation error, we accept +- 0.05
# p_hat - sample proportion of people who have sight problems in a particular group

# error = qnorm(1-alpha/2)*sqrt(p_hat*(1-p_hat)/n)
# p_hat*(1-p_hat)/n = (error/qnorm(1-alpha/2))^2
# n = p_hat*(1-p_hat)/(error/qnorm(1-alpha/2))^2
# a) nothing is known about p_hat, assume p_hat=0.5
p_hat = 0.5
n = p_hat*(1-p_hat)/(error/qnorm(1-alpha/2))^2
ceiling(n)
# 542 people should be examined to obtain an estimation error +-0.05 on confidence level 0.98
#b) p_hat = 0.3
p_hat = 0.3
n = p_hat*(1-p_hat)/(error/qnorm(1-alpha/2))^2
ceiling(n)
# 455 people should be examined to obtain an estimation error +-0.05 on confidence level 0.98
```