Part II: R and Probability distributions

TMA4268 Statistical Learning V2020. Module 1: INTRODUCTION TO STATISTICAL LEARNING

Martina Hall, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, NTNU

January 06, 2020

(Latest changes: 03.01.20: first version for 2020)

Before working with this file you should have worked with the `Rbeginner` file here.

If you read the solutions version (file name ending in "sol") the **solutions to Exercises are included.**

Random variables and probability distributions

To be able to do statistics, a good understanding of random variables and probability distributions is vital.

Background

Do you read Norwegian and want to brush up on these topics (theory, not R)? Then go to the thematic pages from TMA4240/TMA4245 Statistics and read about:

- Random variables and probability distributions
- Important discrete distributions
- Important continuous distributions

We will also make a function to do a z-test (so, just a bit of inference) and to calculate a p-value.

Probability distributions in R

Tools for working with probability distributions are included in the default environment in R, and we will now look at the *binomial* (discrete) distribution and the *normal*, *chi-squared*, t and Fisher F (continuous distribution). The multivariate normal distribution will be covered in the interatice lecture of Module 2 of TMA4268 (and is also a large part of TMA4267).

For each of these distributions, there exists functions that calculate the pdf (probability density or mass function, often denoted f in previous courses), the cumulative distribution function (often denoted F), the inverse cumulative distribution function (often denoted F^{-1}) and drawing random numbers form the given distribution. For the normal distribution these functions are called:

Function	Meaning
`dnorm()`	density (pdf)
`pnorm()`	cumulative distribution function (cdf)
`qnorm()`	inverse cumulative distribution (quantile function)
`rnorm()`	draw random variable from normal distribution

Corresponding functions for the binomial distribution are called `dbinom()`, `pbinom()`, `qbinom()` and `rbinom()`. Similar functions exists for many distributions, here is a short list of some distributions that might come in

handy (only listed with density):

Function	Distribution
`dnorm()`	Normal
`dt()`	Student-t
`dchisq()`	Chi-squared
`df()`	F-distribution
`dunif()`	Uniform
`dbinom()`	Binomial
`dbinom(..,size=1)`	Bernoulli
`dpois()`	Poisson
`dexp()`	Exponential

Remember to check the help pages of the functions you want to use, to see what is used as input for each function. We will now look at each of these functions.

Probability density or mass function $f(x)$

Binomial distibution

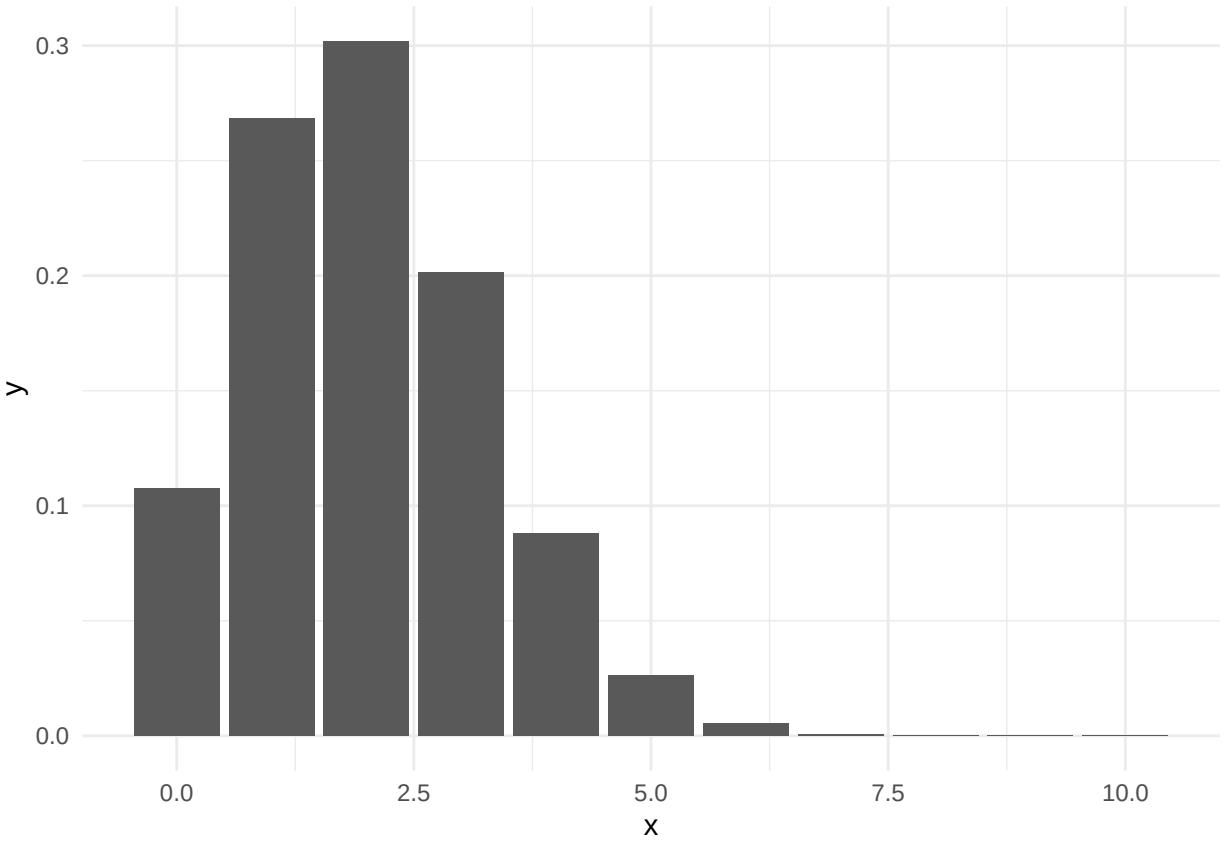
If we study a discrete distibution – like the binomial distribution – the probability mass function gives the probability of observing x, $f(x) = P(X = x)$.

Exercise: What are the requirements that a random variable X follows a binomial distribution? What are the parameters of the distribution?

Solution: X follows a binomial distribution if its probability mass function is given by $P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$ for $x = 0, 1, \ldots, n$, where the parameter n is a positive integer and the parameter p is a number between 0 and 1. Typically, X is the number of successes in a Bernoulli process consisting of n trials where the probability of success in each trial is p.

To plot the pdf of the binomial we may construct a barplot. We do this now using the `ggplot2` package.

```
library(ggplot2)
n = 10
p = 0.2
probs = dbinom(0:10, n, p)
df = data.frame(x = 0:10, y = probs)
p <- ggplot(data = df, aes(x = x, y = y)) + geom_bar(stat = "identity") +
    theme_minimal()
p
```



Exercise: Do the same, but with $p = 0.5$. Which value for p do you think will be best if you need to approximate the binomial with a normal distribution?

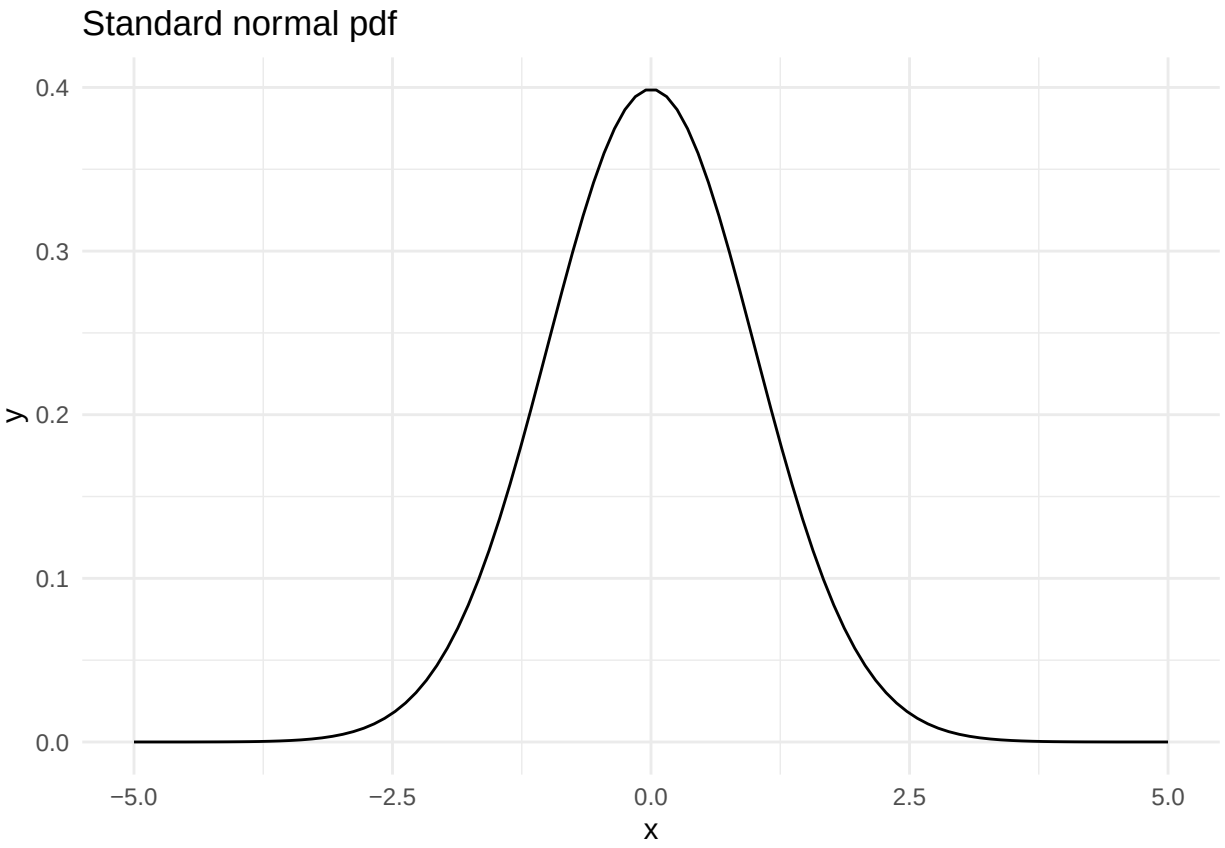
Solution: The approximation is best for $p = 0.5$. Then the probability mass function is symmetric, as is the nomral probability density function. The approximation gets worse the closer p is to 0 or 1.

Normal distribution

For the continuous distributions – like the normal distributions – the probability density function gives the probability of observing a value between a and b when we integrate the pdf from a to b: $P(a < X \leq b) = \int_a^b f(x)dx$. Remember: $f(x)$ does not give the point probability of x - the point probability is 0. Why?

We can plot the pdf for the normal distribution:

```
x = seq(from = -5, to = 5, length = 100)
y = dnorm(x = x, mean = 0, sd = 1)
df = data.frame(x = x, y = y)
p = ggplot(data = df, aes(x = x, y = y)) + geom_line() + ggtitle("Standard normal pdf") +
    theme_minimal()
p
```

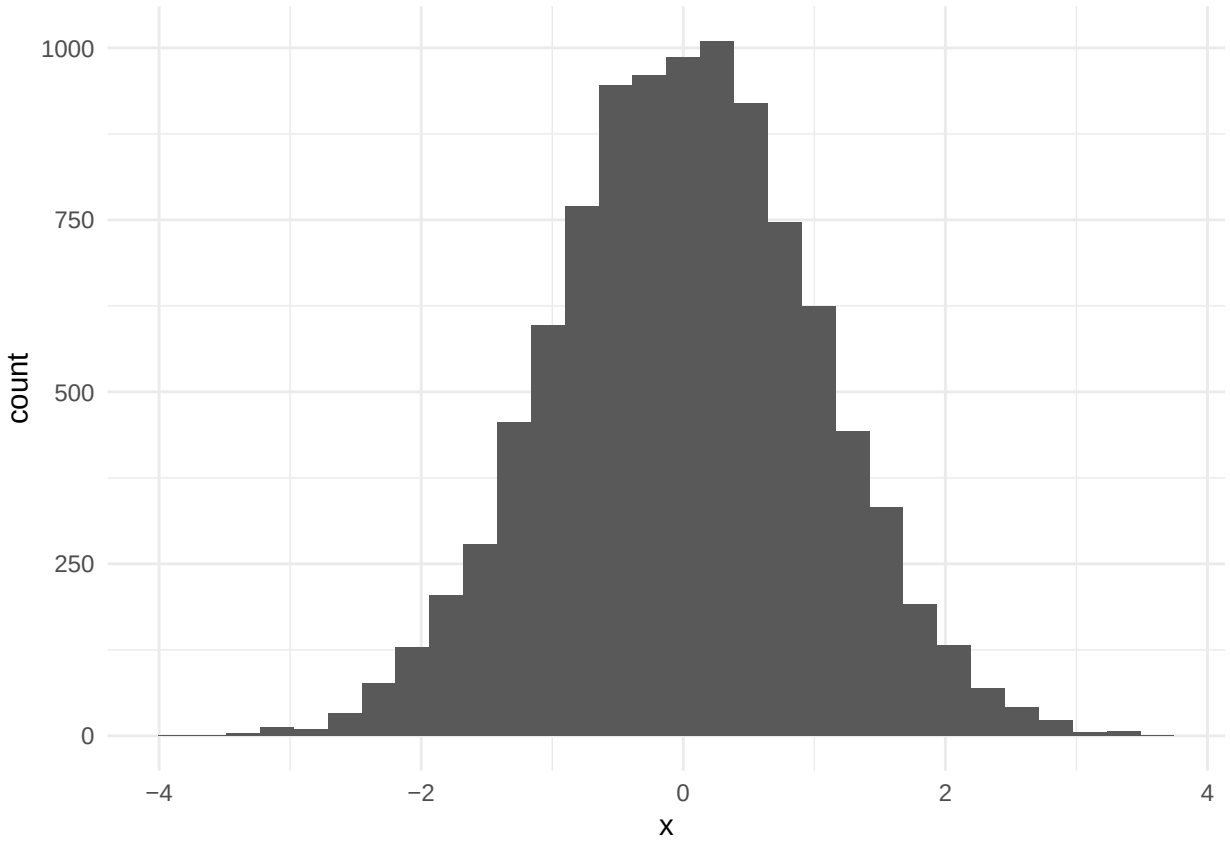


Exercise: Find $f(1.4)$ looking at the figure **and** using the `dnorm`. How can you interpret this value?

Solution: `dnorm(1.4)` gives `0.1497275`. For a small δ, $P(1.4-\delta < Z < 1.4+\delta) \approx 2 \cdot 0.15 \cdot \delta$ for a standard normally distributed Z.

We can also plot the distribution using randomly generated values for the distribution. The function `rnorm()` draws random values from the given distribution. Below we draw `n=10000` values and make a histogram.

```
n = 10000
x = rnorm(n = n, mean = 0, sd = 1)
p = ggplot(data.frame(x = x), aes(x)) + geom_histogram() + theme_minimal()
p
```



Try using smaller values of `n` and see what happends. Explain.

We can easily check the sample mean, variance and standard deviation of our drawn values by

```
mean(x)
var(x)
sd(x)
```

Exercise: Check your drawn values to see that the mean and variance agree with what you specified. (No solution provided.)

t-distribution

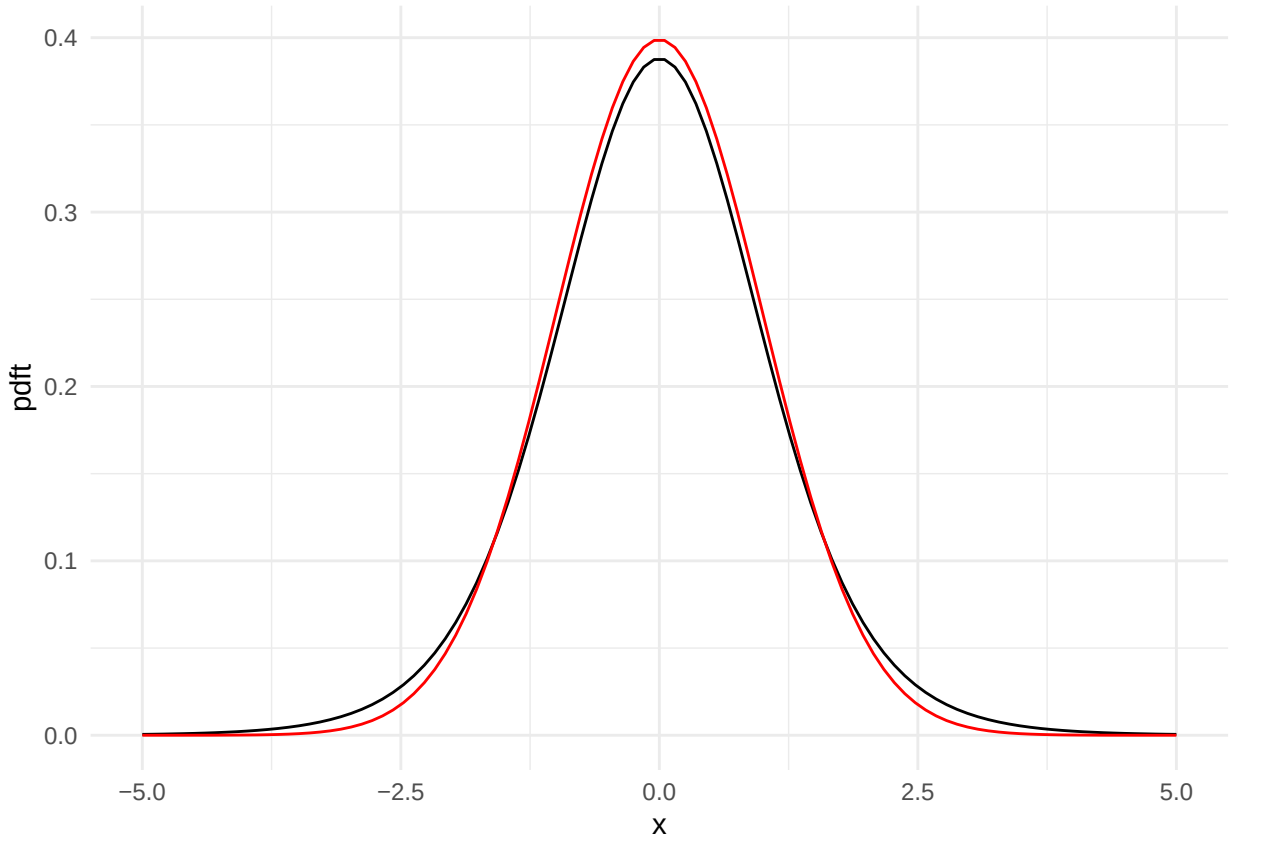
Let X_i be independent normal variables with mean μ and variance σ^2, and $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$. Further let

$$T = \frac{\bar{X} - \mu}{S/\sqrt{n}} \ .$$

We can plot the pdf for the t-distribution with $n-1$ degrees of freedom (here $n = 10$), and add the standard normal for reference. Is the red curve the t or normal curve? How can you see that?

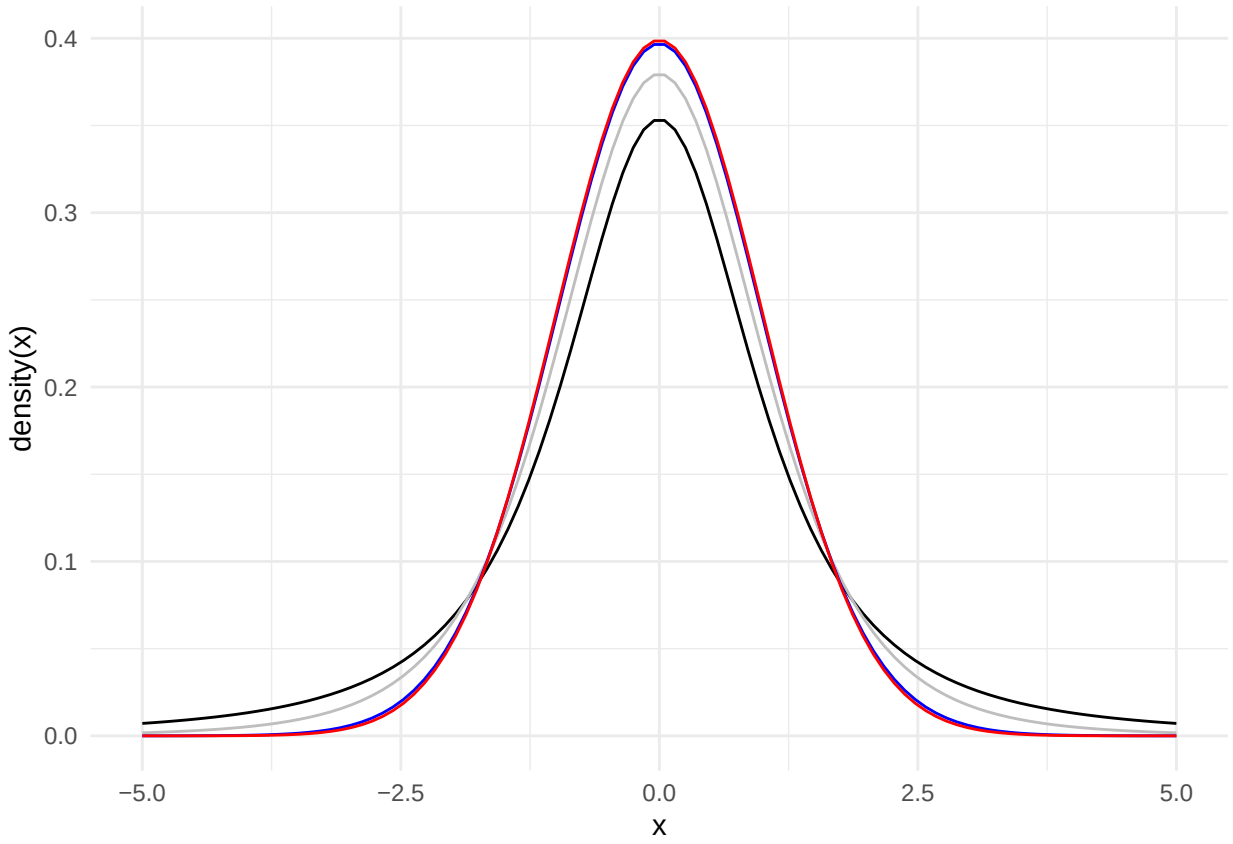
```
library(ggplot2)
n = 10
x = seq(from = -5, to = 5, length = 100)
pdft = dt(x = x, df = n - 1)
pdfn = dnorm(x)
df = data.frame(x = x, pdft = pdft, pdfn = pdfn)
p = ggplot(data = df, aes(x = x)) + geom_line(aes(x = x, y = pdft)) +
```

```
    geom_line(aes(x = x, y = pdfn), colour = "red") + theme_minimal()
p
```



We can also plot the pdf for the t-distribution with other degrees of freedom, for example `df=2`, `df=5` or `df=50`.

```
x = seq(from = -5, to = 5, length = 100)
pdft2 = dt(x = x, df = 2)
pdft5 = dt(x = x, df = 5)
pdft50 = dt(x = x, df = 50)
pdfn = dnorm(x)
df = data.frame(x = x, pdft2 = pdft2, pdft5 = pdft5, pdft50 = pdft50,
    pdfn = pdfn)
p = ggplot(data = df, aes(x = x)) + geom_line(aes(x = x, y = pdft2)) +
    geom_line(aes(x = x, y = pdft5), colour = "grey") + geom_line(aes(x = x,
    y = pdft50), colour = "blue") + geom_line(aes(x = x, y = pdfn), colour = "red") +
    theme_minimal() + labs(y = "density(x)")
p
```



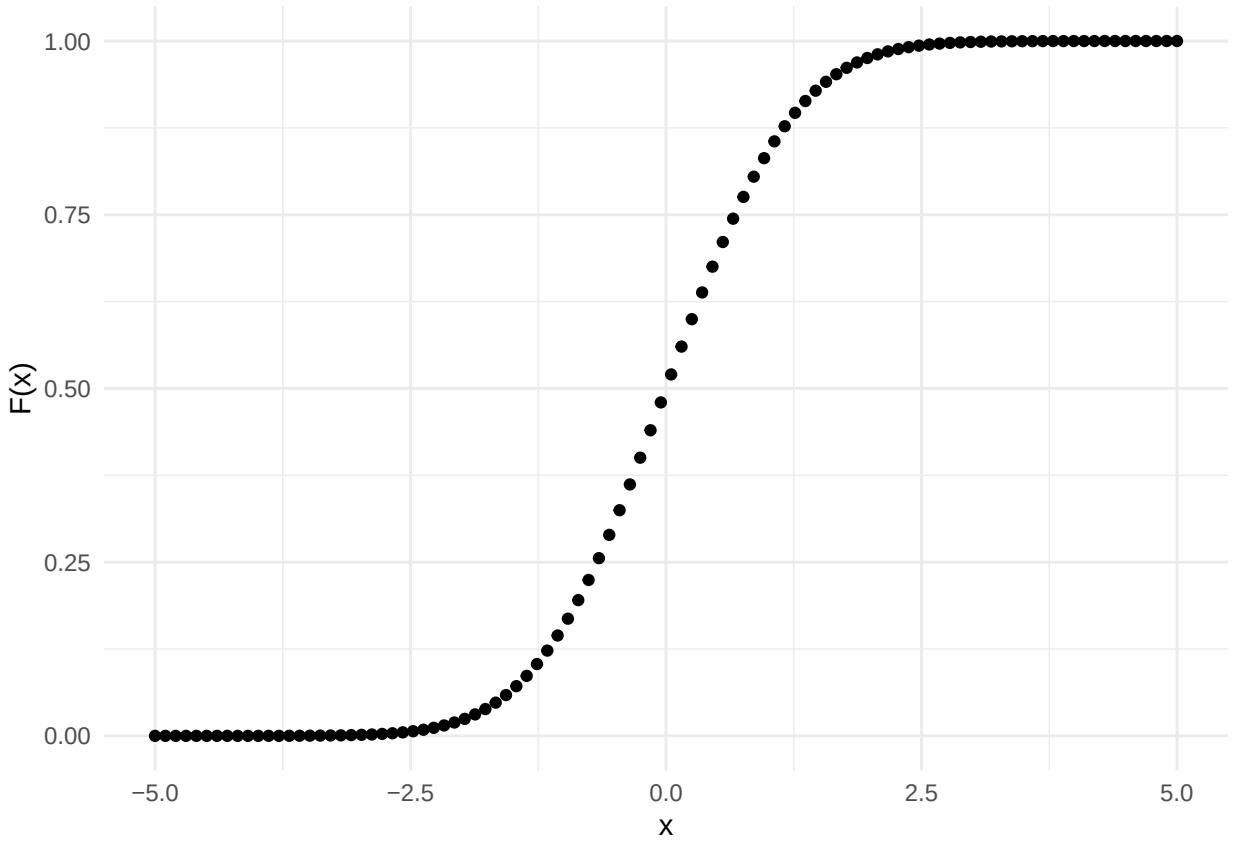
Which graph represents the different degrees of freedom, and what happens when the degrees of freedom increase?

Cumulative distribution function $F(x)$

The cumulative distribution function (cdf) is the probability that your observation is less or equal to x,

$$F(x) = P(X \leq x) = \begin{cases} \int_{-\infty}^{x} f_x(t)dt & \text{for continuous distributions} \\ \sum_{t \leq x} f(t) & \text{for discrete distributions} \end{cases}$$

The following plot shows the cdf of a standard normal distribution.



Critical values and significance level

To find the significance level and critical values, `pnorm()` (cdf-function) and `qnorm()` (quantile function - inverse cdf) are helpful functions. For example, if we want to find the critical values of a standard normal distribution with significance level $\alpha = 0.05$, we can use the quantile function

```
z_l = qnorm(0.025, mean = 0, sd = 1)  #lower
z_u = qnorm(0.975, mean = 0, sd = 1)  #upper
c(z_l, z_u)
```

```
## [1] -1.959964  1.959964
```

Using the lower and upper quantiles (`z_l` and `z_u`), we can invert the procedure and find the significance level (cdf of the critical values, $P(Z > z_{\alpha/2}) = \alpha/2$ and $P(Z < z_{1-\alpha/2}) = \alpha/2$). Instead of integrating, we can use the cumulative distribution function `pnorm()`. For a one-sided test we would obtain

```
alpha_lower = pnorm(q = z_l, mean = 0, sd = 1, lower.tail = TRUE)
alpha_upper = 1 - pnorm(q = z_u, mean = 0, sd = 1, lower.tail = TRUE)
c(alpha_lower, alpha_upper)
```

```
## [1] 0.025 0.025
```

Note that for the upper tail, we could have written `alpha_upper = pnorm(q = q, mean = 0, sd = 1, lower.tail = FALSE)`. The logical option `lower.tail` indicate if you want to calculate the upper, $P(X > x)$, or the lower, $P(X \leq x)$, tail. The default is `TRUE`, meaning that if you don't include this option as input it will automatically calculate the lower tail.

Exercise Find the value of the cdf for $x = -5$ and $x = 5$ from looking at the figure **and** using the distribution function in R.

Solutions: `pnorm(-5)` and `pnorm(5)` give `2.866516e-07` and `0.9999997`, respectively, i.e. close to 0 and 1. This agrees with the figure of the cdf.

Exercise Assume a student t-distribution with 1 degree of freedom, as in the figure below. Use the distribution function to

1. Calculate the cdf of $x = 1.8$, $F_x(1.8) = \int_{-\infty}^{1.8} f_x(x)dx$.
2. Calculate $P(0 \leq x \leq 2)$.
3. Find z_α when $P(X > z_\alpha) = 0.1$.

Solution: We can use the following code:

```
pt(q = 1.8, df = 1)
```

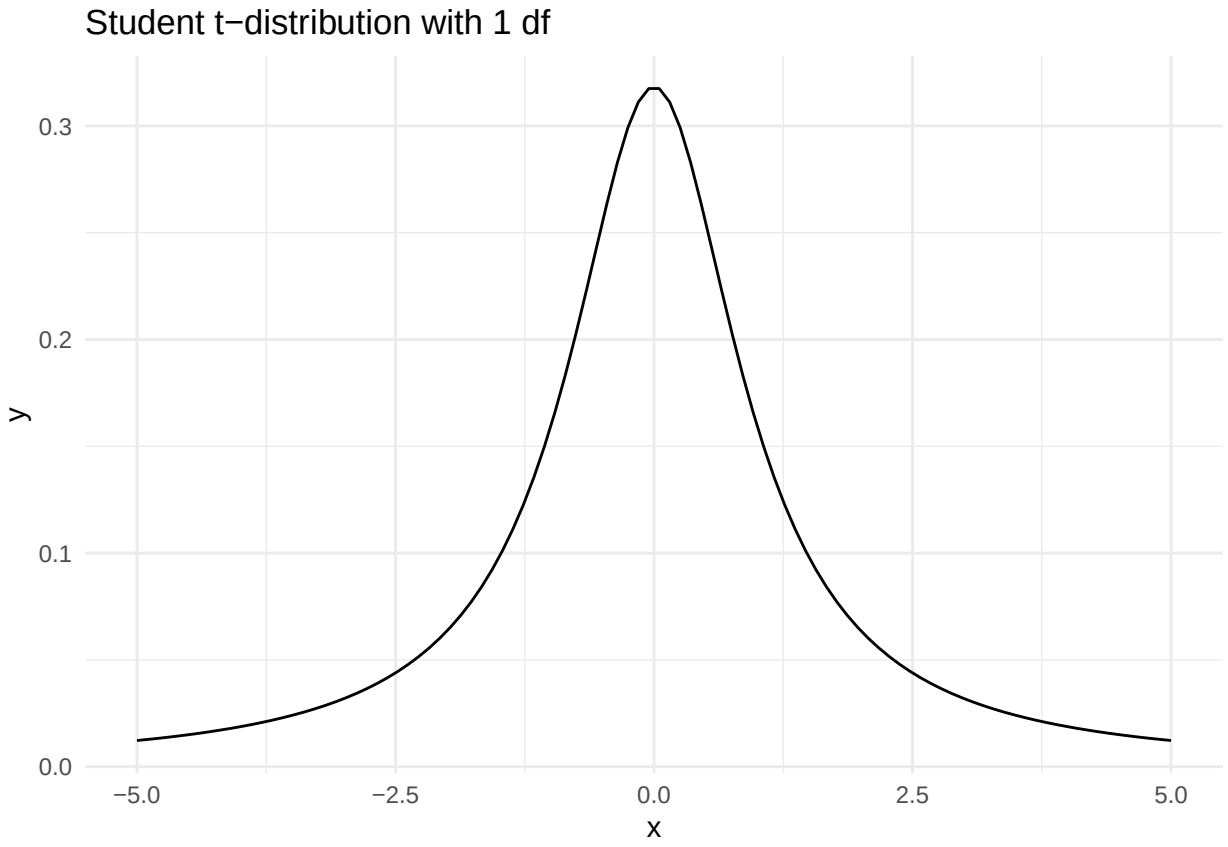
```
## [1] 0.8385855
```

```
pt(q = 2, df = 1) - pt(q = 1, df = 1)
```

```
## [1] 0.1024164
```

```
qt(p = 0.1, df = 1, lower.tail = F)
```

```
## [1] 3.077684
```



Writing a simple Z-test as a function

Let us assume that X is a random variable from a normal distribution with know variance σ^2 and we want to perform a hypothesis test to test that the mean of X is not equal to 0, $\mu \neq 0$, that is, a two-sided test.

Assume that have collected a random sample $(X_1, X_2, \ldots, X_n)$, and that we may assume that $\frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$ follows a standard normal distribution (and σ^2 is known). Here $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. A t-test is implemented in R (function `t.test`), but we will now implement our own z-test.

To this end, we need a function with the data and given (known) standard deviation as input. We assume that the data are given in a vector named `x`.

Read and discuss what is done here.

```
myz.test <- function(x, sd) {
    n <- length(x)
    xbar <- mean(x)
    zobs <- xbar/(sd/sqrt(n))
    # if you what to print remove hashtag in next line
    # print(paste('Zobs',zobs))
    pval <- 2 * pnorm(abs(zobs), lower.tail = FALSE)
    # if you what to print remove hashtag in next line
    # print(paste('P-value',pval))
    return(list(statistic = zobs, p.value = pval))
}
```

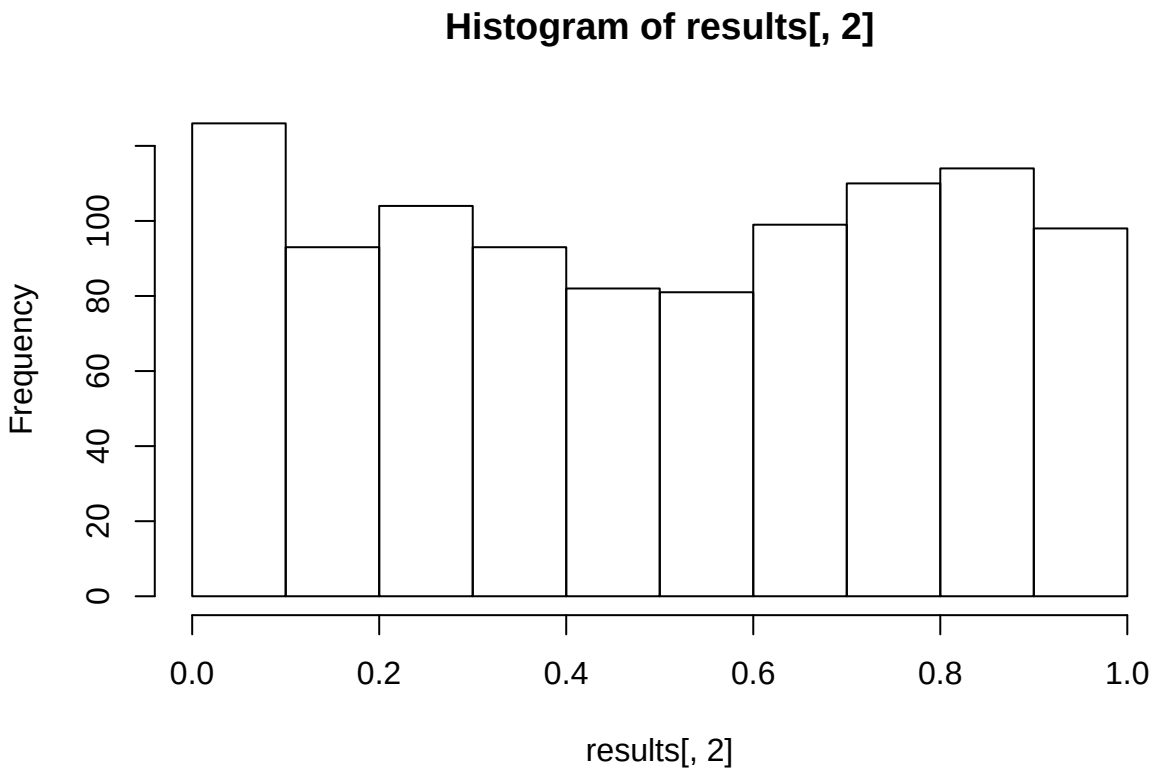
To use the function we first generate some data and then call the function. To ensure that we all get the same results, we also set a seed first.

```
set.seed(23422)
testds <- rnorm(100, mean = 0.5, sd = 6)
myz.test(testds, sd = 6)
```

```
## $statistic
## [1] 1.150936
##
## $p.value
## [1] 0.2497588
```

If you want to run this function for many simulated data sets and put the results from `myz.test` into a matrix you may use the following simple for-loop:

```
mu = 0
sigma = 4
n = 10  # size of each data set
B = 1000  # number of data sets to simulate
results <- matrix(NA, ncol = 2, nrow = B)
for (i in 1:B) {
    testX <- rnorm(n, mean = mu, sd = sigma)
    thisresult <- myz.test(testX, sigma)
    results[i, ] <- c(thisresult$statistic, thisresult$p.value)
}
hist(results[, 2])
```



Exercise: Run the code on your computer. What is done here? What does the histogram display? Hint: what is the distribution of p-values from true null hypotheses?

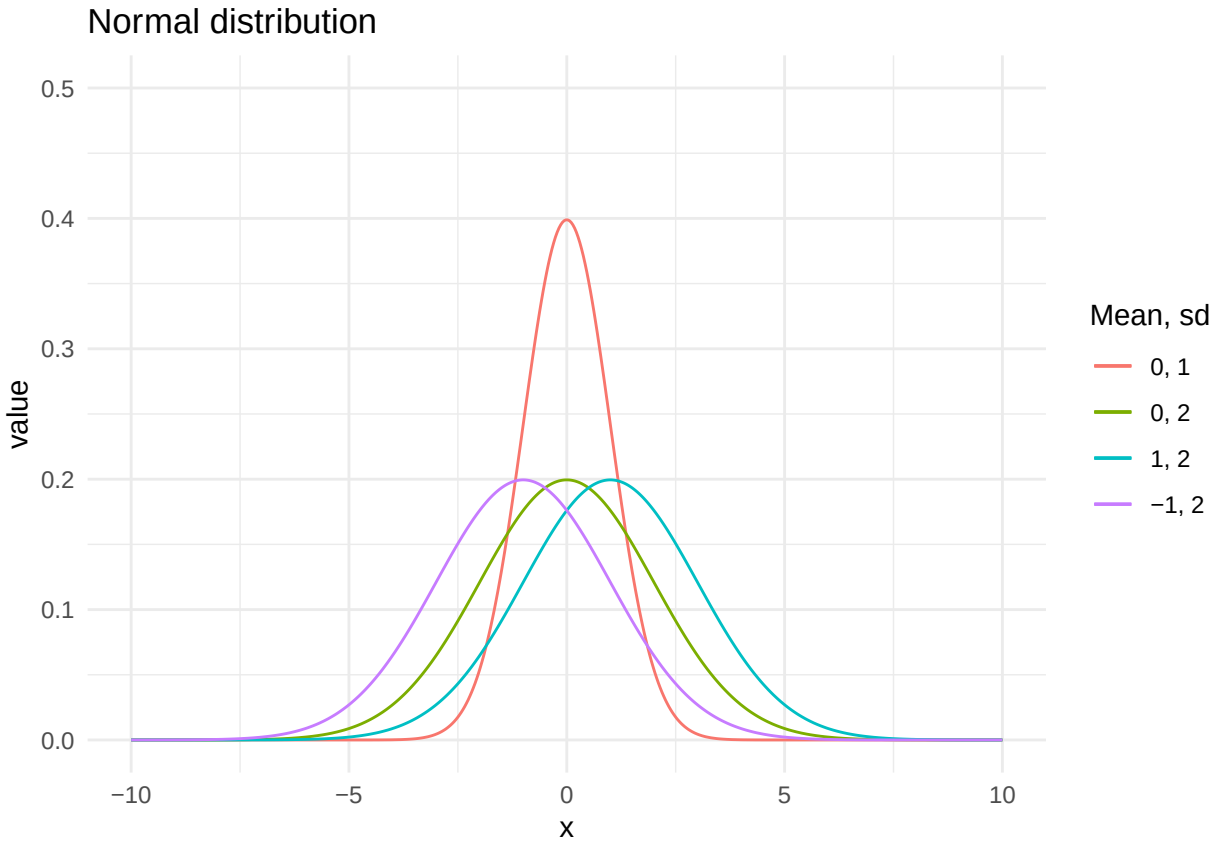
Solution: The test $H_0 : \mu = 0$ versus $H_1 : \mu \neq 0$ is performed 1000 times. The histogram (above) displays the p-values. They seem uniformly distributed, as p-values from a true simple nullhypothis should be with continuous data.

Plotting pdfs

For the user who wants to do a bit more advanced graphics: Below is an example of how to draw multiple plots using `ggplot`.

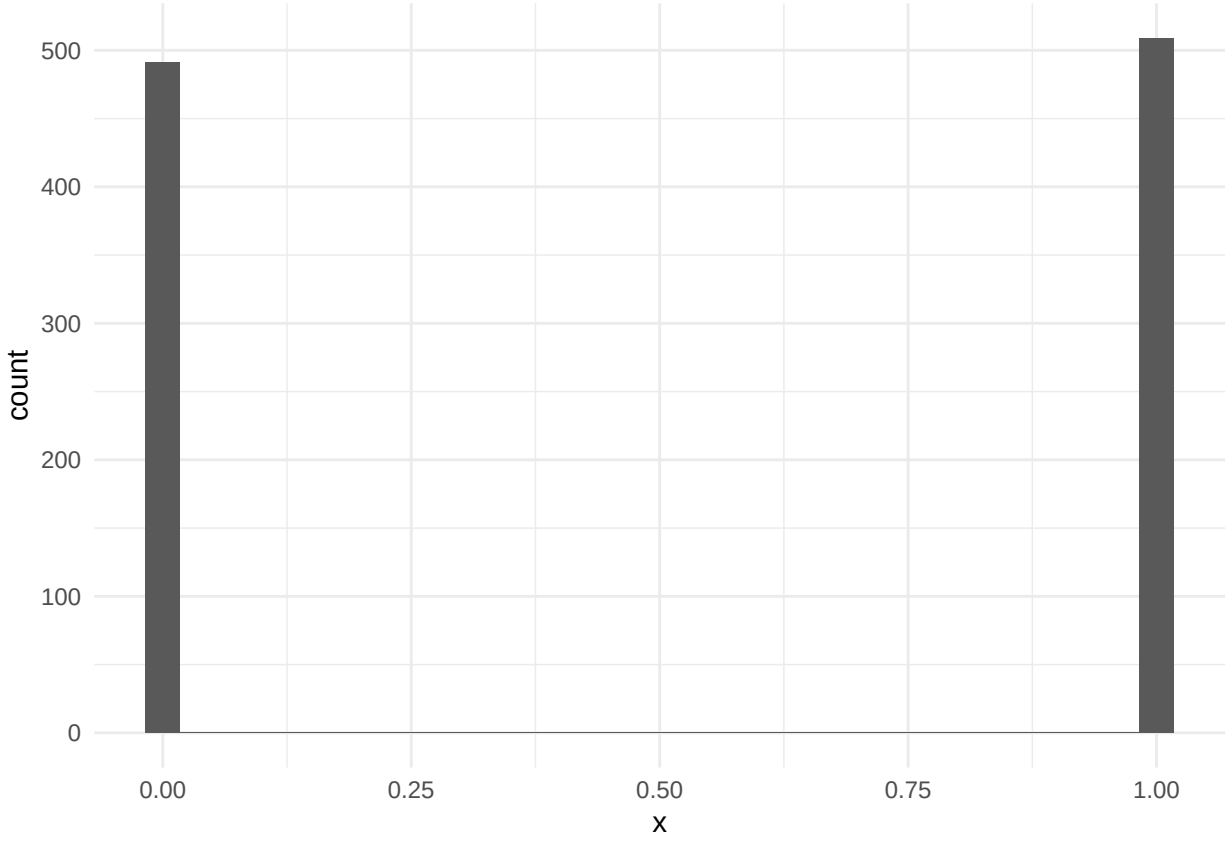
Exercise: What is this plot showing?

```
x = seq(from = -10, to = 10, length = 500)
y1 = dnorm(x, mean = 0, sd = 1)
y2 = dnorm(x, mean = 0, sd = 2)
y3 = dnorm(x, mean = 1, sd = 2)
y4 = dnorm(x, mean = -1, sd = 2)
df = data.frame(x, y1, y2, y3, y4)
ggplot(df, aes(x, y = value, color = variable)) + geom_line(aes(y = y1,
    col = "y1")) + geom_line(aes(y = y2, col = "y2")) + geom_line(aes(y = y3,
    col = "y3")) + geom_line(aes(y = y4, col = "y4")) + ylim(0, 0.5) +
    ggtitle("Normal distribution") + scale_colour_discrete(name = "Mean, sd",
    breaks = c("y1", "y2", "y3", "y4"), labels = c("0, 1", "0, 2", "1, 2 ",
        "-1, 2")) + theme_minimal()
```



Let's assume we flip a coin where the probability of getting head is $p = P(head) = P(X = 1) = 0.5$, and the probabiltiy of getting tails is $1 - p = P(tails) = P(X = 0)$. The following shows the number of times we get head and tails of a throw when we try 1000 times.

```
n = 1000
p = 0.5
x = rbinom(n, size = 1, prob = p)
df = data.frame(x = x)
p = ggplot(df, aes(x = x)) + geom_histogram() + theme_minimal()
p
```



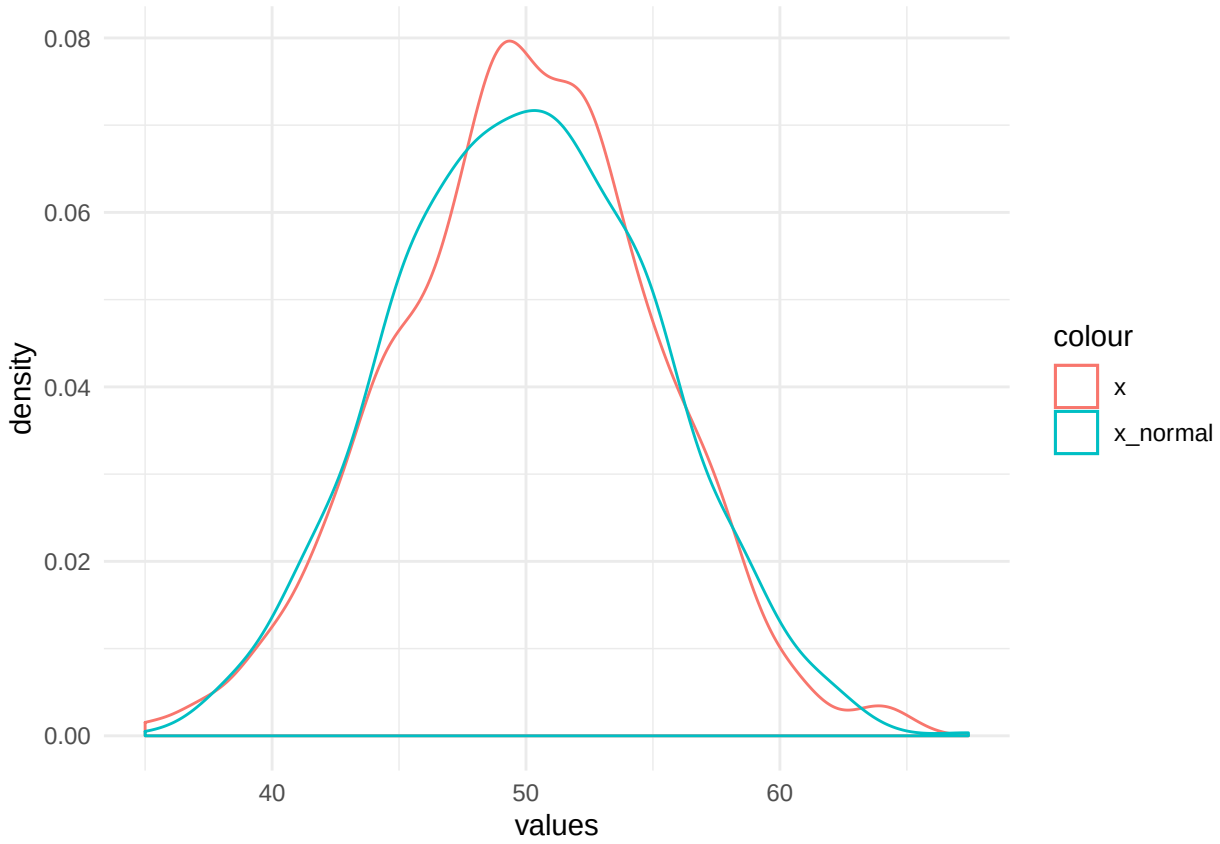
Note here that the option `size` refers the number of trials.

Exercise: Increase the number of trials to 100 when holding the other parameters constant. Can you explain what happens here and why? What is the mean and variance of this distribution? What is the probability of having only one head when you do 100 trials?

Solution: An experiment of flipping a coin 100 times is repeated 1000 times. Each of the 1000 results will be an integer between 0 and 1000 (inclusive). The number of heads in each experiment has the binomial distribution with parameters $n = 100$ and $p = 0.5$. The mean of this distribution is $np = 50$ and the variance is $np(1-p) = 25$. We find the probability of having one or zero heads by `pbinom(1, size=100, prob=0.5)`, which gives `7.967495e-29`, and the probability of having exactly one head by `dbinom(1,100,.5)`, which gives `7.888609e-29`, i.e., very small probabilities.

The R code (below) draws a kernel density estimate (a smoothed version of the histogram) of the number of heads in the 1000 experiments (red). The same is done for a normal approximation to the binomial distribution (blue).

```
n = 1000
p = 0.5
x = rbinom(n, size = 100, prob = 0.5)
x_normal = rnorm(n, mean = 100 * p, sd = sqrt(100 * p * (1 - p)))
# hist(x) hist(x_normal)
df = data.frame(x = x, x_normal = x_normal)
ggplot(data = df, aes(x = values), color = variable) + geom_density(aes(x = x,
    col = "x")) + geom_density(aes(x = x_normal, col = "x_normal")) +
    theme_minimal()
```

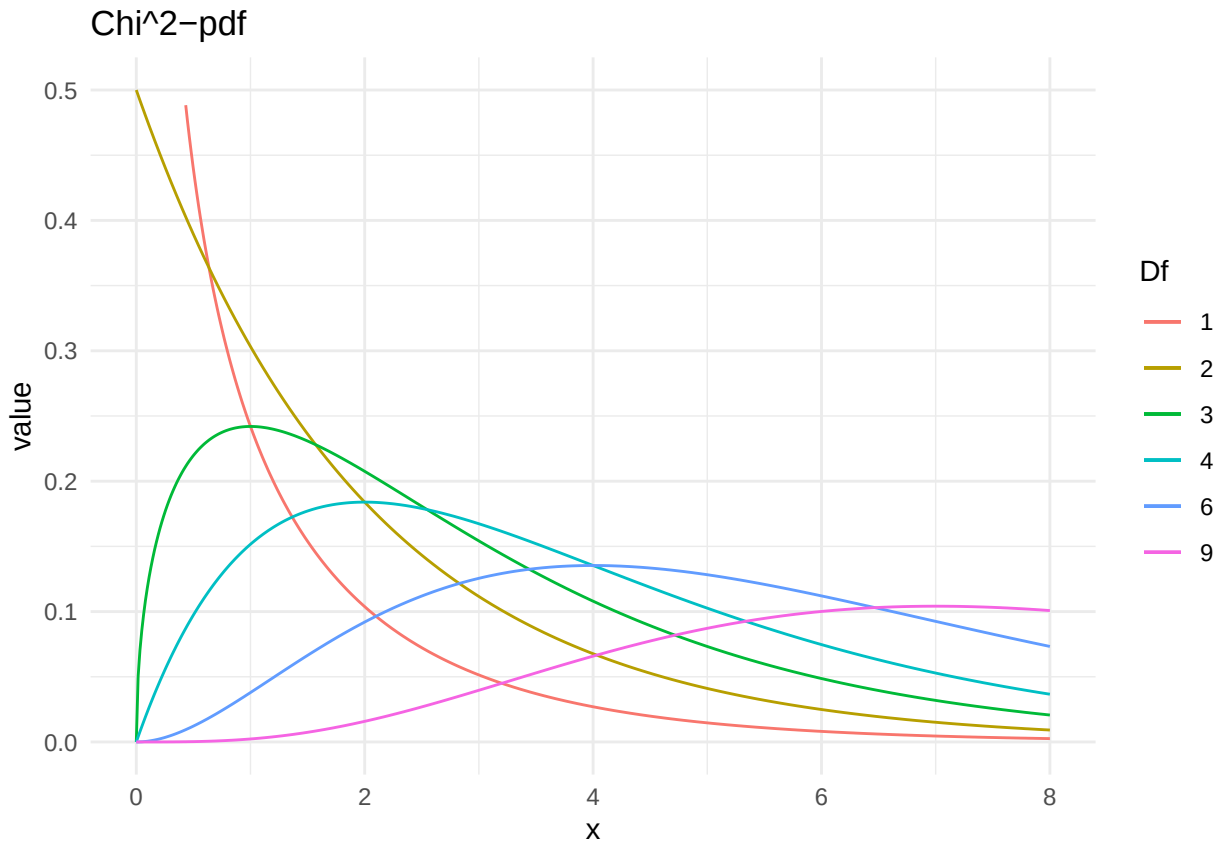


Exercise: Make one plot of the pdf and one plot of the cdf for each of the following distributions

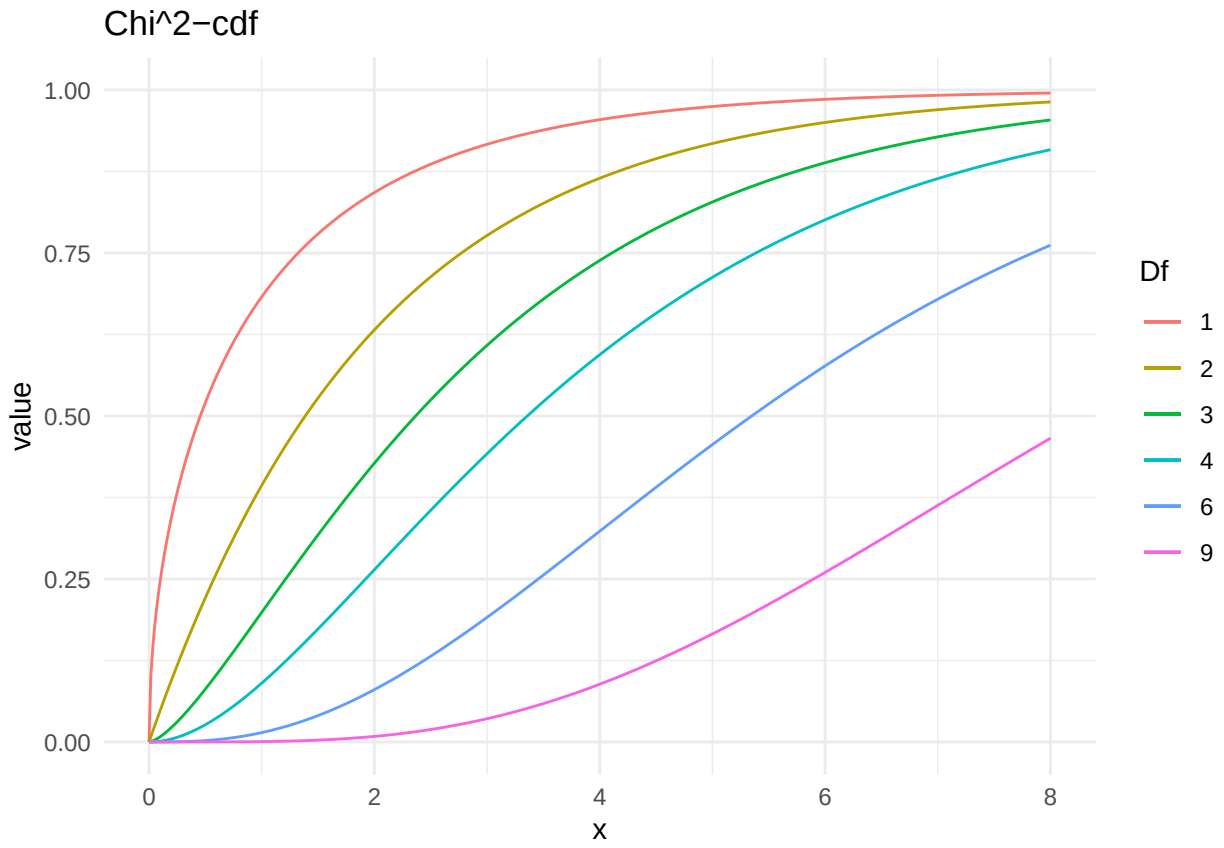
1. χ^2- distributions with 1, 2, 3, 4, 6 and 9 degrees of freedom
2. Student t-distribution with 1, 2, 3, 4 and 5 degrees of freedom.

Solution: R code and plots:

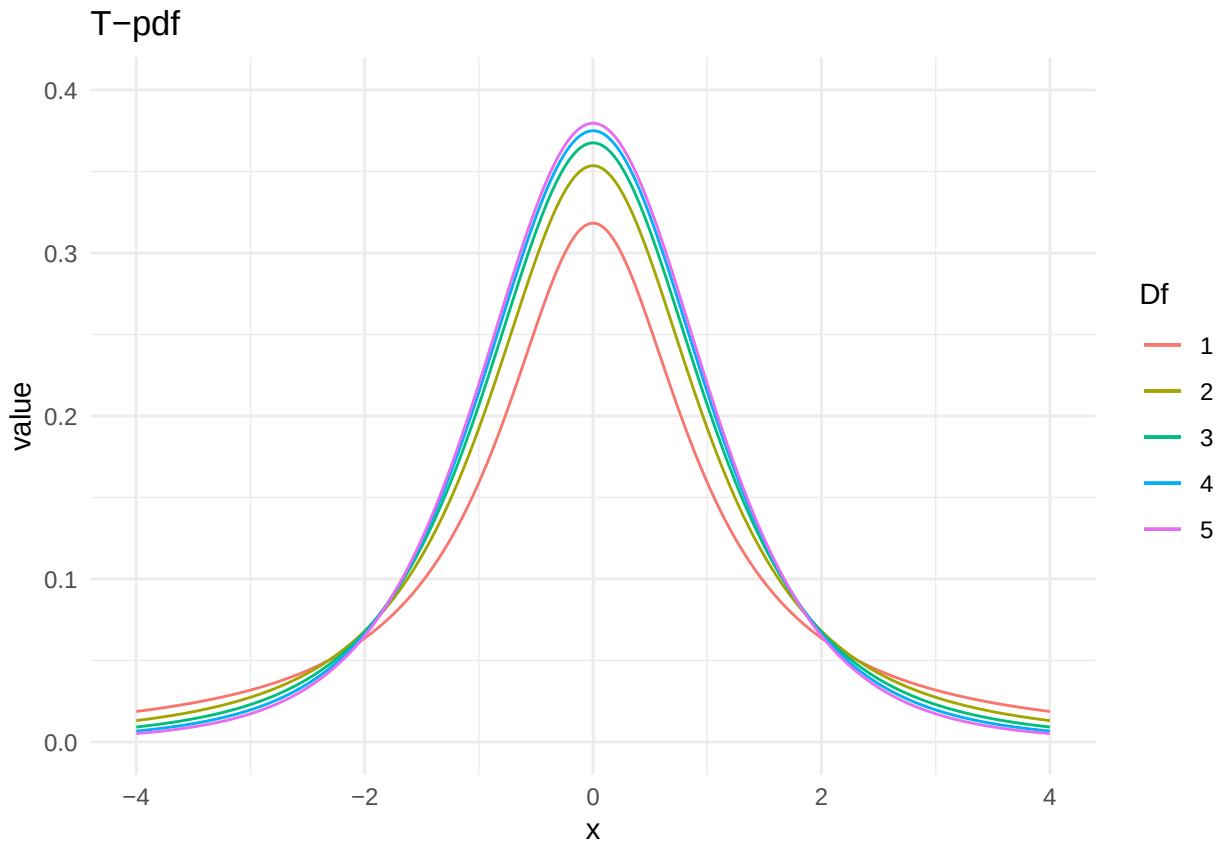
```
x = seq(from = 0, to = 8, length = 500)
y1 = dchisq(x, df = 1)
y2 = dchisq(x, df = 2)
y3 = dchisq(x, df = 3)
y4 = dchisq(x, df = 4)
y6 = dchisq(x, df = 6)
y9 = dchisq(x, df = 9)
df = data.frame(x, y1, y2, y3, y4, y6, y9)
ggplot(df, aes(x, y = value, color = variable)) + geom_line(aes(y = y1,
    col = "y1")) + geom_line(aes(y = y2, col = "y2")) + geom_line(aes(y = y3,
    col = "y3")) + geom_line(aes(y = y4, col = "y4")) + geom_line(aes(y = y6,
    col = "y6")) + geom_line(aes(y = y9, col = "y9")) + ylim(0, 0.5) +
    theme_minimal() + ggtitle("Chi^2-pdf") + scale_colour_discrete(name = "Df",
    breaks = c("y1", "y2", "y3", "y4", "y6", "y9"), labels = c("1", "2",
        "3", "4", "6", "9"))
```



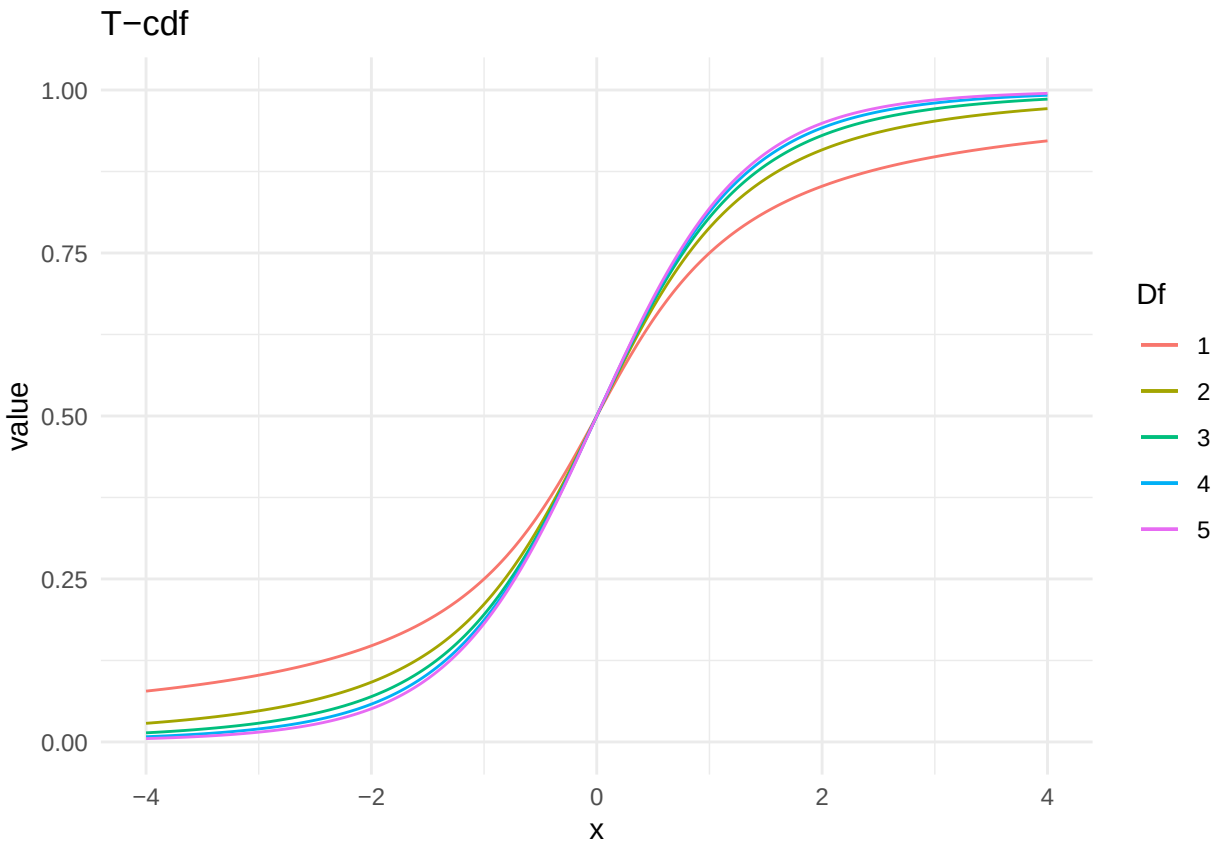

```
x = seq(from = 0, to = 8, length = 500)
y1 = pchisq(x, df = 1)
y2 = pchisq(x, df = 2)
y3 = pchisq(x, df = 3)
y4 = pchisq(x, df = 4)
y6 = pchisq(x, df = 6)
y9 = pchisq(x, df = 9)
df = data.frame(x, y1, y2, y3, y4, y6, y9)
ggplot(df, aes(x, y = value, color = variable)) + geom_line(aes(y = y1,
    col = "y1")) + geom_line(aes(y = y2, col = "y2")) + geom_line(aes(y = y3,
    col = "y3")) + geom_line(aes(y = y4, col = "y4")) + geom_line(aes(y = y6,
    col = "y6")) + geom_line(aes(y = y9, col = "y9")) + ylim(0, 1) +
    ggtitle("Chi^2-cdf") + theme_minimal() + scale_colour_discrete(name = "Df",
    breaks = c("y1", "y2", "y3", "y4", "y6", "y9"), labels = c("1", "2",
        "3", "4", "6", "9"))
```




```
x = seq(from = -4, to = 4, length = 500)
y1 = dt(x, df = 1)
y2 = dt(x, df = 2)
y3 = dt(x, df = 3)
y4 = dt(x, df = 4)
y5 = dt(x, df = 5)
df = data.frame(x, y1, y2, y3, y4, y5)
ggplot(df, aes(x, y = value, color = variable)) + geom_line(aes(y = y1,
    col = "y1")) + geom_line(aes(y = y2, col = "y2")) + geom_line(aes(y = y3,
    col = "y3")) + geom_line(aes(y = y4, col = "y4")) + geom_line(aes(y = y5,
    col = "y5")) + ylim(0, 0.4) + ggtitle("T-pdf") + scale_colour_discrete(name = "Df",
    breaks = c("y1", "y2", "y3", "y4", "y5"), labels = c("1", "2", "3",
        "4", "5")) + theme_minimal()
```




```
x = seq(from = -4, to = 4, length = 500)
y1 = pt(x, df = 1)
y2 = pt(x, df = 2)
y3 = pt(x, df = 3)
y4 = pt(x, df = 4)
y6 = pt(x, df = 5)
df = data.frame(x, y1, y2, y3, y4, y5)
ggplot(df, aes(x, y = value, color = variable)) + geom_line(aes(y = y1,
    col = "y1")) + geom_line(aes(y = y2, col = "y2")) + geom_line(aes(y = y3,
    col = "y3")) + geom_line(aes(y = y4, col = "y4")) + geom_line(aes(y = y6,
    col = "y5")) + ylim(0, 1) + ggtitle("T-cdf") + scale_colour_discrete(name = "Df",
    breaks = c("y1", "y2", "y3", "y4", "y5"), labels = c("1", "2", "3",
        "4", "5")) + theme_minimal()
```



Here is more about using `ggplot` to plot functions in general: http://t-redactyl.io/blog/2016/03/creating-plots-in-r-using-ggplot2-part-9-function-plots.html. Next, you can go to Visualizations in R

Acknowledgements

We thank Øyvind Bakke, Mette Langaas and her PhD students from 2018 and 2019 for building up the original version of this sheet.