Non-Parametric Statistics - Problem Sets

Silvana, Florencia Luque, Simon Schmetz

2025-03-14

Introduction

In the following document, the solutions to three problem sets from Prof. Eduardo García-Portugués book on nonparametric Statistics (https://bookdown.org/egarpor/NP-UC3M/) as final assignment for the Course in Nonparametric Statistics at Universidad Carlos III de Madrid. *Contributions made to this document are made are equal among each of the three authors*

```r
# general libraries
library(ggplot2)
```

Excercise 4.21

```r
#TODO: Silvana
```

Excercises 5.10

Investigate the accuracy of the naive bootstrap confidence intervals implemented in *np::npplot*. To do so:

1. Simulate $M = 500$ samples of size $n = 100$ from the regression model $Y = m(X) + \varepsilon$, where $m(x) = 0.25x^2 - 0.75x + 3$, $X \sim N(0, 1.5^2)$, and $\varepsilon \sim N(0, 0.75^2)$.

```r
library(np)
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-18)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```r
set.seed(1234)
M = 500
n = 100
y_muestras = matrix(0,ncol = n,nrow = M)
m = function(a){
  0.25*a^2-0.75*a+3
}
for (i in 1:500){
  x = rnorm(100,0,1.5)
  epsilon = rnorm(100,0,0.75)
  y_muestras[i,] = m(x)+epsilon
}
```

2. Compute the 95% confidence intervals for $m(x)$ along $x \leftarrow seq(-5, 5, by = 0.1)$, for each of the M samples. Do it for the normal approximation and quantile-based confidence intervals.

```r
x_grid = seq(-5,5,by=0.1)
```

3. Check if $m(x)$ belongs to each of the confidence intervals, for each x.
4. Approximate the actual coverage of the confidence intervals.

Excercise 6.8

The following exercise revolves around the comparison of power between the three nonparametric normality tests:

1. Kolmogorov-Smirnov
2. Anderson-Darling
3. Cramer von Mises tests.

with the test hypothesis:

$H_0 : X \sim N(0,1)$ vs $H_1 : X \nsim N(0,1)$

The comparison is performed by performing these tests M times for samples of size n from a chosen normal or t-student's and comparing the distribution of the resulting M p-values the corresponding rejection rate for significance level α.

```r
# Load Libraries Required for Excercises
library(goftest)
library(nortest)
```

```
##
## Attaching package: 'nortest'

## The following objects are masked from 'package:goftest':
##
##     ad.test, cvm.test
```

```r
library(latex2exp)
```

We begin by setting up a versatile function that performs three steps:

1. Draw M samples of size n from either a normal or t-student's distribution, perform the three nonparametric tests and store the resulting M p-values.
2. For the three normality tests with , calculate the estimated rejection rate (Power of the test) for significance level α.

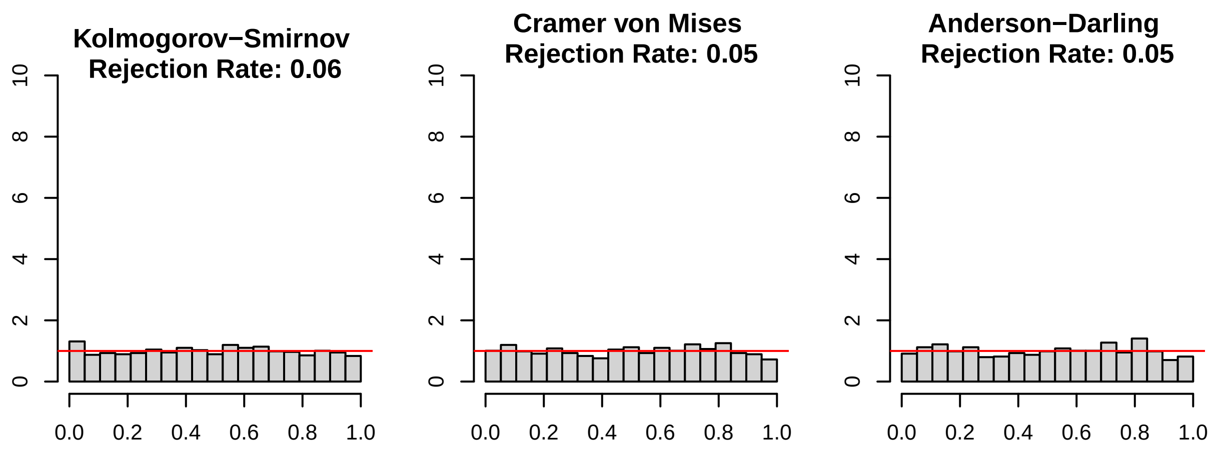
$$\text{Rejection Rate} = \frac{\text{Number of rejections of } H_0}{\text{Total number of tests}}$$

3. Plot the distribution of the M p-values for each test.

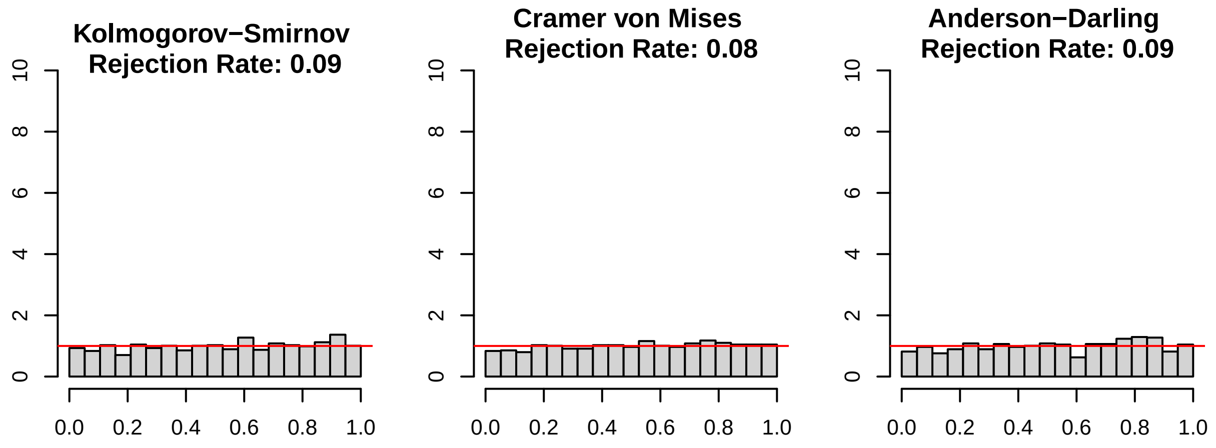
```r
# Function for Simulation of p-values
simulate_pvalues <- function(mu_samp=0, sd_samp=1 , M, n, alpha, dist = "normal") {

  # generate p-Values for samples from Normal
  pvalues <- sapply(1:M, function(i) {

    if(dist == "normal"){
      x = rnorm(n, mean = mu_samp, sd = sd_samp)
    }

    if(dist == "t-students"){
      x = rt(n, df = 10, ncp = mu_samp)
    }

    pval_ks = ks.test(x, "pnorm", mean = 0, sd = 1)$p.value
    pval_cvm = goftest::cvm.test(x = x, null = "pnorm")$p.value
    pval_ad = goftest::ad.test(x = x, null = "pnorm")$p.value
    return(c(pval_ks, pval_cvm, pval_ad))
  })

  # get rejection rates
  ks_rejection_rate = sum(pvalues[1,] < alpha) / M
  cvm_rejection_rate = sum(pvalues[2,] < alpha) / M
  ad_rejection_rate = sum(pvalues[3,] < alpha) / M

  # plot
  par(mfrow = c(2, 3), mar = c(2, 2, 2, 2))  # Increase margins

  hist(pvalues[1,],
       breaks = seq(0, 1, l = 20),
       probability = TRUE,
       main = paste("\nKolmogorov-Smirnov", "\nRejection Rate:", round(ks_rejection_rate, 2)),
       ylim = c(0, 10),
       ylab = "p-value")
  abline(h = 1, col = "red")

  hist(pvalues[2,],
       breaks = seq(0, 1, l = 20),
       probability = TRUE,
       main = paste("Cramer von Mises", "\nRejection Rate:", round(cvm_rejection_rate, 2)),
       ylim = c(0, 10),
       ylab = "p-value")
  abline(h = 1, col = "red")

  hist(pvalues[3,],
       breaks = seq(0, 1, l = 20),
       probability = TRUE,
       main = paste("Anderson-Darling", "\nRejection Rate:", round(ad_rejection_rate, 2)),
       ylim = c(0, 10),
       ylab = "p-value")
  abline(h = 1, col = "red")
}
```

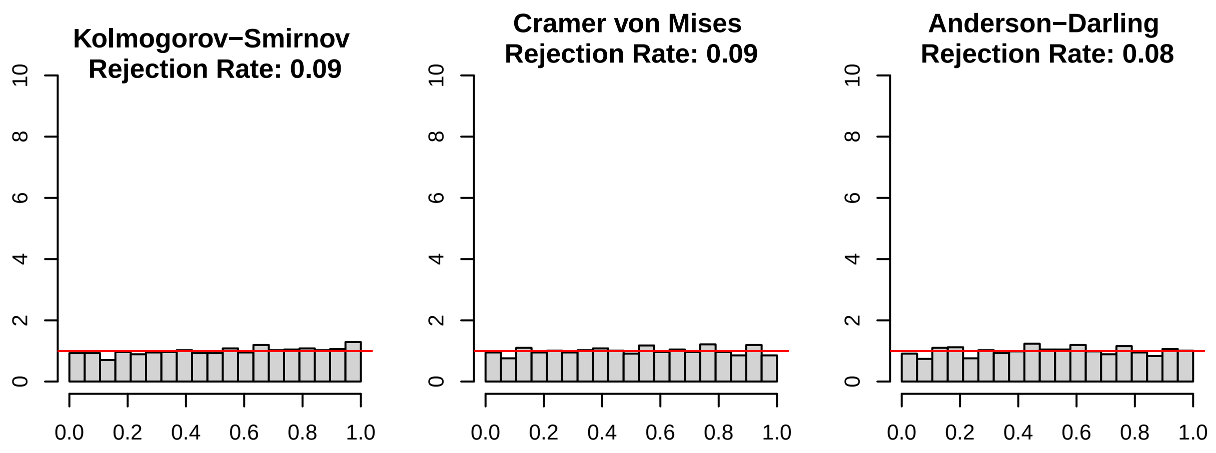
With the function in place, we can now simulate the p-values for the three tests for samples drawn from a normal distribution $N(0,1)$ for different sample sizes n and significance levels α. With the sample drawn from a normal distribution, we expect the three test to overwhelmingly not reject the null Hypothesis $H \in F_N$ and the distribution of the p-Values to be a Uniform $U(0,1)$. In fact, for both $n = 25$ and $n = 100$ and as well as $\alpha = 0.05$ and $\alpha = 0.1$, theses expectations are met, with the rejection rates corresponding to the chosen significance levels.

```
# H0 True
set.seed(123)
simulate_pvalues(mu_samp = 0, sd_samp = 1, M = 1000, n = 25, alpha = 0.05)
simulate_pvalues(mu_samp = 0, sd_samp = 1, M = 1000, n = 25, alpha = 0.1)
```



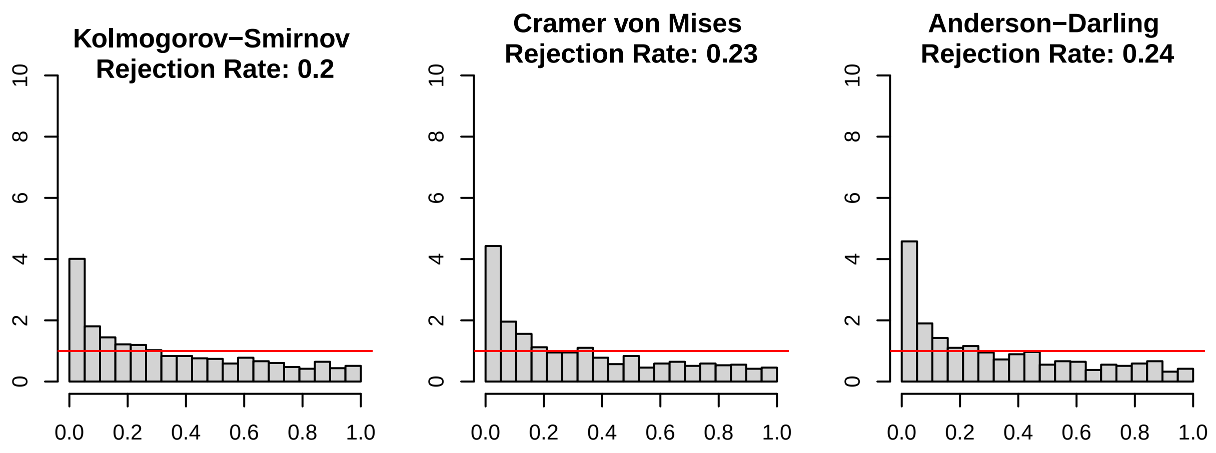
```r
simulate_pvalues(mu_samp = 0, sd_samp = 1, M = 1000, n = 100, alpha = 0.1)
```


Kolmogorov−Smirnov
Rejection Rate: 0.09
Cramer von Mises
Rejection Rate: 0.08
Anderson−Darling
Rejection Rate: 0.09
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

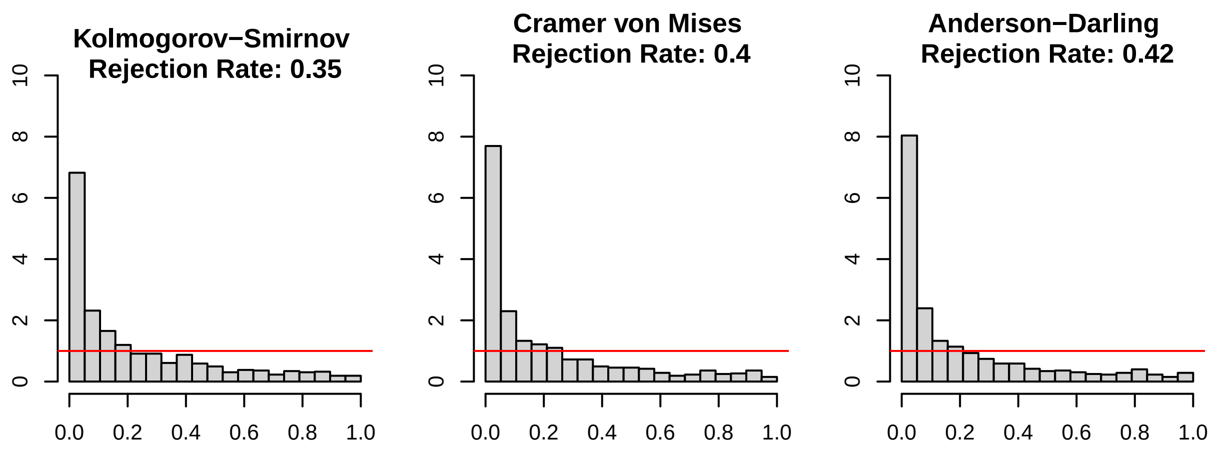


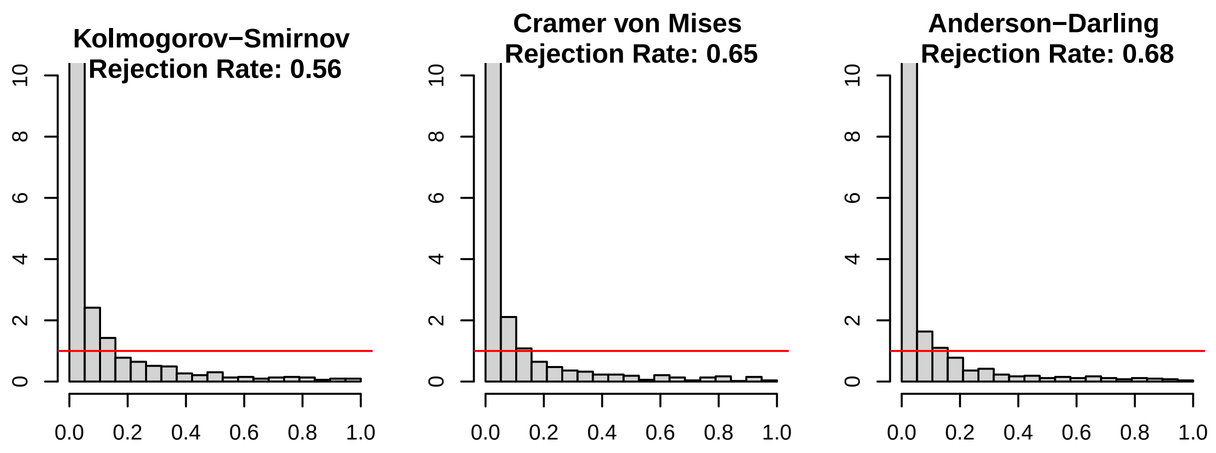
Performing the test for a sample drawn from a normal distribution (μ, σ) where $\mu \neq 0$, $\sigma \neq 1$, again for different sample sizes and a constant significance level $\alpha = 0.05$, results are more mixed. With the sample not drawn from a $N(0,1)$ we expect rejection to happen at a greater rate compared to the previous results. In addition, we expect both Cramer von Mises and Anderson Darling to deliver greater rejections rates as they are in general terms more powerful than the Kolomogorov-Smirnov. This is in line with the results acquired from the three parameter combinations as can be seen in the shown rejection rates . Furthermore, increasing the difference of μ from 0, appears to increase rejection rates more than an increase in sample size.

```
# H0 False
simulate_pvalues(mu_samp = 0.25, sd_samp = 1, M = 1000, n = 25, alpha = 0.05)
simulate_pvalues(mu_samp = 0.25, sd_samp = 1, M = 1000, n = 50, alpha = 0.05)
```



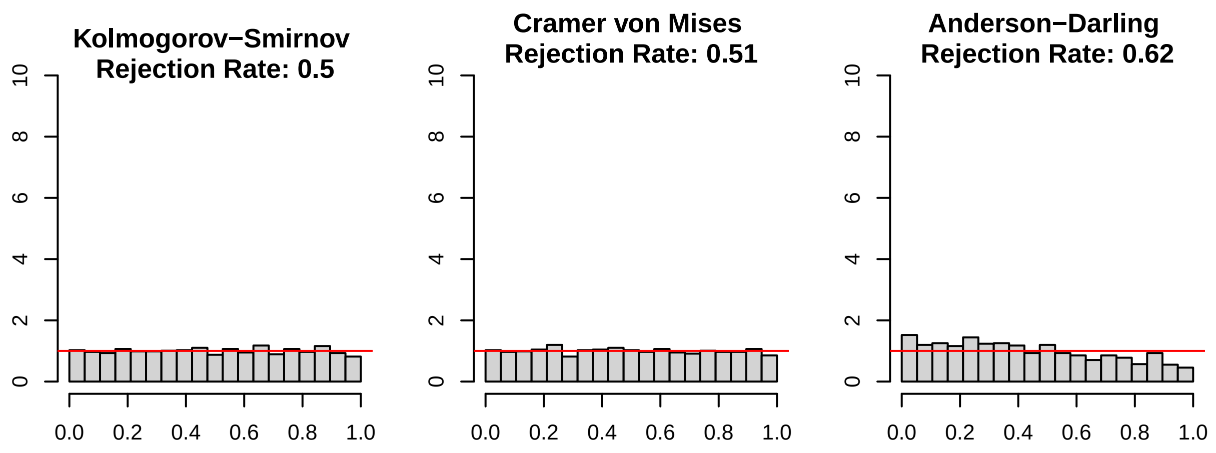
```r
simulate_pvalues(mu_samp = 0.50, sd_samp = 1, M = 1000, n = 25, alpha = 0.05)
```


Kolmogorov−Smirnov
Rejection Rate: 0.35
Cramer von Mises
Rejection Rate: 0.4
Anderson−Darling
Rejection Rate: 0.42
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0



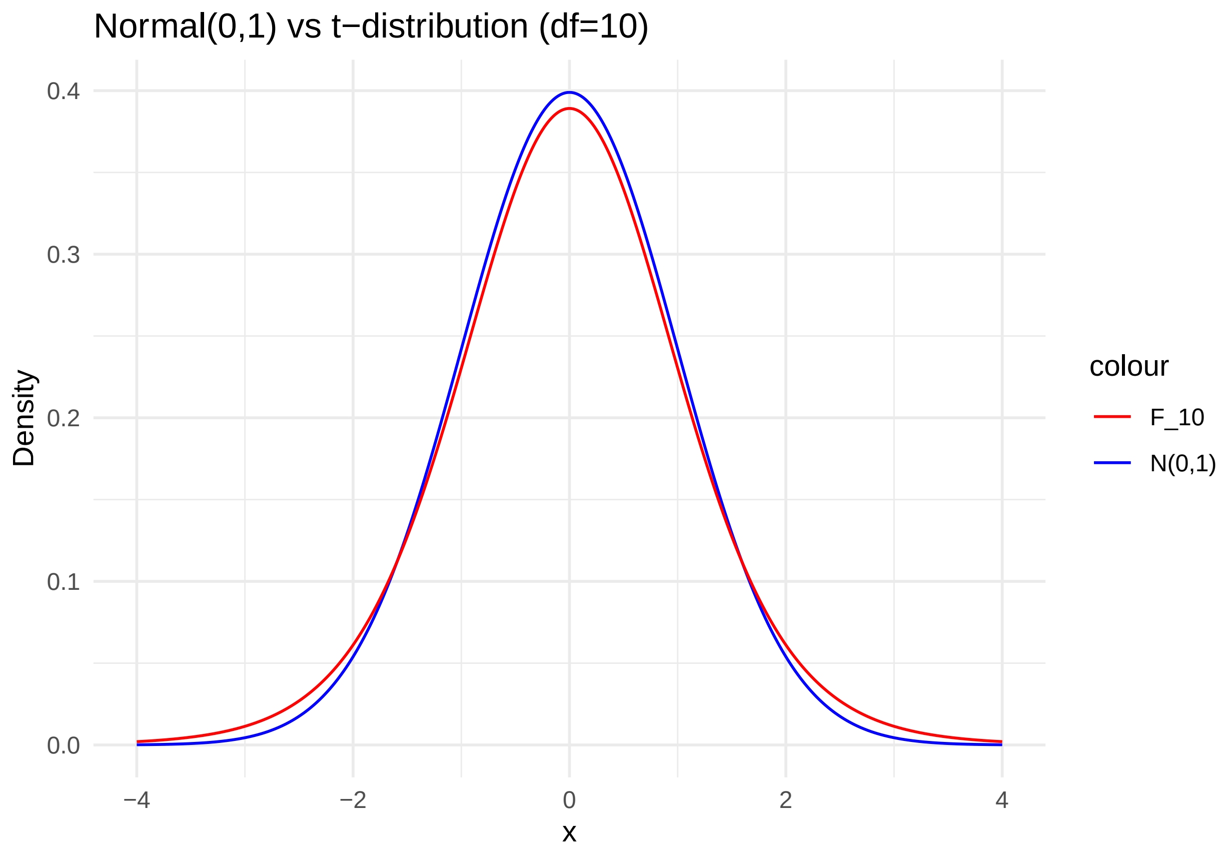
As last experiment, we now draw the samples from a t-students distribution with $\mu = 0$ and $df = 10$. Like before, we expect rejection rates to be high as the null hypothesis $H_0 : F = F_N$ is false. This is inline with the results, with an additional observation that the rejection rate of the Anderson-Darling test is higher than those of the other two test. This can be explained with the emphasises of the Anderson-Darling test on the tails of the distribution, picking up better the different behaviors Normal and t-students distribution have at the tails.

```
#
simulate_pvalues(mu_samp = 0,M = 1000, n = 50, alpha = 0.5, dist = "t-students")
```



This difference in the tails can be seen by comparing both distributions as done in the plot below.

```r
# Plot differences between Normal/t-students

x = seq(-4, 4, length.out = 1000)

normal_density = dnorm(x, mean = 0, sd = 1)
t_density = dt(x, df = 10)

df = data.frame(x = x, normal_density = normal_density, t_density = t_density)

# Plot
ggplot(df, aes(x)) +
  geom_line(aes(y = normal_density, color = "N(0,1)")) +
  geom_line(aes(y = t_density, color = "F_10")) +
  labs(title = "Normal(0,1) vs t-distribution (df=10)",
       y = "Density", x = "x") +
  scale_color_manual(values = c("N(0,1)" = "blue", "F_10" = "red")) +
  theme_minimal()
```


Normal(0,1) vs t−distribution (df=10)
Density
0.4
0.3
0.2
0.1
0.0
−4
−2
0
2
4
x
colour
F_10
N(0,1)