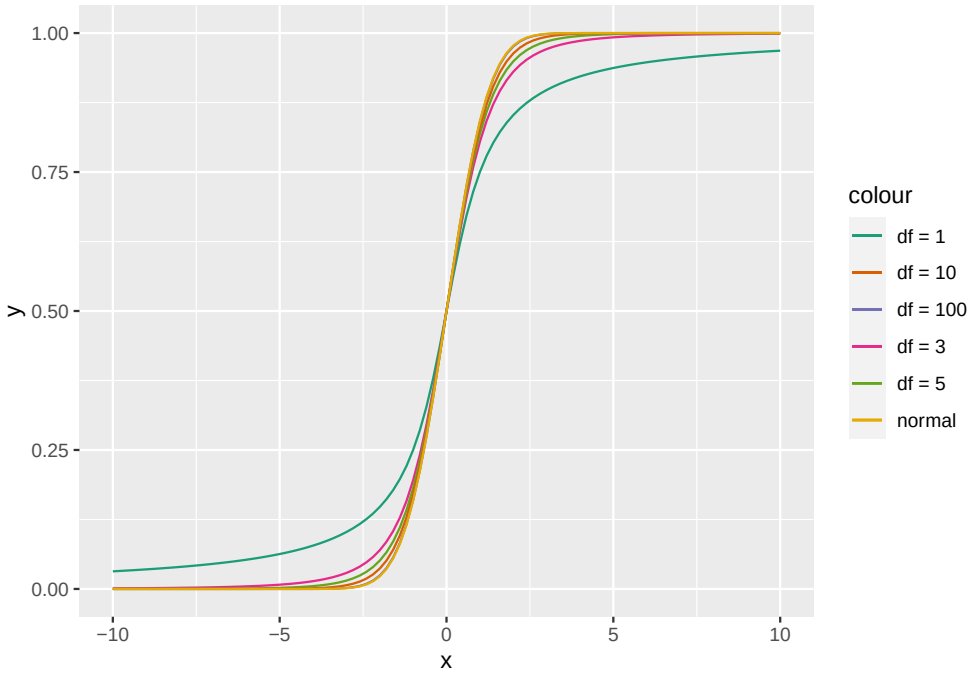
TFALDS Lab 2

Klaudia Weigel

1 Exercise 1

Use the `pt` *and* `pnorm` *functions in R to draw one graph with cdfs of the standard normal distribution and the Student's distribution with degrees of freedom* $df \in \{3, 5, 10, 100\}$.

```
library(ggplot2)

ggplot(data.frame(x=c(-10, 10)), aes(x)) +
  stat_function(fun= pt, args = list(df=1),aes(colour="df = 1"))+
  stat_function(fun= pt, args = list(df=3),aes(colour="df = 3"))+
  stat_function(fun= pt, args = list(df=5),aes(colour="df = 5"))+
  stat_function(fun= pt, args = list(df=10),aes(colour="df = 10"))+
  stat_function(fun= pt, args = list(df=100),aes(colour="df = 100"))+
  stat_function(fun= pnorm,aes(colour="normal"))+
  scale_colour_brewer(palette="Dark2")
```



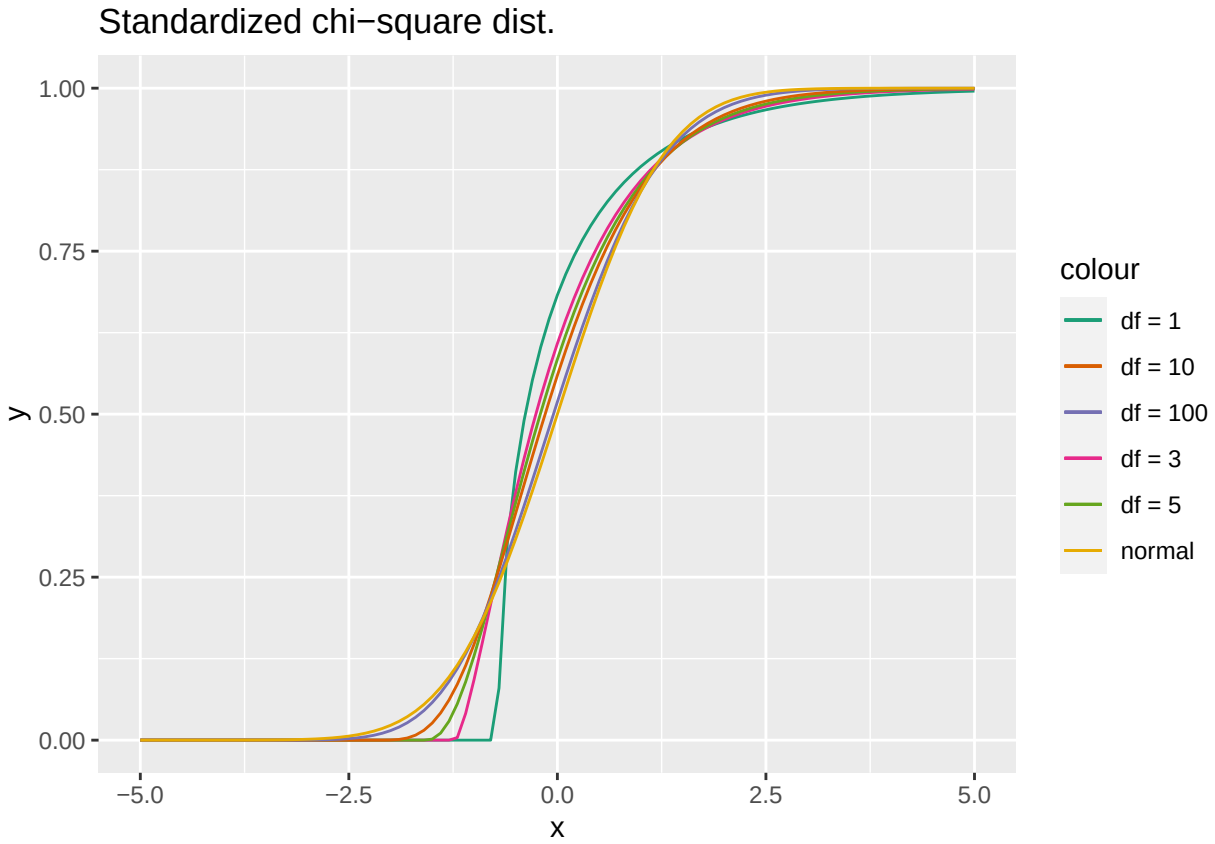
We see that as the degrees of freedom increse the standard normal distribution becomes closer to the Student's t-distribution.

2 Exercise 2

Use the pnorm and **`pchisq`** *functions in R to draw one graph with cdfs of the standard normal distribution and the standardized chi-square distribution with degrees of freedom* $df \in \{1, 3, 5, 10, 100\}$.
The standarization of the normal distribution has the following form:

$$T = \frac{\chi^2_{df} - df}{\sqrt{2df}}.$$

```
standardized_pchisq = function(x, df) {
  return(pchisq(x*sqrt(2*df) + df, df))
}
```



We see that the standardized chi-square distribution gets closer to the normal distribution as the degrees of freedom increase.

3 Exercise 3

a) *Let* $X_1, ..., X_n$ *(n=100) be a sample from the Poisson distribution. Consider a test* $H_{0i} : E(X_i) = 5$ *against* $H_{1i} : E(X_i) > 5$*, which rejects the null hypothesis for large values of* $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$*. Write a function in R to calculate the p-value for this test.*

$$\text{p-value} = P_{H_0}(\frac{1}{n}\sum_{i=1}^{n} X_i > \frac{1}{n}\sum_{i=1}^{n} x_i) = P_{H_0}(\sum_{i=1}^{n} X_i > \sum_{i=1}^{n} x_i)$$

$\sum_{i=1}^{n} X_i$ has the Poisson distribution with mean $n\lambda$, where λ is the mean of X_1.

```r
lambda_H0 = 5
n = 100
pois_sample_H0 = rpois(n = n, lambda = lambda_H0)
ppois(sum(pois_sample_H0), lambda = n*lambda_H0, lower.tail = FALSE)
```

```
## [1] 0.6945545
```

b) *Now, consider 1000 of similar hypothesis and the meta problem of testing the global hypothesis* $H_0 = \bigcap_{j=1}^{1000} H_{0j}$ *and use simulations to estimate the probability of the type I error for the Bonferroni and Fisher tests at the signifcance level* $\alpha = 0.05$. *Draw histogram of p-values and discuss their distribution. Why the probability of the type I error of Fisher's test might be slightly different from* α*?*

In the Bonferroni's method we test each H_{0i} at level α/n, where n is the number null hypotheses, and reject H_0, whenever any of the H_{0i} is rejected. Thus we reject H_0 when

$$min_i\ p_i \leq \alpha/n$$

Where p_i is a p-value for the $i-th$ test.
The type I error (the size) of the Bonferroni's test is approximated by $1 - e^{-\alpha}$.

Fisher's Combination Test is a global test that rejects for large values of the following statistic:

$$T = -\sum_{i=1}^{n} 2\log\ p_i$$

If the p-values $p_1, ..., p_n$ are independent then the Fisher statistic under the enull hypothesis has χ^2_{2n} distribution.

```r
n=100
n_samples=1000
alpha=0.05

pval = function(x){
  ppois(x, lambda = n*5, lower.tail = FALSE)
}

bonferroni = function(p){
  return (min(p) <= alpha/length(p))
}

fishtest = function(p) {
  return (-2*sum(log(p)) > qchisq(1-alpha, 2*length(p)))
}

type_i_err = function(x, k) {
  # 1000 values of the sum statistic in each column
  Xa = matrix(rpois(n_samples*k, lambda = 5*n), n_samples, k)
  Xa = pval(Xa)
  bonf_res = mean(apply(Xa, 2, bonferroni))
  fish_res= mean(apply(Xa, 2, fishtest))
  return(c(bonf_res, fish_res))
}

res_type_i_err50 = sapply(1:5, type_i_err, k=50)
res_type_i_err200 = sapply(1:5, type_i_err, k=200)
res_type_i_err500 = sapply(1:5, type_i_err, k=500)
```

```
res_type_i_err1000 = sapply(1:5, type_i_err, k=1000)

df_type_i_err_bonf = data.frame(res_type_i_err50[1,], res_type_i_err200[1,],
                                res_type_i_err500[1,],  res_type_i_err1000[1,])

colnames(df_type_i_err_bonf) = c("k=50", "k=200", "k=500", "k=1000")

df_type_i_err_fish = data.frame(res_type_i_err50[2,], res_type_i_err200[2,],
                                res_type_i_err500[2,],  res_type_i_err1000[2,])

colnames(df_type_i_err_fish) = c("k=50", "k=200", "k=500", "k=1000")
```

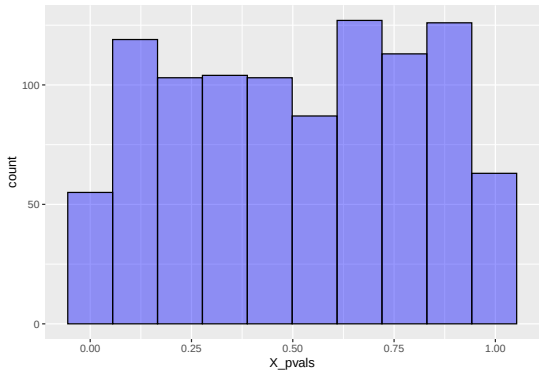
We calculate the probability of commiting the type I error for k simulations of both the tests.

Table 1: Type I error of Bonferroni's test.

	k=50	k=200	k=500	k=1000
1	0.00	0.075	0.070	0.055
2	0.18	0.070	0.040	0.054
3	0.04	0.075	0.058	0.068
4	0.02	0.035	0.058	0.055
5	0.06	0.055	0.056	0.073

Table 2: Type I error of Fisher's test.

	k=50	k=200	k=500	k=1000
1	0.16	0.205	0.178	0.157
2	0.20	0.145	0.162	0.156
3	0.12	0.125	0.148	0.156
4	0.16	0.155	0.158	0.154
5	0.16	0.150	0.146	0.152



The larger the number of the Bonferroni tests the closer the value of the type I error is to the chosen significance level 0.05. In case of Fisher's test however the values differ from the significance level. One of the assumptions of the Fisher's test is the uniform distribution of the p-values. This condition, however is not satisfied in our case as the Poisson distribution is not continuous. For large values of λ the Poisson distribution can be approximated by the normal distribution. We can check how the probability of the type I error will change if we increase the sample size to $n = 1000$.

Table 3: Type I error of the Bonferroni's test.

	k=50	k=200	k=500
1	0.02	0.055	0.056
2	0.08	0.065	0.044
3	0.04	0.060	0.040
4	0.08	0.060	0.052
5	0.02	0.040	0.048

Table 4: Type I error of the Fisher's test.

	k=50	k=200	k=500
1	0.06	0.095	0.060
2	0.08	0.075	0.076
3	0.04	0.080	0.076
4	0.06	0.080	0.090
5	0.08	0.085	0.082

We see that the probability of the type I error is a lot closer to the theoretical value.

c) *Use simulations to compare the power of the Bonferoni and Fisher tests for two alternatives:*

- *Needle in the haystack* $E(X_1) = 7$, *for* $j \in \{2, ..., 1000\}$ $E(X_j) = 5$.
- *Many small effects* $j \in \{1, ..., 100\}$ $E(X_j) = 5.5$ *and for* $j \in \{101, ..., 1000\}$ $E(X_j) = 5$.

Since the Bonferroni's test looks only at the smallest p-value, it is most suited for situations where we expect at least one of the p-values to be very significant, such as in the needle problem. Fisher's Combination Test in the other hand aggregates all of the p-values (in log scale), rather than just looking at the minimum p-value. Hence, we expect this test to be powerful when there are many small effects.

```
n = 100
needle = function(x, k) {
  Xa = matrix(rpois(n_samples*k, lambda = 5*n), n_samples, k)
  # First sample comes from E(7) distribution
  Xa[1,] = rpois(k, lambda = 7*n)
  Xa = pval(Xa)
  bonf_res = mean(apply(Xa, 2, bonferroni))
  fish_res= mean(apply(Xa, 2, fishtest))
  return(c(bonf_res, fish_res))
}

small_effects = function(x, k) {
  Xb = matrix(rpois(n_samples*k, lambda = 5*n), n_samples, k)
  Xb[1:100, ] = rpois(100*k, lambda = 5.5*n)
  Xb = pval(Xb)
  bonf_res = mean(apply(Xb, 2, bonferroni))
  fish_res= mean(apply(Xb, 2, fishtest))
  return(c(bonf_res, fish_res))
}
```

Table 5: Power of the Bonferroni test for the needle problem.

	k=50	k=200	k=500	k=1000
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1

Table 6: Power of the Fisher's test for the needle problem.

	k=50	k=200	k=500	k=1000
1	0.58	0.585	0.550	0.543
2	0.64	0.575	0.576	0.573
3	0.54	0.550	0.578	0.562
4	0.60	0.555	0.592	0.572
5	0.54	0.515	0.578	0.572

Table 7: Power of the Bonferroni test for the small effects problem.

	k=50	k=200	k=500	k=1000
1	1.00	1.000	0.994	0.997
2	0.98	0.995	0.994	0.997
3	1.00	1.000	0.998	0.997
4	1.00	1.000	0.994	0.995
5	0.98	1.000	0.994	0.995

Table 8: Power of the Fisher's test for the small effects problem.

	k=50	k=200	k=500	k=1000
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1

We see that values 5.5 is already large enough for the Bonferroni's test to catch it, we can check what would happen if we were to decrease this value to 5.2.

Table 9: Power of Bonferroni test for the small effects problem.

	k=50	k=200	k=500	k=1000
1	0.18	0.185	0.194	0.197
2	0.10	0.200	0.186	0.199
3	0.22	0.170	0.226	0.184
4	0.20	0.155	0.194	0.170
5	0.20	0.190	0.196	0.189

Table 10: Power of Fisher's test for the small effects problem.

	k=50	k=200	k=500	k=1000
1	0.98	0.990	0.990	0.993
2	0.98	0.985	0.994	0.987
3	0.98	0.980	0.984	0.987
4	0.98	0.990	0.994	0.990
5	1.00	0.980	0.988	0.984

We see that if the value of the mean is equal to 5.2 the Fisher's test performs significantly better than the Bonferroni's test.

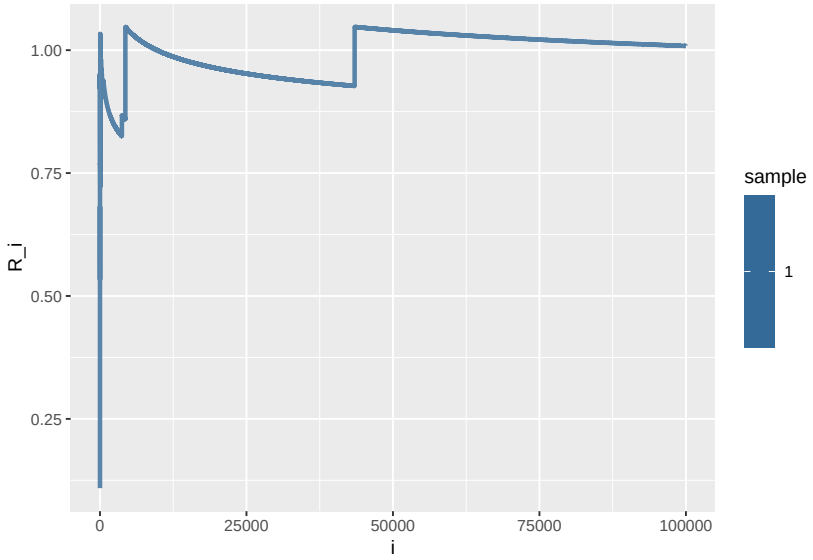
4 Exercise 4

Draw 100000 iid normal variables $X_1, ..., X_{100000}$ *from N(0, 1) and for* $i \in \{2, ..., 100000\}$ *plot the graph of the function* $R_i = \frac{max\{X_j, j=1,...,i\}}{\sqrt{2logi}}$. *Repeat the above experiment 10 times and plot the respective trajectories of* R_i *on the same graph.*

We know that R_i converges to 1 in probability.

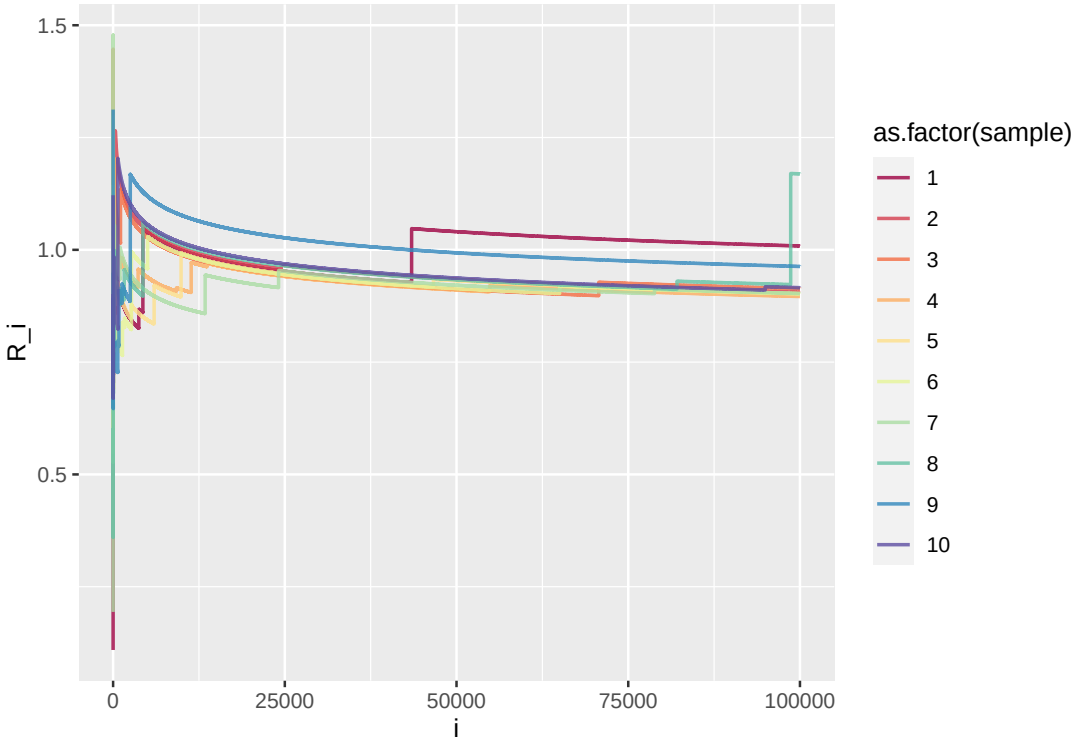
Plot for a single experiment.

```
size = 10^5
i = 2:size
#cumulative max
nsample_cummax = cummax(abs(rnorm(size-1)))
R = nsample_cummax/sqrt(2*log(i))
df = data.frame(i, R, 1)
colnames(df) = c("i", "R_i", "sample")
ggplot(data = df, aes(x=i, y=R_i)) +
   geom_line(size=1.2, alpha=.8, aes(color = sample))
```



```
for (k in 2:10) {
  s = cummax(abs(rnorm(size-1)))
  df2 = data.frame(i, s/sqrt(2*log(i)), k)
  colnames(df2) = c("i", "R_i", "sample")
  df = rbind(df, df2)
}
```

```
ggplot(data = df, aes(x=i, y=R_i, group=as.factor(sample), color=as.factor(sample))) +
  geom_line(size=0.7, alpha=.8) +
  scale_color_brewer(palette="Spectral")
```



We see that the values oscillate around one.

5 Exercise 5

Let $Y = (Y_1, ..., Y_n)$ be the random vector from $N(\mu, I)$ distribution. Consider the statistics L of the optimal Neyman-Pearson test for the hypothesis $H_0 : \mu = 0$ for the classical needle in haystack problem H_A - one of the elements of μ is equal to γ

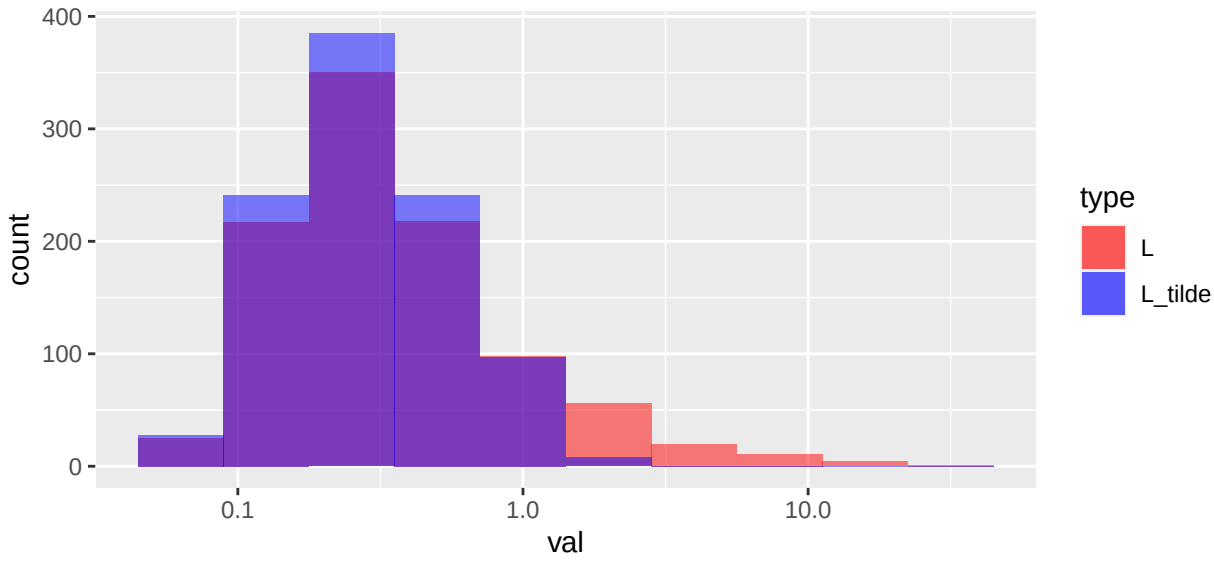
$$L = \frac{1}{n} \sum e^{\gamma Y_i - \gamma^2/2},$$

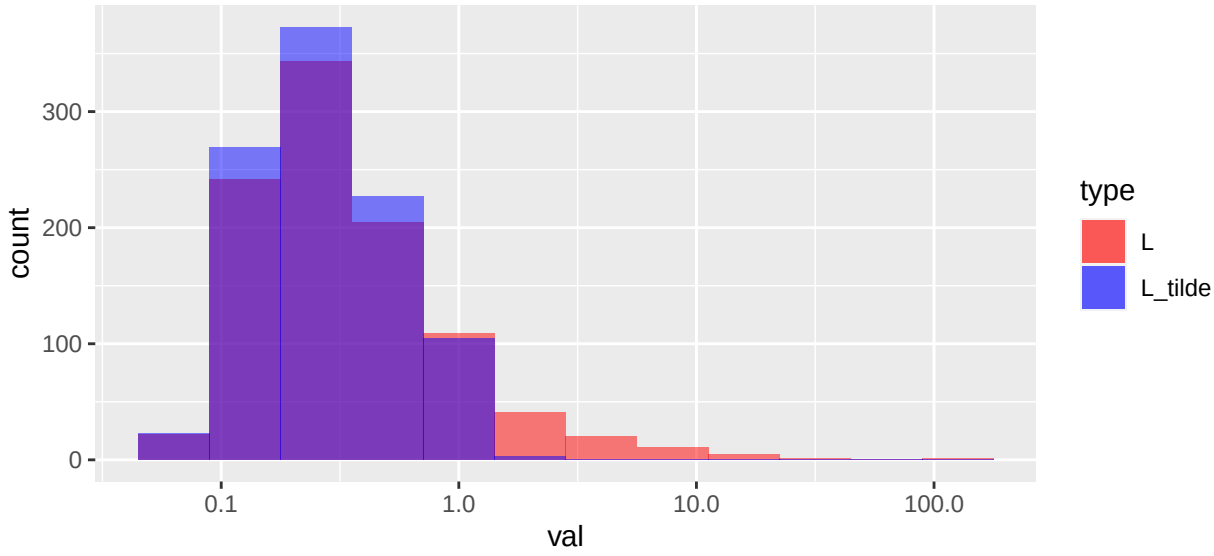
$$\tilde{L} = \frac{1}{n} \sum_{i=1}^{n} \left[e^{\gamma Y_i - \gamma^2/2} 1\{Y_i < \sqrt{2 \log n}\} \right].$$

The histrograms were plotted on a log10 scale for better visibility.

```
eps = 0.1

get_l = function(rep, n) {
  equal = 0
```

```
  y = rnorm(n)
  c = sqrt(2*log(n))
  gamma = (1+eps)*c
  expr = exp(y*gamma - (gamma^2)/2)
  l = mean(expr)
  l_tilde = mean(expr*(y<c))
  if(l == l_tilde) equal = 1
  return(c(l, l_tilde, equal))
}

calc_and_plot = function(n) {
  rep = 1000
  res = lapply(1:rep, get_l, n = n)
  # matrix with each column equal to (l, l_tilde) (res[1, ] - l, res[2, ] - l_tilde)
  res = matrix(unlist(res), nrow = 3)
  prob = sum(res[3,])/rep

  df1 = data.frame(res[1, ]); colnames(df1) = c("val")
  df2 = data.frame(res[2, ]); colnames(df2) = c("val")
  df1 = cbind(df1, "L"); colnames(df1) = c("val", "type")
  df2 = cbind(df2, "L tilde"); colnames(df2) = c("val", "type")

  dat = data.frame(rbind(df1, df2))
  p = ggplot(dat, aes(x=val, fill = as.factor(type))) +
    guides(fill = guide_legend(override.aes= list(alpha = 0.4))) +
    scale_x_log10() +
    geom_histogram(data = subset(dat, type == "L"), alpha = 0.5, binwidth =0.3) +
    geom_histogram(data = subset(dat, type == "L tilde"), alpha = 0.5, binwidth = 0.3)  +
    scale_fill_manual(name = "type", values = c("red", "blue"), labels = c("L", "L_tilde"))

  var_vec = apply(res[1:2,], 1, var)
  return(list(vars = var_vec, prob = prob, plot = p))
}
res10_3 = calc_and_plot(1000)
res10_4 = calc_and_plot(10^4)
res10_5 = calc_and_plot(10^5)
vardf = cbind(res10_3$vars, res10_4$vars, res10_5$vars)
colnames(vardf) = c("n = 10^3", "n = 10^4", "n = 10^5")
rownames(vardf) = c("L", "L_tilde")

probdf = cbind(res10_3$prob, res10_4$prob, res10_5$prob)
colnames(probdf) = c("n = 10^3", "n = 10^4", "n = 10^5")
```


400
300
200
100
0
count
0.1
1.0
10.0
val
type
L
L_tilde


300
200
100
0
count
0.1
1.0
10.0
100.0
val
type
L
L_tilde

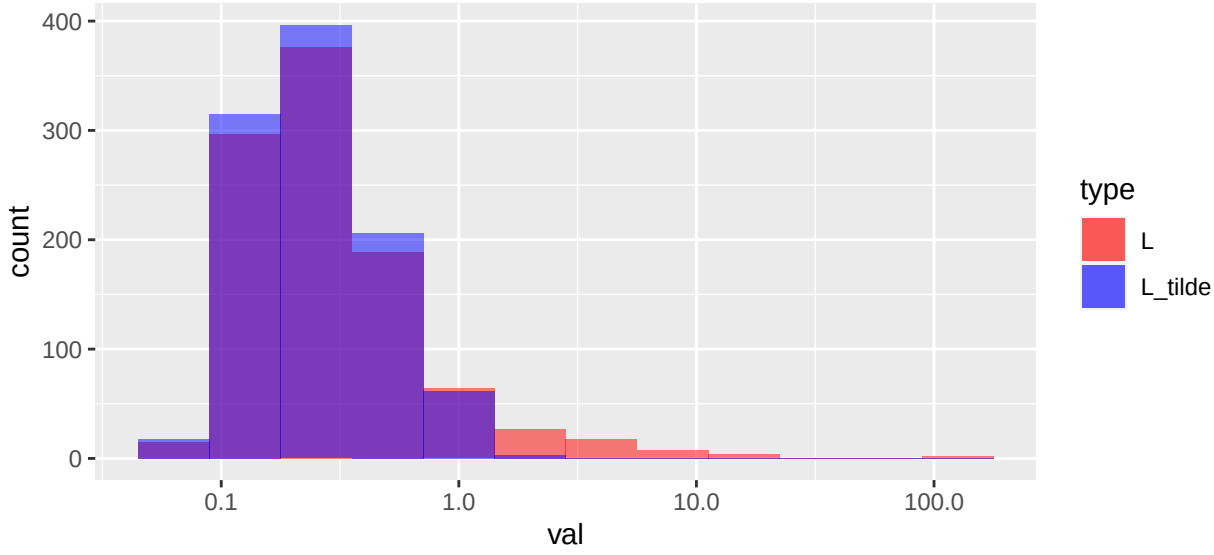

400
300
200
100
0
count
0.1
1.0
10.0
100.0
val
type
L
L_tilde

Table 11: Variances for L and L_tilde.

	n = 10^3	n = 10^4	n = 10^5
L	3.0377101	11.5866625	23.1899003
L_tilde	0.0774096	0.0686101	0.0510894

It was shown that the variance of $\tilde{L}$ converges to zero, which can also be seen from the simulation.

Table 12: P(L = L_tilde)

n = 10^3	n = 10^4	n = 10^5
0.903	0.912	0.935

We see from the plots that most of the values for $\tilde{L}$ lie relatively close together, whereas the distribution L has a heavy tail.

We have $P(\tilde{L} \neq L) \leq P(\max Y_i \geq \sqrt{2 \log n}) \to 0$. Therefore $P(\tilde{L} = L)$ should converge to 1, which also can be seen from the simulations.

6 Exercise 6

Use simulations to find the critical value of the optimal Neyman Pearson test and compare the power of this test and the Bonferoni test for the needle in haystack problem with $n \in \{500, 5000, 50000\}$ *and the needles -* $\mu_1 = 1.05\sqrt{2 \log n}, \mu_2 = ... = \mu_n = 0$

We consider the following testing problem

$$H_{0i} : \mu_i = 0 \quad \text{against} \quad H_{1i} : \mu_i \neq 0, \text{ where i is uniformly distributed.}$$

The global hypothesis is $\bigcap_i H_{0i}$.

We want to find a value such that

$$P_{H_0}(L > c_\alpha) = \alpha$$
$$c_\alpha = Quantile(1 - \alpha).$$

Since we don't know L's distribution, c_α is a sample quantile. We will calculate it by generating multiple values of L and using the `quantile` function.

```
# critical value for L
L = function(X, eps) {
  p = length(X)
  m = (1+eps)*sqrt(2*log(p))
  mean(exp(X*m - m^2/2))
}

rep = 200
# one sample in each row
M500 = matrix(rnorm(rep*500), nrow = rep)
M5000 = matrix(rnorm(rep*5000), nrow = rep)
```

```
M50000 = matrix(rnorm(rep*50000), nrow = rep)

l1.05_1 = sapply(1:rep, function(i) L(M500[i,], 0.05))
l1.05_2 = sapply(1:rep, function(i) L(M5000[i,], 0.05))
l1.05_3 = sapply(1:rep, function(i) L(M50000[i,], 0.05))

l1.2_1 = sapply(1:rep, function(i) L(M500[i,], 0.2))
l1.2_2 = sapply(1:rep, function(i) L(M5000[i,], 0.2))
l1.2_3 = sapply(1:rep, function(i) L(M50000[i,], 0.2))

results2 = data.frame("1.05" = c(quantile(l1.05_1, probs=0.95),
                                 quantile(l1.05_2, probs=0.95), quantile(l1.05_3, probs=0.95)),
                      "1.2" = c(quantile(l1.2_1, probs=0.95),
                                quantile(l1.2_2, probs=0.95), quantile(l1.2_3, probs=0.95)),
                      row.names = c("n=500", "n=5000", "n=50000"))

colnames(results2) = c("1.05", "1.2")
```

Table 13: Critical values.

	1.05	1.2
n=500	2.500166	1.8212897
n=5000	1.429663	0.9708201
n=50000	3.405468	2.5032512

From the lecture we know that the Bonforreni test for finding a needle when the all but one of the test statistics come from the standard normal distribution, (the needle being from the normal distribution with $\mu \neq 0$), only performs well the mean of the needle is greater than $(1+\epsilon)\sqrt{2\log n}$. If that mean is smaller then the Bonferroni's test does no better than a random throw of a coin.

```
# powers of tests
n = c(500, 5000, 50000)
k=800 # number of simulations
alpha=0.05
mi_1 = sapply(n, function(x) 1.05*sqrt(2*log(x)))
mi_2 = sapply(n, function(x) 1.2*sqrt(2*log(x)))

# p-value for a two-sided alternative
pval_norm = function(x){
  return (2*pnorm(-abs(x)))
}

bonf_power_1.05 = vector(mode = "numeric", length = 3)
bonf_power_1.2 = vector(mode = "numeric", length = 3)

for (i in 1:3) {
  # n values of test statistics for one global test in each column
  X = matrix(rnorm(n[i]*k), nrow = n[i], k)
  X1 = X2 = X
  # first test statistic in each column has a different mean
  X1[1,] = X[i,] + mi_1[i]
  X1 = pval_norm(X1)
```

```
  X2[1,] = X[i,] + mi_2[i]
  X2 = pval_norm(X2)

  bonf_power_1.05[i] = mean(apply(X1, 2, bonferroni))
  bonf_power_1.2[i] = mean(apply(X2, 2, bonferroni))
}

NP_power = function(k, n, eps, critical_val) {
  m = (1+eps)*sqrt(2*log(n))
  # n values of test statistics in each column
  Z = matrix(rnorm(n*k), nrow = n, ncol = k)
  Z[1, ] = Z[1, ] + m
  mean(sapply(1:k, function(i) L(Z[,i], eps) > critical_val))
}

k=100
NP_powers1.05 = c(NP_power(k, n[1], 0.05, l1.05_1), NP_power(k, n[2], 0.05, l1.05_2),
                  NP_power(k, n[3], 0.05, l1.05_3))
NP_powers1.05
```

```
## [1] 0.82505 0.85085 0.82120
```

```
NP_powers1.2 = c(NP_power(k, n[1], 0.2, l1.2_1), NP_power(k, n[2], 0.2, l1.2_2),
                  NP_power(k, n[3], 0.2, l1.2_3))
NP_powers1.2
```

```
## [1] 0.94550 0.92705 0.94635
```

```
res = cbind(NP_powers1.05, NP_powers1.2, bonf_power_1.05, bonf_power_1.2)
colnames(res) = c("NP 1.05", "NP 1.2", "Bonf 1.05", "Bonf 1.2")
```

Table 14: Powers of the NP and Bonf tests.

NP.1.05	NP.1.2	Bonf.1.05	Bonf.1.2
0.82505	0.94550	0.47125	0.64125
0.85085	0.92705	0.49375	0.71500
0.82120	0.94635	0.51125	0.76625

We see that the Bonferroni's test is weaker than the Neyman-Pearson test. This is the expected result as the Neyman-Pearson test is a uniformly most powerful test for a simple hypothesis testing.