Exam 2017-11-09

Maximilian Pfundstein

2019-03-17

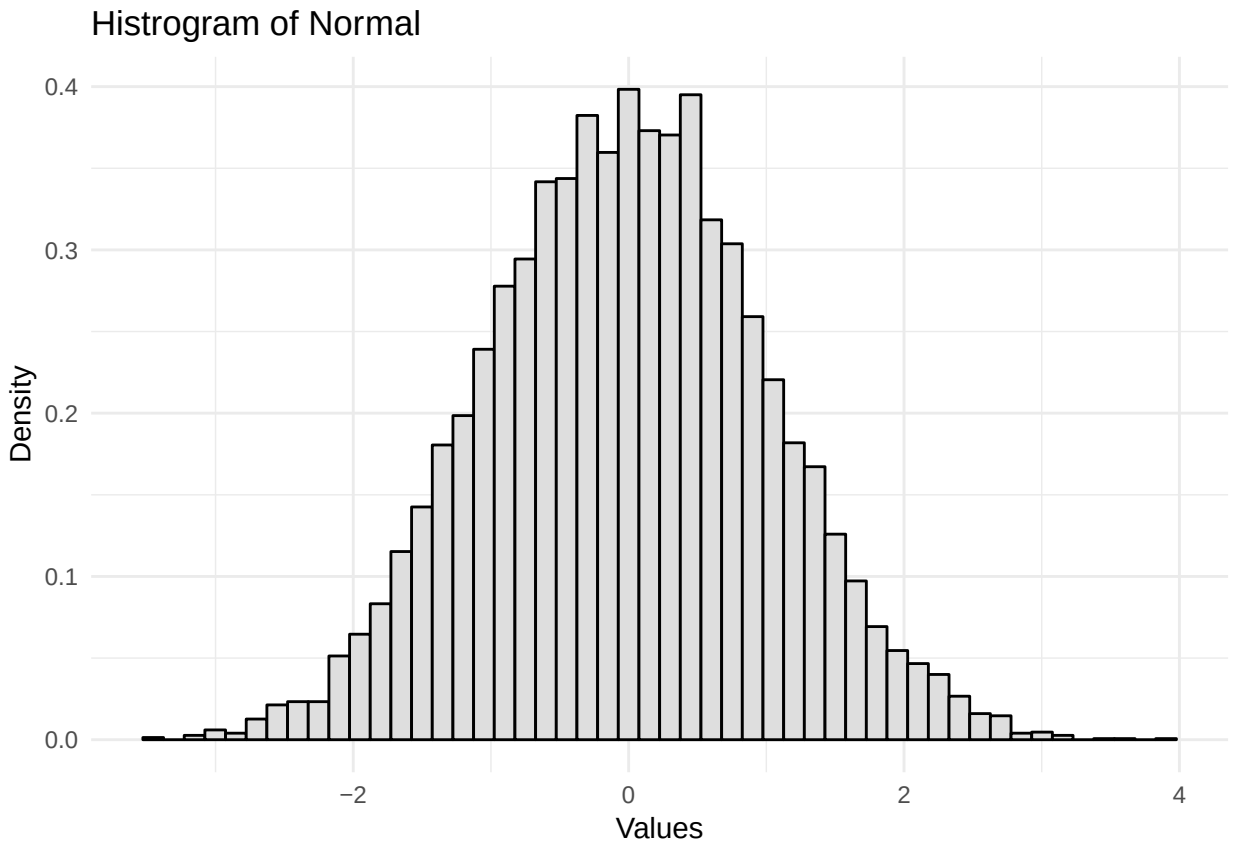
Contents

1 Assignment 1

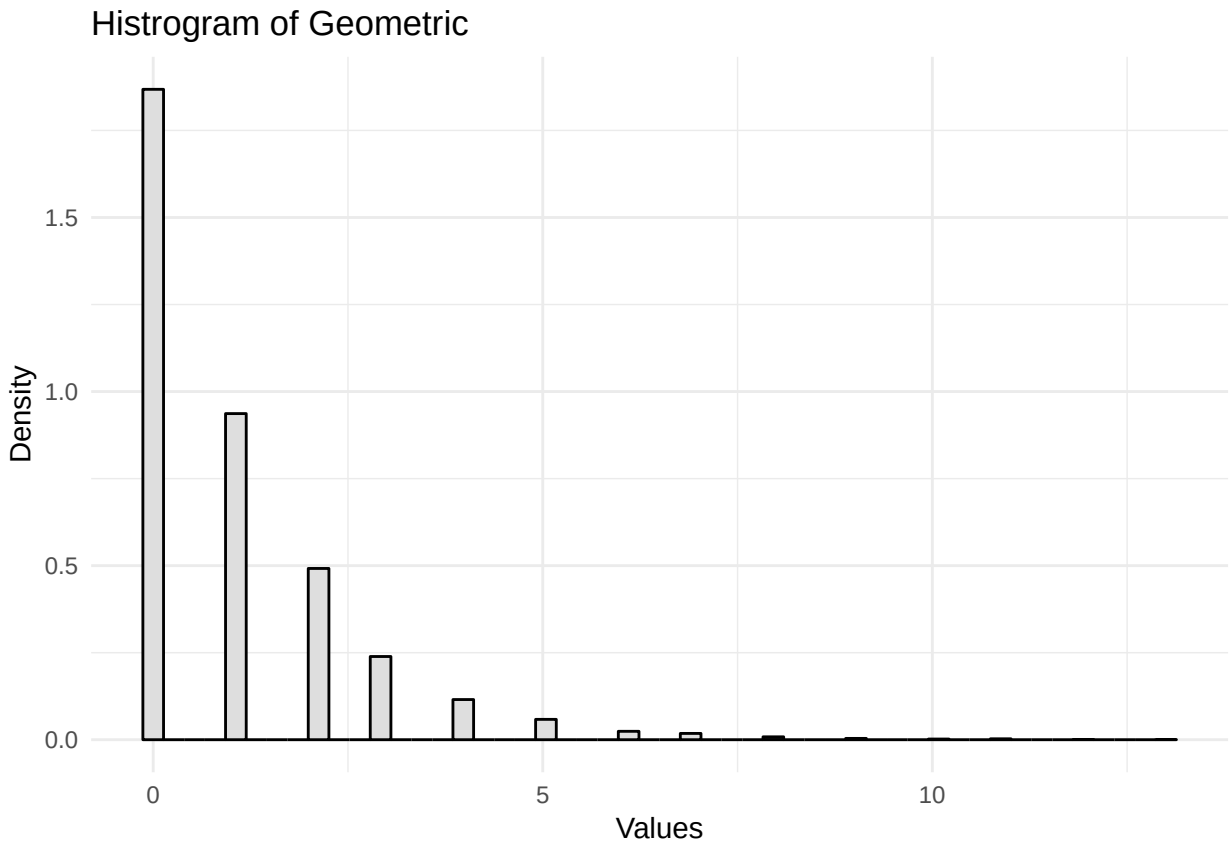
Be aware that there is something wrong! The optimum should be closer (or exact) to the mean or median depending on the metric, but I cannot find the error!

2 Question 1.1

```
rng_rnorm = function(n = 1) {

  theta = runif(ceiling(n/2), 0, 2*pi)
  d = runif(ceiling(n/2))

  res1 = sqrt(-2 * log(d)) * cos(theta)
  res2 = sqrt(-2 * log(d)) * sin(theta)

  res = c(res1, res2)

  return(res[1:n])
}

samples_norm = rng_rnorm(10000)

ggplot(data.frame(samples_norm)) +
  geom_histogram(aes(x = samples_norm, y=..density..),
                 color = "black", fill = "#dedede", bins = 50) +
  labs(title = "Histrogram of Normal",y = "Density",x = "Values",
       color = "Legend") +
  theme_minimal()
```



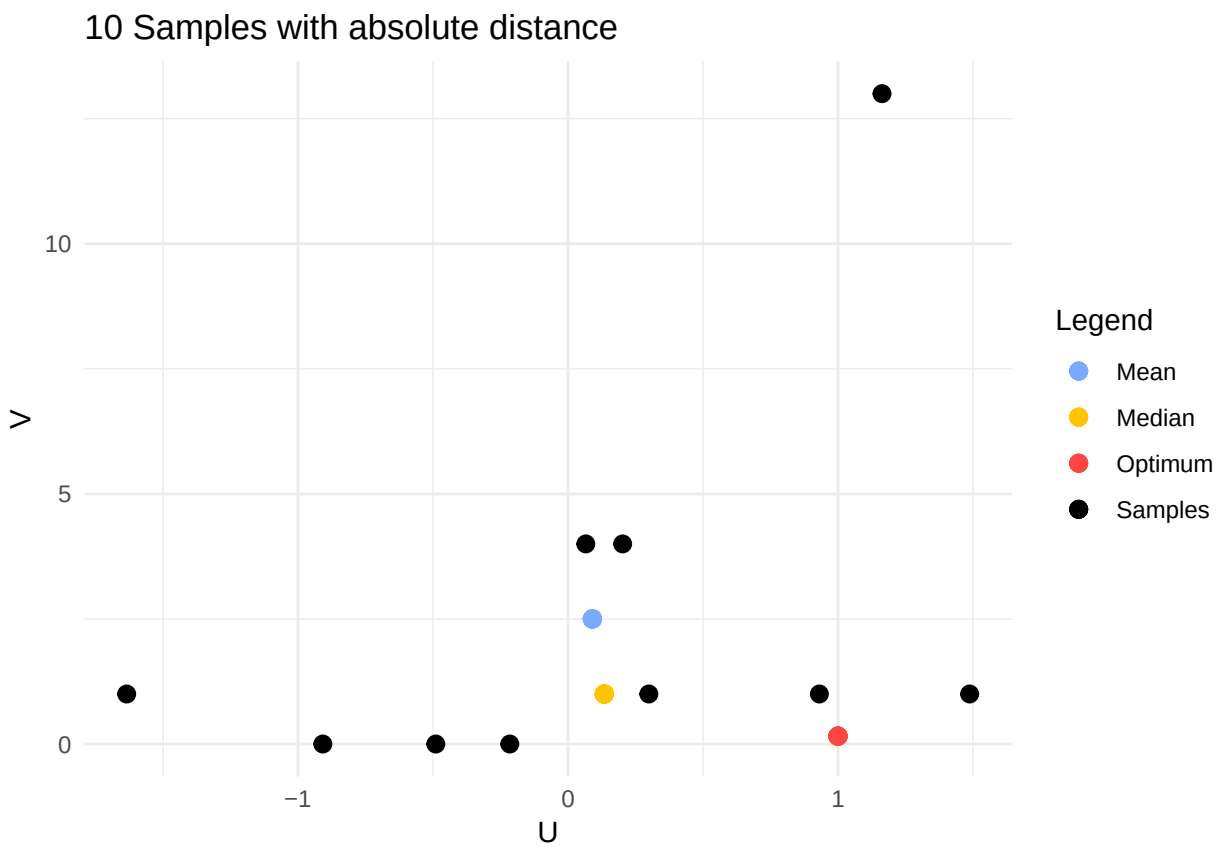

```
rng_rgeom = function(n = 1, p = 0.5) {

  samples = numeric(n)

  for (i in 1:n) {
    cnt = 0
    while (sample(c(0,1), 1 ,replace=FALSE, prob=c(1-p,p)) == 0) {
      cnt = cnt +1
    }
    samples[i] = cnt
  }
  return(samples)
}

samples_geom = rng_rgeom(10000)

ggplot(data.frame(samples_geom)) +
  geom_histogram(aes(x = samples_geom, y=..density..),
                 color = "black", fill = "#dedede", bins = 50) +
  labs(title = "Histrogram of Geometric",y = "Density",x = "Values",
       color = "Legend") +
  theme_minimal()
```



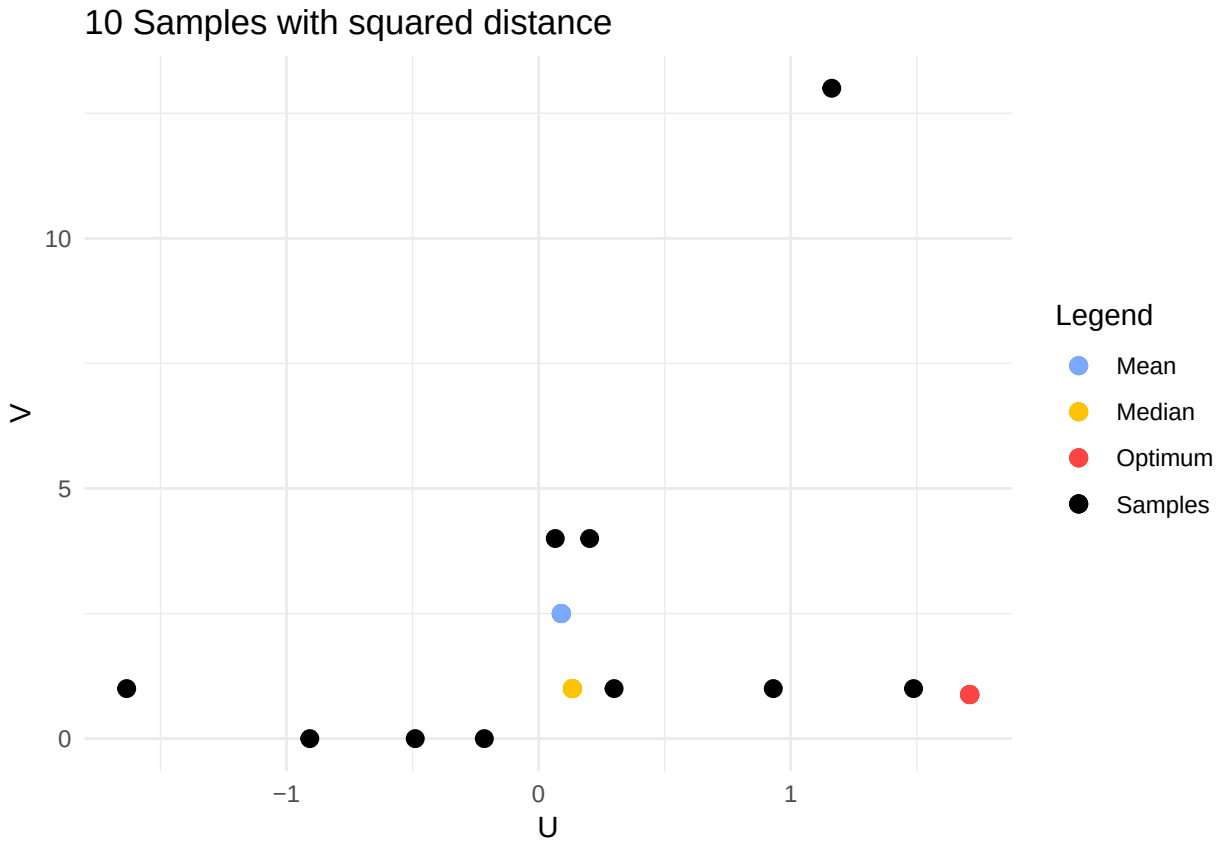

```r
d_abs = function(x, y) {
  return(rowSums(abs(x - y)))
}

d_sq = function(x, y) {
  return(rowSums((x - y)^2))
}

fratchet_mean = function(y, X, w = 1, dist) {
  return(sum(w * dist(X, y)))
}

rng_bivariate = function(n, p) {
  U = rng_rnorm(n)
  V = rng_rgeom(n, p)

  return(as.matrix(data.frame(U, V)))
}

sample_10 = rng_bivariate(10, p = 1/3)
sample_50 = rng_bivariate(50, p = 1/3)
y_init_10 = c(mean(sample_10[,1]), mean(sample_10[,2]))
y_init_50 = c(mean(sample_50[,1]), mean(sample_50[,2]))

y_10_opt_abs = optim(y_init_10, fratchet_mean, method ="BFGS", w = 1,
                     dist = d_abs, X = sample_10, control = list(fnscale = 1))
```

```
y_10_opt_sq = optim(y_init_10, fratchet_mean, method ="BFGS", w = 1,
                    dist = d_sq, X = sample_10, control = list(fnscale =1))

y_50_opt_abs = optim(y_init_50, fratchet_mean, method ="BFGS", w = 1,
                    dist = d_abs, X = sample_50, control = list(fnscale =1))

y_50_opt_sq = optim(y_init_50, fratchet_mean, method ="BFGS", w = 1,
                    dist = d_sq, X = sample_50, control = list(fnscale =1))
```

```
samples_10_df_abs = data.frame(sample_10)
samples_10_df_abs$meanU = mean(samples_10_df_abs$U)
samples_10_df_abs$meanV = mean(samples_10_df_abs$V)
samples_10_df_abs$medianU = median(samples_10_df_abs$U)
samples_10_df_abs$medianV = median(samples_10_df_abs$V)
samples_10_df_abs$optU = y_10_opt_abs$par[1]
samples_10_df_abs$optV = y_10_opt_abs$par[2]

ggplot(samples_10_df_abs) +
  geom_point(aes(x = U, y = V, colour = "Samples"), size = 2, stroke = 1) +
  geom_point(aes(x = meanU, y = meanV, colour = "Mean"), size = 2, stroke = 1) +
  geom_point(aes(x = medianU, y = medianV, colour = "Median"), size = 2, stroke = 1) +
  geom_point(aes(x = optU, y = optV, colour = "Optimum"), size = 2, stroke = 1) +
  labs(title = "10 Samples with absolute distance", y = "V", x = "U", color ="Legend") +
  scale_color_manual(values = c("#7BA9FF", "#FFC300", "#ff4444", "black")) +
  scale_shape_manual(values = c(21, 21, 21)) +
  theme_minimal()
```

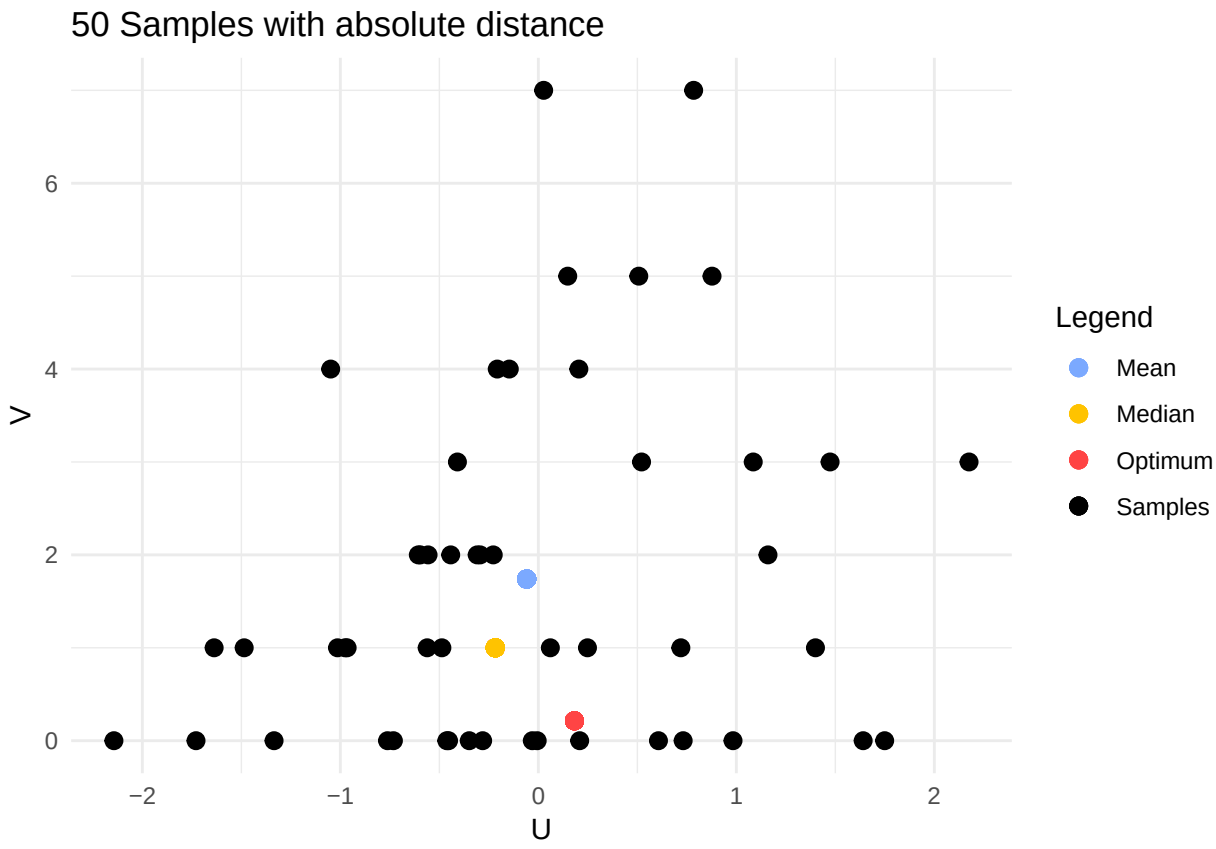


```r
samples_10_df_sq = data.frame(sample_10)
samples_10_df_sq$meanU = mean(samples_10_df_sq$U)
samples_10_df_sq$meanV = mean(samples_10_df_sq$V)
samples_10_df_sq$medianU = median(samples_10_df_sq$U)
samples_10_df_sq$medianV = median(samples_10_df_sq$V)
samples_10_df_sq$optU = y_10_opt_sq$par[1]
samples_10_df_sq$optV = y_10_opt_sq$par[2]

ggplot(samples_10_df_sq) +
  geom_point(aes(x = U, y = V, colour = "Samples"), size = 2, stroke = 1) +
  geom_point(aes(x = meanU, y = meanV, colour = "Mean"), size = 2, stroke = 1) +
  geom_point(aes(x = medianU, y = medianV, colour = "Median"), size = 2, stroke = 1) +
  geom_point(aes(x = optU, y = optV, colour = "Optimum"), size = 2, stroke = 1) +
  labs(title = "10 Samples with squared distance", y = "V", x = "U", color ="Legend") +
  scale_color_manual(values = c("#7BA9FF", "#FFC300", "#ff4444", "black")) +
  scale_shape_manual(values = c(21, 21, 21)) +
  theme_minimal()
```

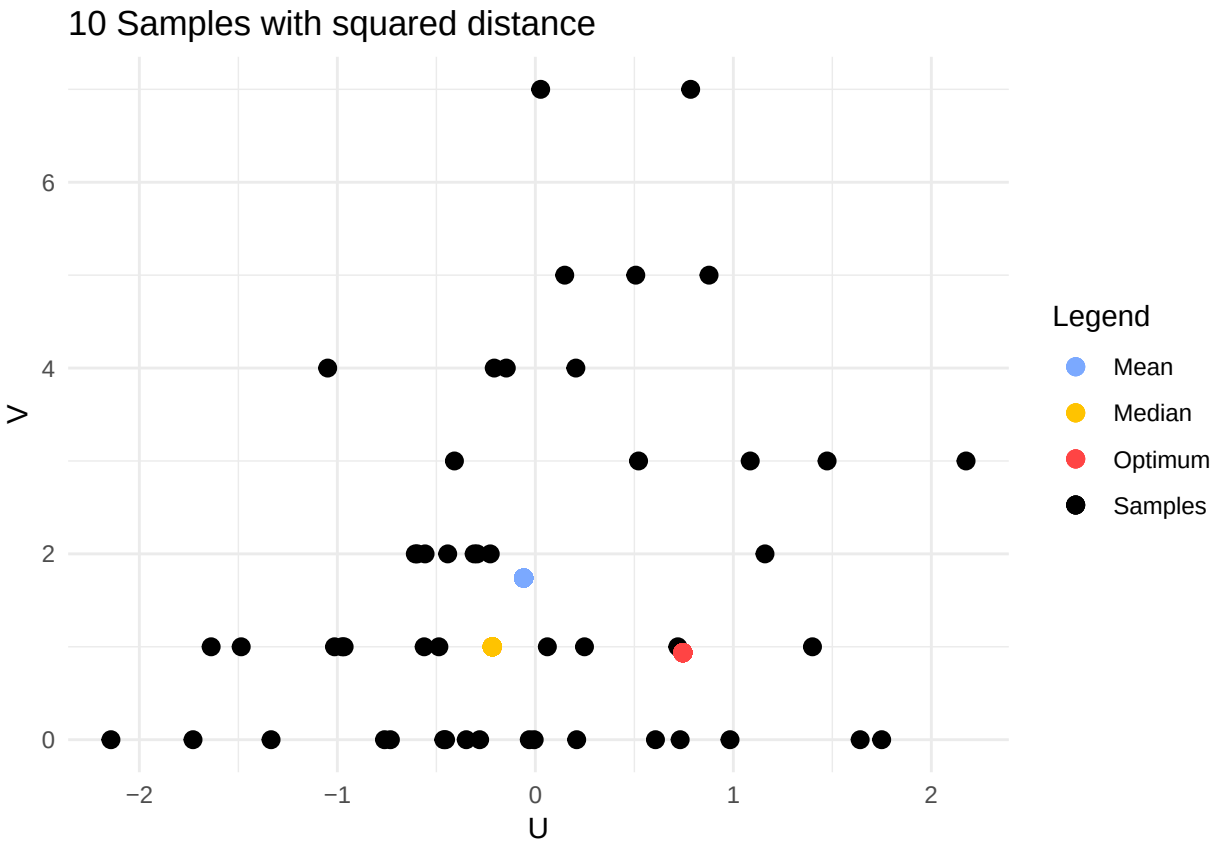



```r
samples_50_df_abs = data.frame(sample_50)
samples_50_df_abs$meanU = mean(samples_50_df_abs$U)
samples_50_df_abs$meanV = mean(samples_50_df_abs$V)
samples_50_df_abs$medianU = median(samples_50_df_abs$U)
samples_50_df_abs$medianV = median(samples_50_df_abs$V)
samples_50_df_abs$optU = y_50_opt_abs$par[1]
samples_50_df_abs$optV = y_50_opt_abs$par[2]

ggplot(samples_50_df_abs) +
```

```
  geom_point(aes(x = U, y = V, colour = "Samples"), size = 2, stroke = 1) +
  geom_point(aes(x = meanU, y = meanV, colour = "Mean"), size = 2, stroke = 1) +
  geom_point(aes(x = medianU, y = medianV, colour = "Median"), size = 2, stroke = 1) +
  geom_point(aes(x = optU, y = optV, colour = "Optimum"), size = 2, stroke = 1) +
  labs(title = "50 Samples with absolute distance", y = "V", x = "U", color ="Legend") +
  scale_color_manual(values = c("#7BA9FF", "#FFC300", "#ff4444", "black")) +
  scale_shape_manual(values = c(21, 21, 21)) +
  theme_minimal()
```




```
samples_50_df_sq = data.frame(sample_50)
samples_50_df_sq$meanU = mean(samples_50_df_sq$U)
samples_50_df_sq$meanV = mean(samples_50_df_sq$V)
samples_50_df_sq$medianU = median(samples_50_df_sq$U)
samples_50_df_sq$medianV = median(samples_50_df_sq$V)
samples_50_df_sq$optU = y_50_opt_sq$par[1]
samples_50_df_sq$optV = y_50_opt_sq$par[2]

ggplot(samples_50_df_sq) +
  geom_point(aes(x = U, y = V, colour = "Samples"), size = 2, stroke = 1) +
  geom_point(aes(x = meanU, y = meanV, colour = "Mean"), size = 2, stroke = 1) +
  geom_point(aes(x = medianU, y = medianV, colour = "Median"), size = 2, stroke = 1) +
  geom_point(aes(x = optU, y = optV, colour = "Optimum"), size = 2, stroke = 1) +
  labs(title = "10 Samples with squared distance", y = "V", x = "U", color ="Legend") +
  scale_color_manual(values = c("#7BA9FF", "#FFC300", "#ff4444", "black")) +
  scale_shape_manual(values = c(21, 21, 21)) +
  theme_minimal()
```



3 Assignment 2

3.1 Question 1

```
rng_runif = function(a, m, x_zero, nmax) {
  if (!(a >= 0 && a<m)) stop("a in [0, m) is required")
  if(nmax%%1 != 0) stop("nmax has to be an integer")

  storage = vector(mode = "numeric", length = nmax+1)
  storage[1] = x_zero

  for(i in 1:(nmax)) {
    storage[i+1] = ((a * storage[i]) %% m)
  }

  return(storage[2:length(storage)-1]/ m)}
```

3.2 Question 2

```
simulate_walk = function(m) {
  position = c(1, 1)
  distance_T = 0
  while (sum(position == c(m, m)) != 2) {
```

```
    # [  0, .25) Up
    # [.25, .50] Right
    # [.50, .75] Down
    # [.75,   1] Left

    old_position = position
    sample = runif(1) # Could be faster with pooling

    if (sample < .25) {
      position[2] = position[2] + 1
    }
    else if (sample >= 0.25 && sample < 0.5) {
      position[1] = position[1] + 1
    }
    else if (sample >= 0.5 && sample < 0.75) {
      position[2] = position[2] - 1
    }
    else {
      position[1] = position[1] -1
    }

    # Check if invalid, if yes, revert
    if (position[1] < 1 || position[1] > m || position[2] < 1 || position[2] > m) {
      position = old_position
    }
    else {
      distance_T = distance_T + 1
    }
  }
  return(distance_T)
}


simulate_walks = function (n = 1, m = 4) {

  # Just call the function n times with grid size m
  samples = rep(m, n)
  return(sapply(samples, simulate_walk))
}
```

```
samples_4 = simulate_walks(1000, 4)
samples_8 = simulate_walks(1000, 8)

df4 = data.frame(samples_4)
df8 = data.frame(samples_8)

print(mean(samples_4 == 4*4))
```

```
## [1] 0.043
```

```
print(mean(samples_8 == 4*8))
```

```
## [1] 0.005
```

3.3 Question 3

The mean for `m = 4` is `44.206` and for `m = 8` it's 303.444.

```
mean_4 = mean(samples_4)
mean_8 = mean(samples_8)

mean_statistics = function(samples, ind) {
  return(mean(samples[ind]))
}

boot4 = boot(samples_4, mean_statistics, R = 1000)
boot8 = boot(samples_8, mean_statistics, R = 1000)
ci_boot4 = boot.ci(boot4)
```

```
## Warning in boot.ci(boot4): bootstrap variances needed for studentized
## intervals
```

```
ci_boot8 = boot.ci(boot8)
```

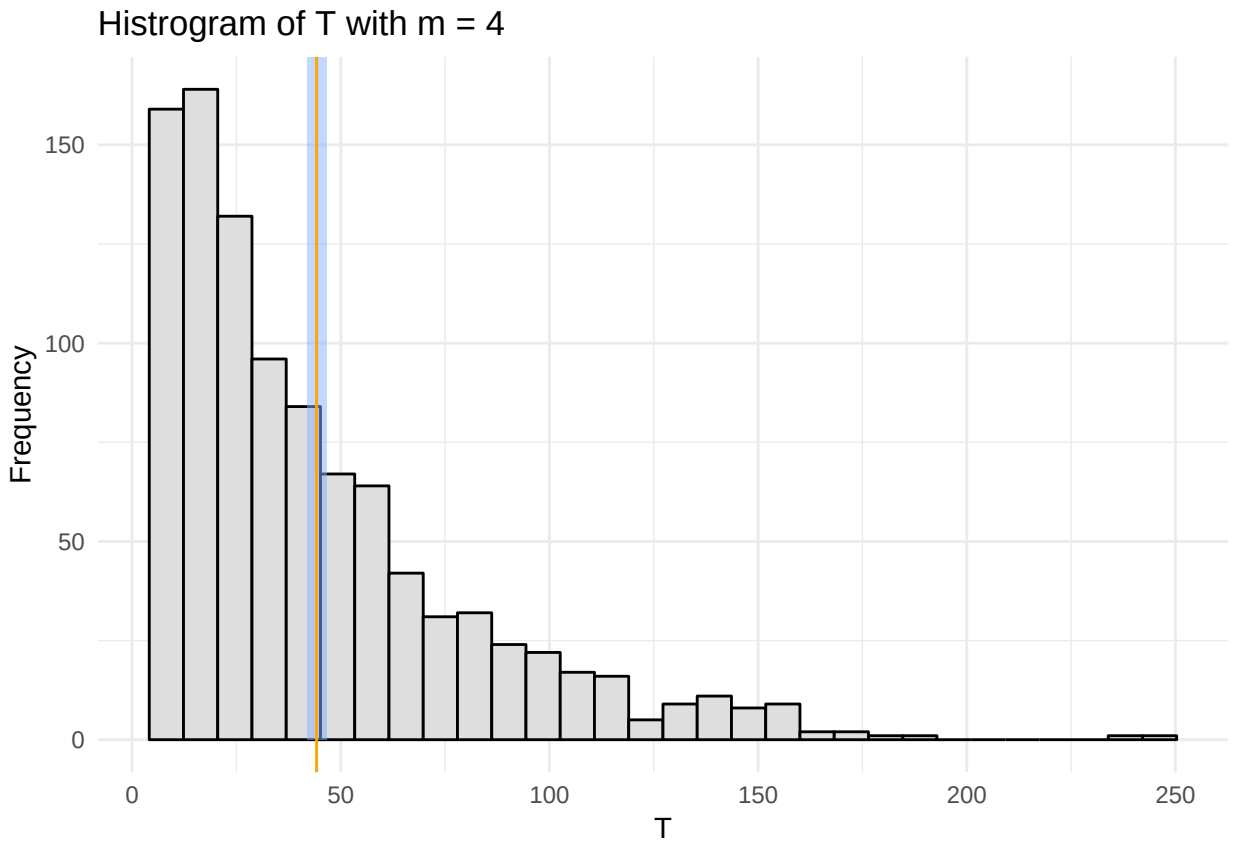
```
## Warning in boot.ci(boot8): bootstrap variances needed for studentized
## intervals
```

```
print(ci_boot4)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot4)
##
## Intervals :
## Level      Normal              Basic
## 95%   (41.96, 46.48 )   (41.77, 46.46 )
##
## Level     Percentile            BCa
## 95%   (41.95, 46.64 )   (42.08, 46.72 )
## Calculations and Intervals on Original Scale
```

```
ggplot(df4) +
  geom_histogram(aes(x = samples_4), color = "#000000", fill = "#dedede") +
  annotate("rect", xmin=ci_boot4$percent[4],
                   xmax=ci_boot4$percent[5],
                   ymin=0,
                   ymax=Inf, alpha=0.5, fill="#86b4ff") +
  geom_vline(aes(xintercept = mean_4),
             color = "orange") +
  labs(title = "Histrogram of T with m = 4", y = "Frequency", x = "T",
       color = "Legend") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
print(ci_boot8)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot8)
##
## Intervals :
## Level      Normal              Basic
## 95%   (287.7, 320.4 )   (287.1, 319.6 )
##
## Level     Percentile            BCa
## 95%   (287.3, 319.8 )   (289.5, 323.1 )
## Calculations and Intervals on Original Scale
```

```
ggplot(df4) +
  geom_histogram(aes(x = samples_8), color = "#000000", fill = "#dedede") +
  annotate("rect", xmin=ci_boot8$percent[4],
                   xmax=ci_boot8$percent[5],
                   ymin=0,
                   ymax=Inf, alpha=0.5, fill="#86b4ff") +
  geom_vline(aes(xintercept = mean_8),
             color = "orange") +
  labs(title = "Histrogram of T with m = 8", y = "Frequency", x = "T",
       color = "Legend") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```