Code appendix

$ex_1 23$

June 2024

1 Code Appendix

1.1 Exercises 2 june 7th

1.1.1 Simulate a geometric distribution

```
    ```{r}
library(gridExtra)

## Get uniform continuous

n <- 10*1000

rand_nums <- runif(n,0,1)

p <- 0.1

geo_nums <- floor(log(rand_nums)/(log(1-p)))+1

p1 <- geo_nums %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="steelblue",bins = 30) +
  scale_x_continuous(n.breaks = 25) +
  ggtitle("Transformed sample")

geo_nums2 <- (rgeom(n,p)+1) %>% as.tibble()

p2 <- geo_nums2 %>%
  ggplot(aes(value)) +
  geom_histogram(fill="steelblue",bins = 30) +
  scale_x_continuous(n.breaks = 25) +
  ggtitle("System geometric sample")

grid.arrange(p1,p2)

## Compare counts
geo_nums2 %>%
  filter(value<=5) %>%
  group_by(value) %>%
```
```

~~~
  summarise(
    n())
  )
```
~~~

1.1.2 Simulate a 6 point distribution

```
    ```{r}
## ---------------Direct crude method
start <- Sys.time()
## Define classes
classes <- 1:6

prob_vec <- c(7/48,5/48,1/8,1/16,1/4,5/16)

prob_vec %>%
  as.tibble() %>%
  rename("probs"=value) %>%
  mutate(cums = cumsum(probs))

## Setup if between cumulative probability

rand_nums %>%
  as.tibble() %>%
  mutate(
    discrete_distribution  = case_when(between(value,0,0.1458333) ~ 1,
              between(value,0.1458333,0.25) ~ 2,
              between(value,0.25,0.3750000) ~ 3,
              between(value,0.3750000 ,0.4375000 ) ~ 4,
              between(value,0.4375000 ,0.6875000 ) ~ 5,
              between(value,0.6875000 ,1) ~ 6,
    ) ) %>%
  group_by(discrete_distribution) %>%
  summarise(freq = n()/length(rand_nums)) %>%
  mutate(real_p = prob_vec) %>%
  ggplot(aes(discrete_distribution,freq)) +
  geom_bar(fill="pink",stat = "identity") +
  ggtitle("6 point distribution crude method")

## Calculate chi sqrd test

crude_table <- rand_nums %>%
  as.tibble() %>%
  mutate(
    discrete_distribution  = case_when(between(value,0,0.1458333) ~ 1,
              between(value,0.1458333,0.25) ~ 2,
              between(value,0.25,0.3750000) ~ 3,
              between(value,0.3750000 ,0.4375000 ) ~ 4,
              between(value,0.4375000 ,0.6875000 ) ~ 5,
              between(value,0.6875000 ,1) ~ 6,
    ) ) %>%
  group_by(discrete_distribution) %>%
  summarise(freq = n()/length(rand_nums)) %>%
  mutate(real_p = prob_vec)

stop_time <- Sys.time()
```
```

```
Time_comp <- stop_time-start
obs <- crude_table$freq*(10000)

exp <- crude_table$real_p*10000

stat <- sum(((obs-exp)^2)/exp)

pchisq(stat,df=6-1,lower.tail = F)

cat("computation time",Time_comp)

Time_comp
```

    ```{r}
```
------------------ Simple rejection method
n <- 10*1000
classes <- 1:6

prob_vec <- c(7/48,5/48,1/8,1/16,1/4,5/16)

c <- max(prob_vec)
samples <- c()
num_samples <- 0
start <- Sys.time()
while(num_samples<=n){
  rand_samp <- runif(2,0,1)

  I <- floor(6*rand_samp[1])+1
  if(rand_samp[2]<=prob_vec[I]/c){
    samples <- append(samples,I)
    num_samples <- length(samples)
  }
}

stop <- Sys.time()

time <- stop-start
rej_table <- samples %>%
  as.tibble() %>%
  group_by(value) %>%
  summarise(
    freq = n()/(10*1000)
) %>%
  mutate(real_p = prob_vec)

Plot observed frequencies
rej_table %>%
  as.tibble() %>%
  ggplot(aes(value,freq)) +
```
```

````
  geom_bar(fill="purple",stat="identity") +
  ggtitle("6 point distribution aquired by rejection method")

## Perform chisq test

obs <- rej_table$freq*(10000)

exp <- rej_table$real_p*10000

stat <- sum(((obs-exp)^2)/exp)

pchisq(stat,df=6-1,lower.tail = F)
time
```

```{r}
### ----------------- ALIAS METHOD
## Probabilities

prob_vec

## Generate uniform sample

U <- runif(n)

## Generate alias table and F values
start <- Sys.time()
F_vals <- 6*prob_vec

F_table <- F_vals %>%
  as.tibble() %>%
  mutate(class = 1:6) %>%
  mutate(less_than1 = F_vals<=1)

G <- F_table %>%
  filter(less_than1==F) %>%
  pull(class)

S <- F_table %>%
  filter(less_than1==T) %>%
  pull(class)

L <- 1:6

while(length(S)!=0){
  i = G[1] ; j = S[1]
  L[j]=i ;  F_vals[i] = F_vals[i]-(1-F_vals[j])
  if(F_vals[i]<1){
    G=G[-1] ; S <- append(S,i)
  }
  S = S[-1]
````

```
}

## Perform algorithm

U %>%
  as.tibble() %>%
  mutate(U_2 = runif(n)) %>%
  mutate(I = floor(6*value)+1) %>%
  mutate(output = ifelse(U_2<=F_vals[I],I,L[I])) %>%
  group_by(output) %>%
  summarise(
    freq = n()/n
  ) %>%
  ggplot(aes(output,freq)) +
  geom_bar(stat="identity",fill="blue") +
  ggtitle("6 point distribution through alias method")

stop <- Sys.time()

Time_comp <- stop-start
### Perform chi squared test

obs <- U %>%
  as.tibble() %>%
  mutate(U_2 = runif(n)) %>%
  mutate(I = floor(6*value)+1) %>%
  mutate(output = ifelse(U_2<=F_vals[I],I,L[I])) %>%
  group_by(output) %>%
  summarise(
    freq = n()/n
  ) %>%
  pull(freq)

obs <- obs*10000
exp <- prob_vec*10000
stat <- sum(((obs-exp)^2)/exp)

pchisq(stat,6-1,lower.tail = F)

‘‘‘
```

1.2 Exercises 3 June 7th

1.2.1 Generate samples from continuous distributions

```
    ‘‘‘{r}
## ----------- EXPONENTIAL DISTRIBUTION
### Generate for the following distributions.

lambda <- 3
```

````
exp_randnums <- -log(rand_nums)/3


p1 <- exp_randnums %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="steelblue") +
  ggtitle("Transformed sample of exponential distribution")

real_exp <- rexp(10000,rate=lambda)

p2 <- real_exp %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="steelblue") +
  ggtitle("system sample of exponential distribution")

grid.arrange(p1,p2)

## Test with Kolmogorov smirnov

ks.test(exp_randnums,"pexp",lambda)

```

    ```{r}
## ------------- BOX MULLER NORMAL DISTRIBUTION
n <- 10*1000
U_1 <- runif(n)
U_2 <- runif(n)

samp_data <- cbind(U_1,U_2) %>% as.tibble()

bm_norm <- samp_data %>%
  mutate(BM_normal = sqrt(-2*log(U_1))*cos(2*pi*U_2)) %>%
  pull(BM_normal)

p1 <- bm_norm %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="brown") +
  ggtitle("Normal distribution by BOX MULLER")

real_norm <- rnorm(n)

p2 <- real_norm %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="black",color="pink") +
  ggtitle("System standard normal distribution")
````

```
grid.arrange(p1,p2)

## Perform KS test

ks.test(bm_norm,"pnorm",0,1)

bm_norm
```
```

    ```{r}
beta <- 1

k_vals <- c(2.05,2.5,3,4)
plots <- list()
for(i in 1:length(k_vals)){

k <- k_vals[i]

X <- beta*(rand_nums^(-1/k))

X %>% as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram()

plots <- append(plots,p)
X %>% hist(xlim=c(0,18),breaks=1000)
}

## Construct table comparing values and creating plots.
paret_table <- k_vals %>%
  as.tibble() %>%
  group_by(value) %>%
  mutate(samp = map(value,~(rand_nums))) %>%
  mutate(pareto = map2(value,samp,~(
    (.y)^(-1/.x)
  ))) %>%
  mutate(
    plot = map2(value,pareto,~(
      .y %>%
        as.tibble() %>%
        ggplot(aes(value)) +
        geom_histogram(fill="green",color="black") +
        xlim(0,5) +
        ggtitle(paste("Pareto transformation with k=",.x))
    )
  )) %>%
  mutate(sampmean_paret = map_dbl(pareto,~(
    mean(.x)
  ))) %>%
  mutate(thrymean_paret = beta*(value/(value-1)))
```
```

```
plots <- paret_table$plot

grid.arrange(plots[[1]],
             plots[[2]],
             plots[[3]],
             plots[[4]])

## Test each transformed sample

for(i in 1:length(k_vals)){
  k_choice <- paret_table$value[i]

real_samp <- rpareto(10*1000,1,k_choice)

trans_samp <- paret_table$pareto[[i]]

p_val <- ks.test(trans_samp,real_samp)$p.value

cat("for k=",k_choice,"the p_value is = ",p_val,"    ")
}

### Comparing means

paret_table %>%
  select(sampmean_paret,thrymean_paret)

## Comparing variance

paret_table %>%
  select(value,pareto) %>%
  mutate(
    sample_var = map_dbl(pareto,~(
      var(.x)
    )
  )) %>%
  mutate(theory_var = value/(((value-1^2)*(value-2)))) %>%
  select(-pareto)
```

1.2.2 Generate 100 CI for mean and variance

```
    ```{r}
## Construct 100 confidence intervals for the mean

p1 <- 1:100 %>%
  as.tibble() %>%
  ## take sample of size 10 from box muller normal distribution 100 times
  mutate(
    sample = map(value,~(
      sample(bm_norm,size=10,replace=F)
```
```

```
    )
  )) %>%
  ## Construct lower bound for the mean
  mutate(
    lower_bound_mean = map_dbl(sample,~(
      mean(.x)-qt(0.025,df=9)*(sd(.x)/10)
    )
  )) %>%
  ## Construct upper bound for mean
  mutate(
    upper_bound_mean = map_dbl(sample,~(
      mean(.x)+qt(0.975,df=9)*(sd(.x)/10)
    )
  )) %>%
  ## Construct logical if real value is within the interval
  mutate(contains_value = between(rep(0,100),lower_bound_mean,upper_bound_mean)) %>%
  ## Create plot
  ggplot(aes(value,ymin = lower_bound_mean,ymax=upper_bound_mean)) +
  geom_linerange(aes(color=contains_value)) +
  scale_color_manual(values = c("TRUE" = "darkgreen", "FALSE" = "red")) +
  coord_flip() +
  geom_hline(yintercept = 0) +
  xlab("Interval index") +
  theme(legend.position = "none") +
  ggtitle(paste("confidence intervals for the mean"))


### Generate plot for variance
p2 <- 1:100 %>%
  as.tibble() %>%
  mutate(
    sample = map(value,~(
      sample(bm_norm,size=10,replace=T)
    )
  )) %>%
  mutate(
    lower_bound_mean = map_dbl(sample,~(
      ((10-1)*var(.x))/(qchisq(0.975,9))
    )
  )) %>%
  mutate(
    upper_bound_mean = map_dbl(sample,~(
      ((10-1)*var(.x))/(qchisq(0.025,9))
    )
  )) %>%
  mutate(contains_value = between(rep(1,100),lower_bound_mean,upper_bound_mean)) %>%
  ggplot(aes(value,ymin = lower_bound_mean,ymax=upper_bound_mean)) +
  geom_linerange(aes(color=contains_value)) +
  scale_color_manual(values = c("TRUE" = "darkgreen", "FALSE" = "red")) +
  coord_flip() +
  geom_hline(yintercept = 1) +
```

```
  xlab("") +
  theme(legend.position = "none") +
  ggtitle(paste("confidence intervals for the variance"))

  grid.arrange(p1,p2,ncol=2)

‘‘‘
```

1.2.3 Simulate through pareto distribution

```
    ‘‘‘{r,warning=F}
## -------- Pareto distribution through composition

n <- 10*1000
### Generate Y through exponential distribution

mu <- 5

Y <- rexp(n,rate=mu)

X <- rexp(n,rate=Y)

## Add mu to X so support is X>=beta

comp_paret <- X + mu

## Thus X should follow a Pareto distribution with location = 5

### Compare by histograms

p1 <- comp_paret %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  geom_histogram(fill="darkgreen",color="black") +
  xlim(mu,10) +
  ggtitle("Pareto distribution through composition sampling")

paret_realsamp <-  rpareto(n,location = mu,shape = 1)

p2 <- paret_realsamp %>%
  as.tibble() %>%
  ggplot(aes(value)) +
  xlim(mu,10) +
  geom_histogram(fill="darkgreen",color="black") +
  ggtitle("Actual sample from Pareto distribution")

grid.arrange(p1,p2)

### Test the sample against actual sample

ks.test(comp_paret,paret_realsamp)
```

‘ ‘ ‘