Ayudantía Funciones y Estructuras de Control

Natalie Julian & Javiera Preuss

Paquetes a utilizar

```
#install.packages("randomNames")
library(randomNames)
library(dplyr)
library(ggplot2)
library(readr)
library(purrr)
```

Pregunta 1

Considere los siguientes datos ficticios en el cual se posee por alumno el registro de calificaciones de las interrogaciones y promedio de laboratorio para un curso de la universidad:

```
set.seed(2021) #Fija semilla 2021
nombres<-randomNames(50, ethnicity = 4) #Generador de nombres aleatorios hispánicos

set.seed(2021) #Fija semilla 2021
interrogaciones<-matrix(round(runif(150, min=1, max=7),1), ncol=3) #Crea una matriz con cali

laboratorio<-round(runif(nrow(interrogaciones), min=4.5, max=7),1)
```

```
data<-data.frame(nombres, interrogaciones, laboratorio)
names(data)<-c("Nombre", "I1", "I2", "I3", "prom_laboratorio")
```

La nota de presentación al examen se calcula como sigue:

$$NP = \bar{I} \times 70\% + \bar{L} \times 30\%$$

Donde $\bar{I}$ corresponde al promedio de las tres interrogaciones y $\bar{L}$ al promedio de laboratorio.

Ítem a)

En base a la nota de presentación se tienen los siguientes casos:

- Si $NP \geq 5,0$ y *todas* las interrogaciones son mayores o iguales a 4.0 el alumno aprueba el curso con nota la nota de presentación a examen.
- Si $NP < 3,95$ el alumno reprueba con opción para rendir examen y subir la nota final del curso.
- Todos los otros casos rinden examen.

En este contexto, sería de mucha utilidad tener una función que reciba la data.frame y entregue (en objetos separados) las calificaciones y el resultado de presentación al examen (Aprueba sin examen, Reprueba con opción a rendir examen, Rinde examen) incluyendo el nombre del alumno. Explore distintas maneras de crear la función deseada con ayuda de las funciones vistas en clase y compare el tiempo computacional requerido de cada una usando la función *system.time()*.

```
#### FORMA 1: UTILIZANDO FOR PARA RECORRER CADA FILA E IF, IF ELSE Y ELSE PARA CADA CONDICIÓ

resultado_examen1<-function(df){
```

```
  df$NP<-round(((df$I1+df$I2+df$I3)/3)*0.7+df$prom_laboratorio*0.3,2)
  df$resultado<-NA

  for(i in 1:nrow(df)){
  if((df$NP[i]>=5)&(df$I1[i]>4)&(df$I2[i]>4)&(df$I3[i]>4)){
  df$resultado[i]<-"Aprueba sin examen"
  }
    else if(df$NP[i]<3.95){
      df$resultado[i]<-"Reprueba con opción de rendir examen"
    }
    else {
      df$resultado[i]<-"Rinde examen"
    }
  }
  return(list("Notas"=df[, 1:5],"Status"=data.frame("Nombre"=df$Nombre, "Status"=df$resultad

  }
```

```
resultado_examen1(data)
```

```
#### FORMA 2: UTILIZANDO APPLY

resultado_examen2<-function(df){

df$NP<-apply(df[,-1], MARGIN = 1, FUN = function(x) {
  (x[1] + x[2] + x[3])/3*0.7 + x[4]*0.3})
```

```
df$resultado<-ifelse((df$NP>=5)&(df$I1>4)&(df$I2>4)&(df$I3>4), "Aprueba sin examen",
                           ifelse((df$NP<3.95), "Reprueba con opción de rendir examen",
                                  "Rinde examen"))

  return(list("Notas"=df[, 1:5],"Status"=data.frame("Nombre"=df$Nombre, "Status"=df$resultad

  }
```

```
resultado_examen2(data)
```

```
#### FORMA 3: Usando dplyr a secas

resultado_examen3<-function(df){

df<-df%>%
    mutate(NP=round(((I1+I2+I3)/3)*0.7+prom_laboratorio*0.3,2),
           resultado=ifelse((NP>=5)&(I1>4)&(I2>4)&(I3>4), "Aprueba sin examen",
                           ifelse((NP<3.95), "Reprueba con opción de rendir examen",
                                  "Rinde examen")))

  return(list("Notas"=df[, 1:5],"Status"=data.frame("Nombre"=df$Nombre, "Status"=df$resultad

  }
```

```
resultado_examen3(data)
```

```
#Probar con distintos tamaños en la bbdd de calificaciones para ver las diferencias más noto

system.time(resultado_examen1(data))
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
system.time(resultado_examen2(data))
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

```
system.time(resultado_examen3(data))
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
#elapsed es el tiempo total que tomó ejecutar la función

#user consiste en el tiempo ejecutado en la sesión actual
#system consiste en el tiempo ejecutado por el sistema operativo (si se abren archivos u se
```

Ítem b) Evalúe la data.frame antes creada en la función creada, ¿cuántos alumnos aprobaron sin examen? Visualice esta información en un gráfico.

```
resultado_examen3(data)
```

```
head(resultado_examen3(data)$Status)
```

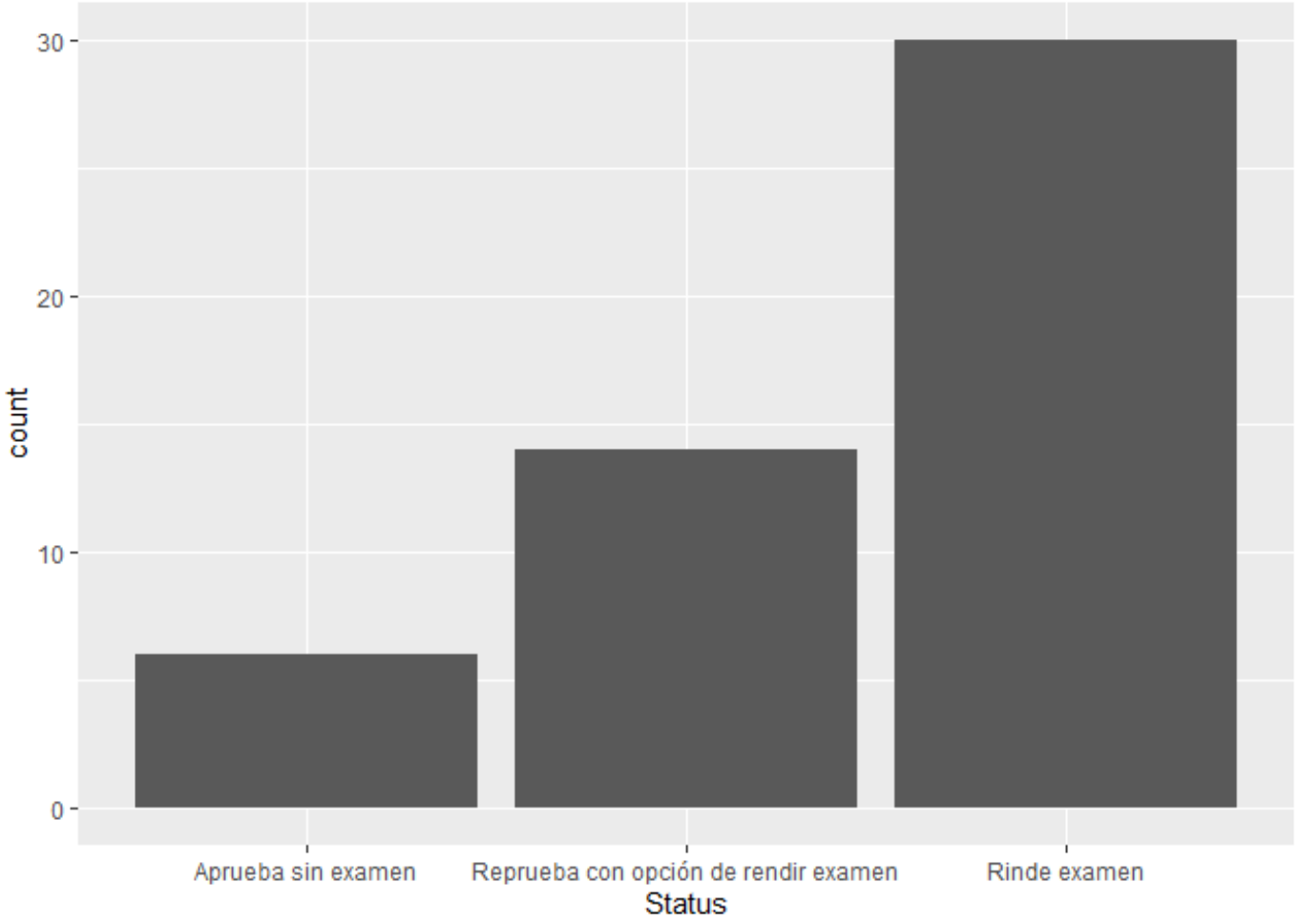
```
##                       Nombre                                Status
## 1            Rivera, Darius                          Rinde examen
## 2            Aguirre, Leyba                          Rinde examen
## 3         Rodriguez, Mariah Reprueba con opción de rendir examen
## 4   Castillo Meraz, Yaneisi                          Rinde examen
## 5 Sanchez Gonzalez, Jonathan                          Rinde examen
## 6       Restrepo, Stephanie                          Rinde examen
```

```
table(resultado_examen3(data)$Status[, 2])
```

```
##
##                   Aprueba sin examen Reprueba con opción de rendir examen
##                                    6                                    14
##                         Rinde examen
##                                   30
```

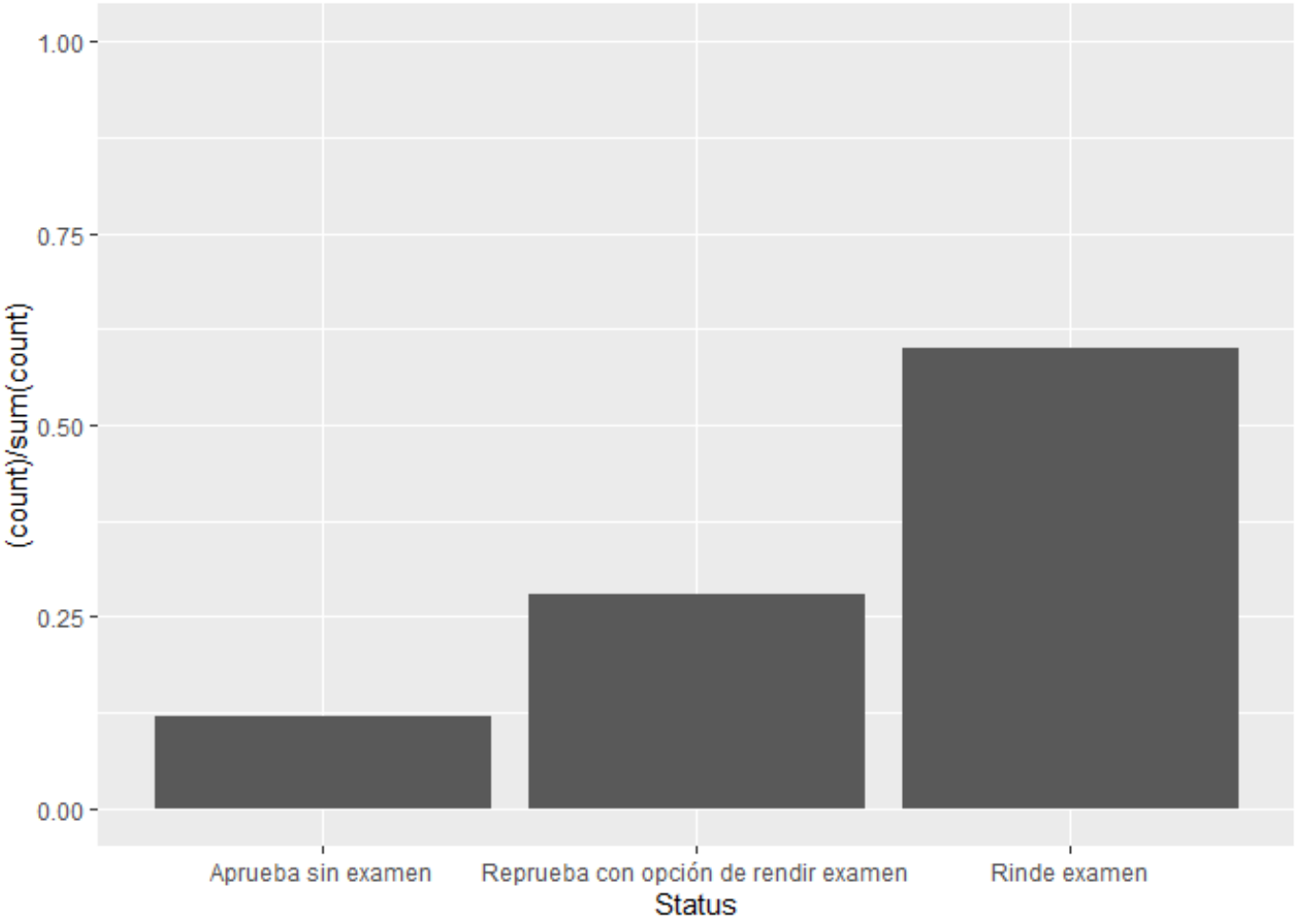
6 personas aprobaron sin rendir examen, equivalente al 12% del curso.

```r
library(ggplot2)
#Gráfico de frecuencias (cantidad de alumnos)
ggplot(resultado_examen3(data)$Status, aes(Status))+
  geom_bar()
```




```r
#Gráfico de porcentajes
ggplot(resultado_examen3(data)$Status, aes(Status))+
```

```
  geom_bar(aes(y = (..count..)/sum(..count..))) +ylim(0, 1)
```



Pregunta 2

La secuencia de Fibonacci es una sucesión definida por recurrencia. Esto significa que para calcular un término de la sucesión se necesitan los términos que le preceden.

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, \ldots$$

Ítem a) Cree dos funciones distintas que retornen el n-ésimo valor de la serie de Fibonacci.

```
#Solución 1

fib <- function(n) {

  if (n == 1) {  #Proceso Ramificado

    return(0)
  }
  else if(n == 2) {

    return(1)
  }
  else if(n > 2) {

    return(fib(n - 1) + fib(n - 2))  #Proceso Autorecursivo
  }

}


fib(4)
```

```
## [1] 2
```

```
fib(10)
```

```
## [1] 34
```

```
#Solucion 2

fib2 <- function(n){
  valores <- c()
  valores[1] <- 0
  valores[2] <- 1
  valores[3]<-1

  i<-3   #Se define un contador apropiado
  while(i<=n){      #Proceso condicional
    next_val <- sum(tail(valores,2) )
    valores <- c(valores,next_val)
    i<-i+1
  }
  return(valores[n])
}

fib2(4)
```

```
## [1] 2
```

```
fib2(10)
```

```
## [1] 34
```

Ítem b) Compare los tiempos de demora de ejecución de ambas funciones utilizando la función system.time(). Explique por qué cree que se generan los resultados obtenidos.

```
system.time(fib(30))
```

```
##    user  system elapsed
##    1.25    0.00    1.26
```

```
system.time(fib2(30))
```

```
##    user  system elapsed
##       0       0       0
```

Naturalmente la función 1 tardará más, ya que es un proceso que se requiere a sí misma, por lo cual entra dentro de sí en cada iteración, lo cual tarda más tiempo.

Pregunta 3

Instacart es una empresa que ofrece servicios de delivery de alimentos en los Estados Unidos y Canadá. Los usuarios seleccionan los productos del despacho a través de su sitio web o de la aplicación móvil. La información de los pedidos se encuentra en las bases de datos:

Nombre tabla	Descripción
departments	Indica el departamento que provee el producto (ejemplo: pets, frozen, bakery, etcétera)

Nombre tabla	Descripción
order products train	Contiene por orden (pedido) los productos despachados
products	Contiene información de los productos, nombre de los productos e ID de los productos

Ítem a) Cargue las bases de datos y realice los cruces necesarios para tener para cada pedido la información completa (características del producto y departamento a cargo de proveerlo).

```
setwd("C:/Users/Naty/Desktop/Clases UC/Diplomado Data Science 2021/Clases/Ayudantía 12 de Ju


#Obtiene un vector con el nombre de los archivos csv en el directorio

nombres<- list.files(path = getwd(),
                     pattern = "\\.csv$",
                     full.names = FALSE)


Datas<-lapply(nombres,"read_csv")  #Carga todos los archivos csv en el directorio

names(Datas)<-substr(nombres,1,nchar(nombres)-4) #Añade nombre a cada data
```

```
##Cruces necesarios:

cruce<-Datas$order_products__train%>%
```

```
  left_join(., Datas$products, by="product_id")%>%
  left_join(., Datas$departments, by="department_id")


head(cruce, 5)
```

```
## # A tibble: 5 x 8
##   order_id product_id add_to_cart_order reordered product_name          aisle_id
##      <dbl>      <dbl>              <dbl>     <dbl> <chr>                   <dbl>
## 1        1      49302                  1         1 Bulgarian Yogurt          120
## 2        1      11109                  2         1 Organic 4% Milk Fat ~     108
## 3        1      10246                  3         0 Organic Celery Hearts      83
## 4        1      49683                  4         0 Cucumber Kirby             83
## 5        1      43633                  5         1 Lightly Smoked Sardi~      95
## # ... with 2 more variables: department_id <dbl>, department <chr>
```

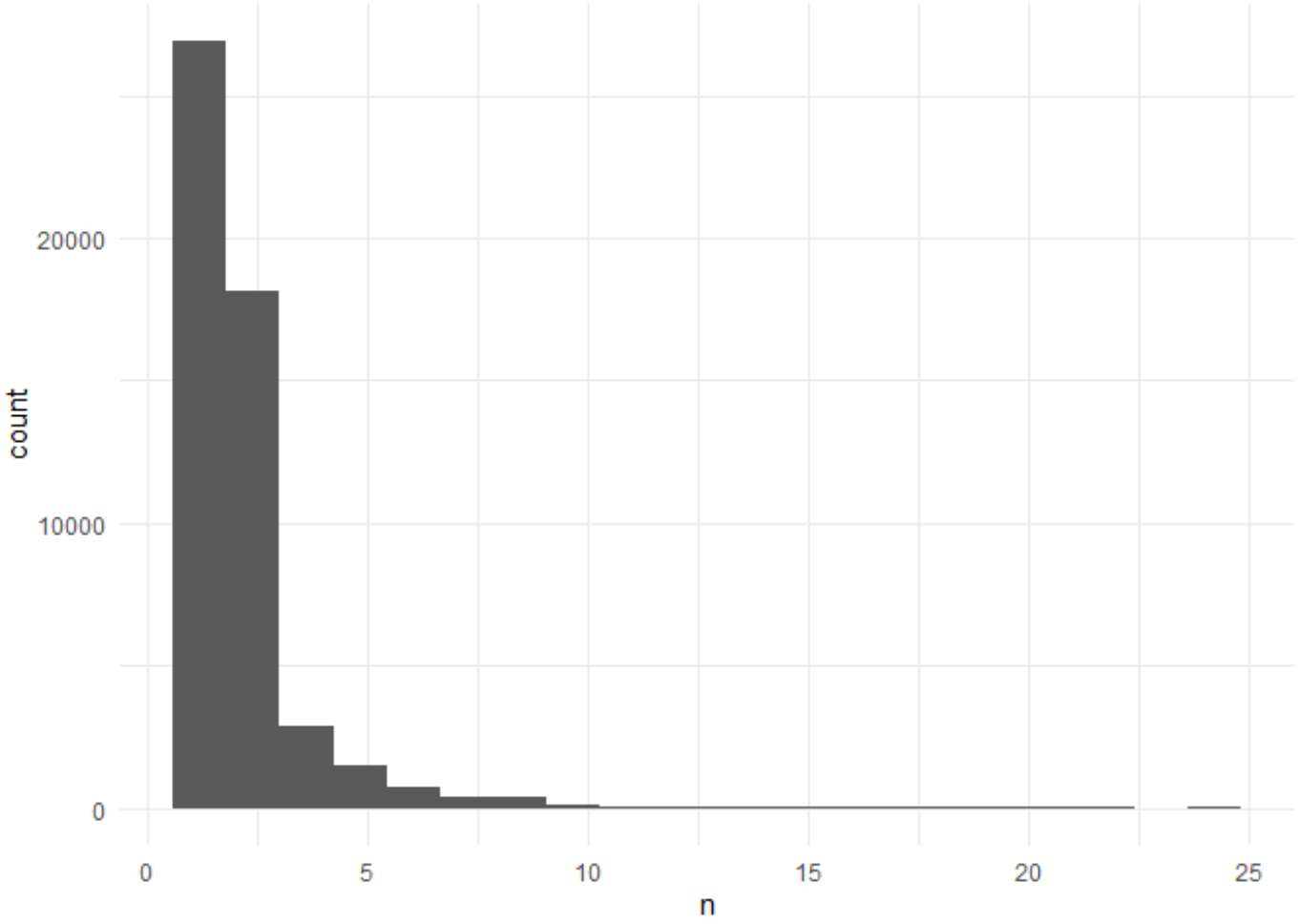
Ítem b) Para el departamento *frozen* obtenga por pedido, la cantidad de productos del pedido (cada fila corresponde a un producto, por ende se debe realizar un conteo de la cantidad de filas por id del pedido) y realice un histograma de la cantidad de productos por pedido.

```
filtrofrozen<-cruce%>%
  filter(department=="frozen")%>%
  group_by(order_id)%>%
  count()

head(filtrofrozen, 5)
```

```
## # A tibble: 5 x 2
## # Groups:   order_id [5]
##   order_id     n
##      <dbl> <int>
## 1       38     1
## 2       96     2
## 3       98     3
## 4      456     1
## 5      473     1
```

```
ggplot(filtrofrozen, aes(n))+
  geom_histogram(bins=20)+
  theme_minimal()
```



Ítem c) Interesa tener la misma información anterior para los departamentos *pets, breakfast, frozen, produce*, de modo que cada departamento pueda recibir y accionar sus propios resultados. Utilice la función *map* para replicar lo anterior a cada departamento.

```
datos<-map(c("pets", "breakfast", "frozen", "produce"),
    function(d){
    filtro<-cruce%>%
  filter(department==d)%>%
```

```
  group_by(order_id)%>%
  count()
  return(filtro)
    })
```

```
#Podemos incluso retornar el gráfico

plots<-map(c("pets", "breakfast", "frozen", "produce"),
    function(d){
    filtro<-cruce%>%
  filter(department==d)%>%
  group_by(order_id)%>%
  count()


  return(ggplot(filtro, aes(n))+
  geom_histogram(bins=20)+
  theme_minimal() +
    ggtitle(paste("Products delivered per order in category", d)))
    })
```

Ítem d) Con la función *ggsave()* es posible guardar cada gráfico creado en el apartado anterior en formato png. Utilice la función *map()* para exportar los gráficos.

```
map(
  seq_along(plots),
  function(g) {
    ggsave(sprintf("category %s.png", g), plots[[g]])
```

```
  }
)
```

```
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```