08-Tarea

1. ENIGH

Para este ejercicio usaremos los datos de la ENIGH 2014. En particular las variables alimentos, vestido, vivienda, salud, comunica, educacion y esparci (esparcimiento) que indican el gasto trimestral en cada una de las categorías.

1. Calcula los deciles de ingreso usando la variable de ingreso corriente (ing_cor).

Debes tomar en cuenta el diseño de la muestra, puedes usar la función `survey_quantile()` del paquete `srvyr` o `svyquantile()` del paquete `survey`. Reporta las estimaciones y sus errores estándar usando el bootstrap de Rao y Wu.

```
## R libraries
library(readr)
library(dplyr)
library(Hmisc)
library(srvyr)
library(gridExtra)
library(tidyr)
```

```
## Importing data
concentrado_hogar <- read_csv("concentradohogar.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   folioviv = col_character(),
##   ubica_geo = col_character(),
##   ageb = col_character(),
##   est_dis = col_character(),
##   upm = col_character(),
##   educa_jefe = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
hogar <- concentrado_hogar %>%
    select(folioviv, foliohog, est_dis, upm, factor_hog, ing_cor, alimentos,
        vestido, vivienda, salud, transporte, comunica, educacion, esparci)
```

2. Crea una nueva variable que indique el decil de ingreso para cada hogar. Tips: 1) una función que puede resultar útil es `cut2()` (de `Hmisc`),

2) si usas el paquete `srvyr` puedes usar `mutate()` sobre el objeto `survey` con pesos de replicaciones bootstrap.

```
## Create new column with the corresponding decile based on the "ing_cor" column
hogar <- hogar %>%
  mutate(ing_cor_decil = ntile(ing_cor, 10)/10) %>%
  filter(ing_cor > 0)
```

```
## Proportions by decile
hogar %>%
  group_by(ing_cor_decil) %>%
  tally()
```

```
## # A tibble: 10 x 2
##    ing_cor_decil     n
##            <dbl> <int>
##  1           0.1  1947
##  2           0.2  1948
##  3           0.3  1948
##  4           0.4  1948
##  5           0.5  1948
##  6           0.6  1948
##  7           0.7  1948
##  8           0.8  1948
##  9           0.9  1948
## 10           1    1947
```

```
hogar
```

```
## # A tibble: 19,478 x 15
##    folioviv foliohog est_dis upm   factor_hog ing_cor alimentos vestido vivienda
##    <chr>       <dbl> <chr>   <chr>      <dbl>   <dbl>     <dbl>   <dbl>    <dbl>
##  1 0100008~        1 005     00670        694  39787.    11173.     0      2199.
##  2 0100008~        1 005     00670        694  19524.     1941.     0         0
##  3 0100008~        1 005     00670        694  99258.    16759.   489.     2989.
##  4 0100008~        1 005     00670        694  87884.     8678.   293.     1100
##  5 0100010~        1 005     00600        660  84427.    12873.     0      3062.
##  6 0100010~        1 005     00600        660 232014.    18530.  4520.     1580
##  7 0100010~        1 005     00600        660  90941.    10819.     0      5090.
##  8 0100010~        1 005     00600        660  70309.    12491.     0      1755
##  9 0100010~        1 005     00600        660  92508.     6853.     0     15456.
## 10 0100018~        1 004     00570        583  80865.    12876.    84.1    2053
## # ... with 19,468 more rows, and 6 more variables: salud <dbl>,
## #   transporte <dbl>, comunica <dbl>, educacion <dbl>, esparci <dbl>,
## #   ing_cor_decil <dbl>
```

3. Estima para cada decil, el porcentaje del gasto en cada categoría (), reporta el error estándar de las estimaciones, usa el bootstrap de Rao y Wu. Tip: 1) agrega una variable que indica para cada hogar el porcentaje de gasto en cada categoría, 2) si usas srvyr puedes usar la función `group_by()` para estimar la media del porcentaje de gasto por decil.

```
## Calculating spenditure per category
rel_cols <- c("alimentos", "vestido", "vivienda", "salud", "transporte", "comunica", "educacion", "espa
hogar_sp <- hogar

for (col in rel_cols){

  ## Define name of new column
  col_mod <- paste(col, "_sp", sep="")

  ## Fill new column with values
  hogar_sp <- hogar_sp %>%
    mutate(!!sym(col_mod) := !!sym(col)/ing_cor*100) %>%
```

```
    select(-!!sym(col))

}

hogar_sp
```

```
## # A tibble: 19,478 x 15
##    folioviv foliohog est_dis upm   factor_hog ing_cor ing_cor_decil alimentos_sp
##    <chr>       <dbl> <chr>   <chr>      <dbl>   <dbl>         <dbl>        <dbl>
##  1 0100008~        1 005     00670        694  39787.           0.8        28.1
##  2 0100008~        1 005     00670        694  19524.           0.4         9.94
##  3 0100008~        1 005     00670        694  99258.           1          16.9
##  4 0100008~        1 005     00670        694  87884.           1           9.87
##  5 0100010~        1 005     00600        660  84427.           1          15.2
##  6 0100010~        1 005     00600        660 232014.           1           7.99
##  7 0100010~        1 005     00600        660  90941.           1          11.9
##  8 0100010~        1 005     00600        660  70309.           0.9        17.8
##  9 0100010~        1 005     00600        660  92508.           1           7.41
## 10 0100018~        1 004     00570        583  80865.           1          15.9
## # ... with 19,468 more rows, and 7 more variables: vestido_sp <dbl>,
## #   vivienda_sp <dbl>, salud_sp <dbl>, transporte_sp <dbl>, comunica_sp <dbl>,
## #   educacion_sp <dbl>, esparci_sp <dbl>
```

```
## Applying RaoWu bootstrap with survey and srvyr libraries

#### Defining survey design
enigh_design <- hogar_sp %>%
  as_survey_design(ids=upm, weights=factor_hog, strata=ing_cor_decil, nest=TRUE)

#### Selecting bootstrap as method to calculate standard error
set.seed(1)
enigh_boot <- enigh_design %>%
  as_survey_rep(type="subbootstrap", replicates=500)

#### Calculating mean for al income expenditures
datalist <- list()
i <- 1
for (col in rel_cols){

  ## Creating names of the columns that will be added
  col_name <- paste(col, "_msp", sep="")
  col_name_se <- paste(col, "_msp_se", sep="")
  col_name_li <- paste(col, "_li", sep="")
  col_name_ls <- paste(col, "_ls", sep="")
  col_var <- paste(col, "_sp", sep="")

  ## Iterating over each concept to obtain the bootstrap result and sotring it in a list
  if (i == 1){

    datalist[[i]] <- enigh_boot %>%
      group_by(ing_cor_decil) %>%
      srvyr::summarise(!!sym(col_name) := survey_mean(!!sym(col_var))) %>%
      mutate(
        !!sym(col_name_li) := !!sym(col_name) - !!sym(col_name_se),
```

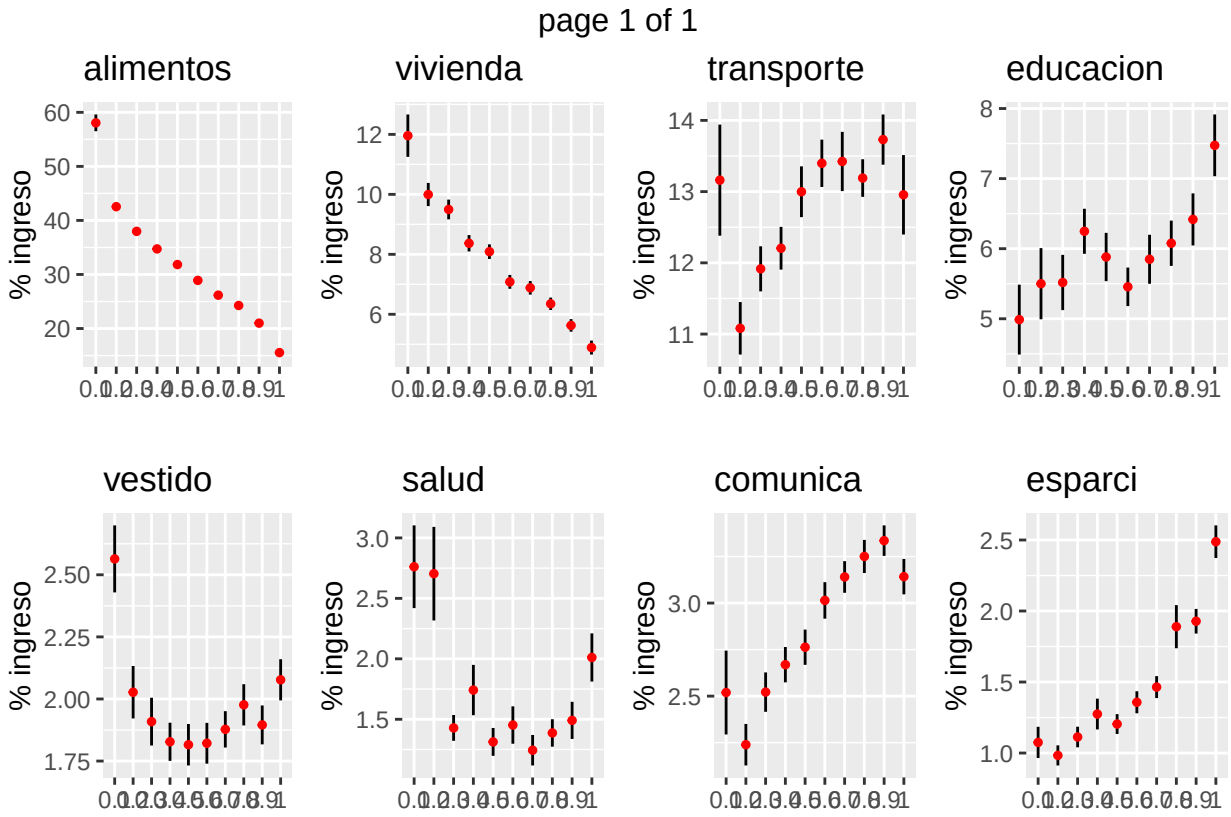
```
        !!sym(col_name_ls) := !!sym(col_name) + !!sym(col_name_se)
        ) %>%
      select(-!!sym(col_name_se)) %>%
      mutate(ing_cor_decil = as.factor(ing_cor_decil))

  } else{

    datalist[[i]] <- enigh_boot %>%
      group_by(ing_cor_decil) %>%
      srvyr::summarise(!!sym(col_name) := survey_mean(!!sym(col_var))) %>%
      mutate(
        !!sym(col_name_li) := !!sym(col_name) - !!sym(col_name_se),
        !!sym(col_name_ls) := !!sym(col_name) + !!sym(col_name_se)
        ) %>%
      select(-!!sym(col_name_se)) %>%
      select(-ing_cor_decil)

  }

  i <- i + 1

}

RaoWu_res <- dplyr::bind_cols(datalist)
RaoWu_res
```

```
## # A tibble: 10 x 25
##    ing_cor_decil alimentos_msp alimentos_li alimentos_ls vestido_msp vestido_li
##    <fct>                 <dbl>        <dbl>        <dbl>       <dbl>      <dbl>
##  1 0.1                    58.1         56.5         59.6        2.56       2.43
##  2 0.2                    42.5         41.8         43.3        2.03       1.92
##  3 0.3                    38.0         37.4         38.6        1.91       1.81
##  4 0.4                    34.7         34.2         35.2        1.83       1.75
##  5 0.5                    31.8         31.3         32.3        1.82       1.73
##  6 0.6                    28.9         28.4         29.4        1.82       1.74
##  7 0.7                    26.2         25.8         26.6        1.88       1.80
##  8 0.8                    24.3         23.9         24.6        1.98       1.89
##  9 0.9                    21.0         20.7         21.3        1.90       1.82
## 10 1                      15.5         15.2         15.9        2.08       1.99
## # ... with 19 more variables: vestido_ls <dbl>, vivienda_msp <dbl>,
## #   vivienda_li <dbl>, vivienda_ls <dbl>, salud_msp <dbl>, salud_li <dbl>,
## #   salud_ls <dbl>, transporte_msp <dbl>, transporte_li <dbl>,
## #   transporte_ls <dbl>, comunica_msp <dbl>, comunica_li <dbl>,
## #   comunica_ls <dbl>, educacion_msp <dbl>, educacion_li <dbl>,
## #   educacion_ls <dbl>, esparci_msp <dbl>, esparci_li <dbl>, esparci_ls <dbl>
```

4. Realiza una gráfica con las estimaciones del paso 3.

```
plot_list = list()
i <- 1

for (act in rel_cols){

  plot_data <- RaoWu_res %>%
    select(ing_cor_decil, starts_with(act))
```

```
  plot <- ggplot(
    plot_data,
    aes_string(
      x="ing_cor_decil",
      y=paste(act, "_msp", sep=""),
      ymin=paste(act, "_li", sep=""),
      ymax=paste(act, "_ls", sep="")
    )
  ) +
    geom_linerange() +
    geom_point(colour="red", size=1) +
    ggtitle(act) +
    xlab("") +
    ylab("% ingreso")

  plot_list[[i]] = plot
  i <- i + 1

}

marrangeGrob(plot_list, nrow=2, ncol=4)
```



2. Componentes Principales

Los datos *marks* (Mardia, Kent y Bibby, 1979) contienen los puntajes de 88 estudiantes en 5 pruebas: mecánica, vectores, álgebra, análisis y estadística. Cada renglón corresponde a la calificación de un estudiante

en cada prueba. Para este ejercicio no es necesario que conozcas componentes principales pues puedes implementar el bootstrap siguiendo el código propuesto y discutiremos los detalles del análisis en la próxima clase.

```
data(marks, package = "ggm")
glimpse(marks)
```

```
## Rows: 88
## Columns: 5
## $ mechanics  <dbl> 77, 63, 75, 55, 63, 53, 51, 59, 62, 64, 52, 55, 50, 65, ...
## $ vectors    <dbl> 82, 78, 73, 72, 63, 61, 67, 70, 60, 72, 64, 67, 50, 63, ...
## $ algebra    <dbl> 67, 80, 71, 63, 65, 72, 65, 68, 58, 60, 60, 59, 64, 58, ...
## $ analysis   <dbl> 67, 70, 66, 70, 70, 64, 65, 62, 62, 62, 63, 62, 55, 56, ...
## $ statistics <dbl> 81, 81, 81, 68, 63, 73, 68, 56, 70, 45, 54, 44, 63, 37, ...
```

Un análisis de componentes principales proseguiría como sigue:

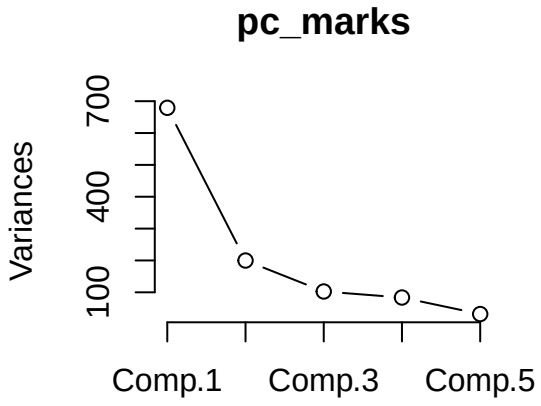
```
pc_marks <- princomp(marks)
summary(pc_marks)
```

```
## Importance of components:
##                           Comp.1     Comp.2      Comp.3     Comp.4     Comp.5
## Standard deviation     26.061142 14.1355705 10.12760414 9.14706148 5.63807655
## Proportion of Variance  0.619115  0.1821424  0.09349705 0.07626893 0.02897653
## Cumulative Proportion   0.619115  0.8012575  0.89475453 0.97102347 1.00000000
```

```
loadings(pc_marks)
```

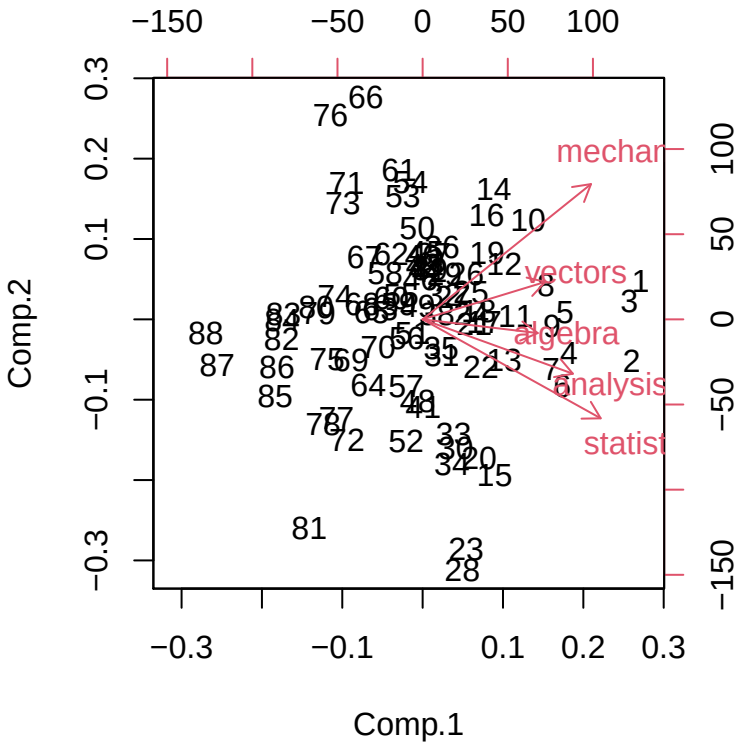
```
##
## Loadings:
##            Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## mechanics   0.505  0.749  0.300  0.296
## vectors     0.368  0.207 -0.416 -0.783  0.189
## algebra     0.346        -0.145        -0.924
## analysis    0.451 -0.301 -0.597  0.518  0.286
## statistics  0.535 -0.548  0.600 -0.176  0.151
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## SS loadings       1.0    1.0    1.0    1.0    1.0
## Proportion Var    0.2    0.2    0.2    0.2    0.2
## Cumulative Var    0.2    0.4    0.6    0.8    1.0
```

```
plot(pc_marks, type = "lines")
```



Y graficamos:

```
biplot(pc_marks)
```



Los cálculos de un análisis de componentes principales involucran la matriz de covarianzas empírica G (estimaciones *plug-in*)

$$G_{jk} = \frac{1}{88}\sum_{i=1}^{8} 8(x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k})$$

para $j, k = 1, 2, 3, 4, 5$, y donde $\bar{x_j} = \sum_{i=1}^{8} 8x_{ij}/88$ (la media de la i-ésima columna).

```
G <- cov(marks) * 87 / 88
G
```

```
##            mechanics   vectors   algebra  analysis statistics
## mechanics   302.2934 125.77686 100.42510 105.06508  116.07076
## vectors     125.7769 170.87810  84.18957  93.59711   97.88688
## algebra     100.4251  84.18957 111.60318 110.83936  120.48567
## analysis    105.0651  93.59711 110.83936 217.87603  153.76808
## statistics  116.0708  97.88688 120.48567 153.76808  294.37177
```

Los *pesos* y las *componentes principales* no son mas que los eigenvalores y eigenvectores de la matriz de covarianzas G, estos se calculan a través de una serie de de manipulaciones algebraicas que requieren cálculos del orden de p^3 (cuando G es una matriz de tamaño p×p).

```
eigen_G <- eigen(G)
lambda <- eigen_G$values
v <- eigen_G$vectors
lambda
```

```
## [1] 679.18311 199.81435 102.56837  83.66873  31.78791
```

```
v
```

```
##            [,1]        [,2]       [,3]         [,4]        [,5]
## [1,] -0.5054457  0.74874751  0.2997888  0.296184264 -0.07939388
## [2,] -0.3683486  0.20740314 -0.4155900 -0.782888173 -0.18887639
## [3,] -0.3456612 -0.07590813 -0.1453182 -0.003236339  0.92392015
## [4,] -0.4511226 -0.30088849 -0.5966265  0.518139724 -0.28552169
## [5,] -0.5346501 -0.54778205  0.6002758 -0.175732020 -0.15123239
```

1. Proponemos el siguiente modelo simple para puntajes correlacionados:

$$\mathbf{x}_i = Q_i \mathbf{v}$$

donde $\mathbf{x}_i$ es la tupla de calificaciones del i-ésimo estudiante, Q_i es un número que representa la habilidad del estudiante y $\mathbf{v}$ es un vector fijo con 5 números que aplica a todos los estudiantes. Si este modelo simple fuera cierto, entonces únicamente el $\hat{\lambda}_1$ sería positivo y $\mathbf{v} = \hat{v}_1$. Sea

$$\hat{\theta} = \sum_{i=1}^{5} \hat{\lambda}_i$$

el modelo propuesto es equivalente a $\hat{\theta} = 1$, inculso si el modelo es correcto, no esperamos que $\hat{\theta}$ sea exactamente uno pues hay ruido en los datos.

```
theta_hat <- lambda[1]/sum(lambda)
theta_hat
```

```
## [1] 0.619115
```

El valor de $\hat{\theta}$ mide el porcentaje de la varianza explicada en la primer componente principal, ¿qué tan preciso es $\hat{\theta}$? La complejidad matemática en el cálculo de $\hat{\theta}$ es irrelevante siempre y cuando podamos calcular $\hat{\theta}^*$ para una muestra bootstrap, en esta caso una muestra bootsrtap es una base de datos de 88×5 $\mathbf{X}^*$, donde las filas $\mathbf{x_i}^*$ de $\mathbf{X}^*$ son una muestra aleatoria de tamaño 88 de la verdadera matriz de datos.

1. Utiliza bootstrap para calcular el error estándar de $\hat{\theta}$.

```
## Simulating lambdas
boot_lambda = list()
for (i in c(1: 500)) {

  boot_sample <- marks[sample(nrow(marks), nrow(marks), replace=TRUE), ]

  boot_lambda_tmp <- tibble(boot_lambda = eigen(cov(boot_sample)*87/88)$values) %>%
    mutate(
      !!sym(toString(i)) := boot_lambda/sum(boot_lambda),
      lambda = sprintf("lambda_%s", seq(1:5))
      ) %>%
    select(-boot_lambda) %>%
    pivot_wider(names_from=lambda, values_from=!!sym(toString(i)))

  boot_lambda[[i]] = boot_lambda_tmp

}

## Binding results
boot_lambda <- bind_rows(boot_lambda)
boot_lambda
```

```
## # A tibble: 500 x 5
##    lambda_1 lambda_2 lambda_3 lambda_4 lambda_5
##       <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
##  1    0.580    0.216   0.114    0.0655   0.0251
##  2    0.596    0.223   0.0982   0.0604   0.0217
##  3    0.599    0.201   0.0927   0.0815   0.0262
##  4    0.627    0.198   0.103    0.0515   0.0206
##  5    0.577    0.211   0.116    0.0698   0.0268
##  6    0.640    0.175   0.0880   0.0703   0.0265
##  7    0.621    0.215   0.0739   0.0610   0.0296
##  8    0.517    0.204   0.140    0.110    0.0297
##  9    0.656    0.151   0.0907   0.0804   0.0220
## 10    0.552    0.204   0.111    0.101    0.0314
## # ... with 490 more rows
```

2. Grafica la distribución bootstrap.

```
ggplot(
  boot_lambda,
  aes(x=boot_lambda$lambda_1)
) +
  geom_histogram()
```

```
## Warning: Use of `boot_lambda$lambda_1` is discouraged. Use `lambda_1` instead.
```

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