**Chapter 2**

**Exploring Data with R**

At the beginning of each session you need to load the *packages* and there are two ways. One way is using "load packages" from the "packages" window. Otherwise you can write from prompt:

library(foreign) # required to import data created in other programs

library(Hmisc) # provides useful functions for data anal and graphics

library(epicalc) # provides useful stat function in epid and examples

**2.1. Reading data in a file**

For reading and writing in files, R uses the working directory. To find this directory, the command **getwd()** (get working directory) can be used, and the working directory can be changed with **setwd("C:/directory")**. It is necessary to give the path to a file if it is not in the working directory.

**Loading a SAS dataset into R**

The function **read.xport()** from the package ‘**foreign**’ or the function **sasxport.get()** from the package ‘**Hmisc**’ reads a file in SAS transport (XPORT) format and return a list of data frames. In SAS create the transport file as follow:

libname one 'C:\your path';

libname rd xport 'C:\your path\R\sasdata.xport';

**proc** **copy** in=one out=rd;

select sasdata;

**run**;

or

**data** rd.sasdata;

set one.sasdata;

**run**;

Then in R:

sasdata<-read.xport(‘sasdata.xport’) # from package foreign

sasdata<-sasxport.get(‘sasdata.xport’) # from package Hmisc

The function **read.sas7bdat()** from the package ‘**sas7bdat**’ should also work and can read a SAS data format directly but this package is still experimental which means that the use of this package is at user’s own risk.

sasdata<-read.sas7bdat(‘sasdata.sas7bdat’) #from package sas7bdat

Otherwise in SAS using the “export data” option from the “file” menu export your data to a .txt or .csv file. Then in R you can use:

sasdata<-read.table(‘sasdata.txt’, header=T)

sasdata<-read.csv(‘sasdata.csv’)

# In RStudio, you can use the “**Import Dataset**” drop-down menu in the Environment window (upper right panel) to import text files (.txt or .csv).

You can also read other file types generated from other statistical software such as SPSS or STATA (from the package ‘**foreign**’)

read.csv read Excel (comma separate values (.csv)) files

read.dta read Stata (.dta) data files

read.spss read SPSS (.sav) data files

**2.2. Reading (Importing) the data for the class**

We have some data from the National Health and Nutrition Examination Survey III (NHANES III) study that we will be using during this course for some of the exercises. The primary objective is to examine *the association between blood lead levels and blood pressure/hypertension in general US adults*. There are several reference papers already published which are available in CTools (under the ‘Readings’ folder). The dataset is stored as **nhanes3.sas7bdat** at the CTools (under the ‘Data’ folder). First you need to convert it to an SAS xport file which can be read in R. Before doing this, let’s set your working directory first.

setwd("C:/epid674") # set working directory

getwd() # show the working directory defined

# You can also set your working directory in the “Files” menu (lower right window) and “More”.

nhanes3<-read.xport("nhanes3.xport")

names(nhanes3) #see variable names

# You can alternatively import a SAS dataset using sasxport.get() from the **Hmisc** package.

nhanes3a<-sasxport.get("nhanes3.xport")

names(nhanes3a)

# Note that all of the variable names are in upper case. To convert them to lower case, the function tolower(names()) can be used.

names(nhanes3)<-tolower(names(nhanes3))

names(nhanes3)

# Directly read a SAS data file (sas7bdat) using read.sas7bdat from the **sas7bdat** package.

library(sas7bdat)

nhanes3b<-read.sas7bdat("nhanes3.sas7bdat")

names(nhanes3b)

# If you data is in csv (or export as csv in SAS), you can use read.csv.

nhanes3c<-read.csv("nhanes3.csv")

names(nhanes3c)

**2.3. Description of the NHANES III data**

The **nhanes3** dataset includes 5,074 subjects randomly selected from the entire NHANES III data and is created only for this class.

**seqn** ID number

**age** age at interview (years)

**sex** sex (men 1; women 2)

**race** race (White 1; Black 2)

**urban** urbanization classification (urban 1; rural 2)

**region** census region (1='Northeast' 2='Midwest' 3='South' 4='West')

**pir** poverty income ratio

**psu** primary sampling unit

**strata** strata

**wt\_mh** sample weight

**bmi** body mass index (kg/m2)

**sbp** systolic blood pressure (mm Hg)

**dbp** diastolic blood pressure (mm Hg)

**hematoc** hematocrit (%)

**bpb** blood lead (μg/dL)

**chol** serum total cholesterol (mg/dL)

**trig** serum triglyceride (mg/dL)

**scalc** serum calcium (mg/dL)

**creat** serum creatinine (mg/dL)

**calc** dietary calcium consumption

**sodium** dietary sodium consumption

**potass** dietary potassium consumption

**educ** education (<12 yrs 1; 12-15 yrs 2; 16+ yrs 3)

**smk** smoking status (never 1; former 2; current 3)

**packyrs** cumulative cigarette (pack-years)

**alc** alcohol consumption (no 0; yes 1)

**phyact** physical activity (no 0; yes 1)

**med\_ht** antihypertensive medication (no 0; yes 1)

**diag\_ca** diagnosis of cancer (no 0; yes 1)

**diag\_dm** diagnosis of type-2 diabetes (no 0; yes 1)

**diag\_ht** diagnosis of hypertension (no 0; yes 1)

**htn** hypertension status (diag or med or sbp/dbp > 140/90) (no 0; yes 1)

**d\_total** total mortality (no 0; yes 1)

**d\_cancer** mortality due to cancer (no 0; yes 1)

**d\_cvd** mortality due to CVD (no 0; yes 1)

**pmon\_int** person-months of follow-up from interview date

**pmon\_mec** person-months of follow-up from MEC date

**2.4. Saving data**

Once you read an external dataset, it is better to create (save) a permanent R dataset (.rda) which can be used later without repeating the previous steps above. Using the **save()** and **load()** functions, you can save and load an R dataset.

save(nhanes3, file="nhanes3.rda")

load("nhanes3.rda")

The function **write.table()** or **write.csv()**writes in a file an object, typically a data frame but this could well be another kind of object (vector, matrix, . . . ).

write.table(nhanes3, file="nhanes3.txt")

write.csv(nhanes3, file="nhanes3.csv")

**2.5. Exploring the data.frame**

**Table 2.1** provides useful R functions to explore data frames.

|  |  |  |
| --- | --- | --- |
| Table 2.1. Useful functions to assess R data objects | | |
| **Function** | **Description** | **Try these examples** |
| *Returns summary objects* | | |
| str | Displays summary of data object structure | str(nhanes3) |
| attributes | Returns list with data object attributes | attributes(nhanes3)  attributes(summary(mod1)) |
| *Returns specific information* | | |
| mode | Returns mode of object | mode(nhanes3) |
| class | Returns class of object, if it exists | class(nhanes3) |
| length | Returns length of object | length(nhanes3$age) |
| dim | Returns vector with object dimension, if applicable | dim(nhanes3) |
| nrow | Returns number of rows, if applicable | nrow(nhanes3) |
| ncol | Returns number of columns, if applicable | ncol(nhanes3) |
| dimnames | Returns list containing vectors of names for each dimension, if applicable | dimnames(nhanes3) |
| rownames | Returns vector of row names of a matrix-like object | rownames(nhanes3) |
| colnames | Returns vector of column names of a matrix-like object | colnames(nhanes3) |
| names | Returns vector of row names for the list, if applicable (for a data.frame it returns the field names) | names(nhanes3) |
| row.names | Returns vector of row names for a data.frame | row.names(nhanes3) |
| head | Display first 6 lines of a data.frame | head(nhanes3)  nhanes3[1:6,] #equivalent |

First, let’s see all the data objects we have created. If you put no arguments between the **ls** function’s parentheses, you will get a list of all your objects. We can also remove the ones we do not want using the **rm()** function.

ls()

rm(x,y,z)

ls()

Examine the R data.frame. The **nhanes3** data.frame is a recursive object whose dimensions are 5,074 observations by 32 variables.

class(nhanes3)

dim(nhanes3) ## dimension of nhanes3 data

names(nhanes3) ## list variable names

nhanes3[1:6,] ## Listing the first 6 observations

head(nhanes3) ## equivalent to nhanes3[1:6,]

Examine one named column in the R data.frame. R does not recognize variable names without specifying the data.frame from which the variables come. Use ‘**data.frame$variable**’ to specify a variable.

length(nhanes3$age)

Attaching the data.frame can eliminate the tedious requirement of prefixing the name of the variable with the data.frame. Let’s attach the **nhanes3** dataset.

attach(nhanes3)

**2.6. Converting numeric (categorical) to categorical (numeric)**

All variables in **‘nhanes3’** are numeric. Let's create a character-type variable. The function **factor()** or **as.factor()** converts a numeric variable into a factor, which is a categorical data type in R. And try to label using the **levels()** function.

class(sex)

sex1 <- as.factor(sex) ## creating sex1 from sex

class(sex1)

is.factor(sex)

is.factor(sex1)

levels(sex1) <- c("male", "female")

sex[1:10]

sex1[1:10]

# The same result can be returned using the argument **labels()** within **factor()**.

sex1 <- factor(sex, levels=c(1,2), labels=c("male", "female"))

sex1 <- factor(sex, labels=c(“male”, “female”))

sex1[1:10]

table(sex1)

Likewise, a factor variable can be converted into a numeric using **as.numeric().**

sex2 <- as.numeric(sex1)

**2.7. Data description functions**

Once a data.frame has been attached, the variables in that data.frame may be directly accessed without using the **data.frame$variable** method. Calculate summary statistics for age and describe the distribution of age in this cohort.

mean(x=age, trim=0, na.rm=FALSE)

mean(na.rm=FALSE, x=age, trim=0)

mean(age, 0, F)

mean(age)

median(age)

quantile(age) ## this gives you 0,25,50,75,100%

quantile(age, c(0.1,0.9)) ## returns 10th and 90th percentiles

sd(age)

IQR(age)

The **summary()** function provides a more efficient method for summary statistics.

summary(age)

summary(nhanes3)

)

**2.8. Exercises**

You can find the dataset “**bpa.sas7bdat**” in CTools. Here's the list of variables.

1 cycle Num 8 Data Release Number

2 gender Num 8 Gender – Adjudicated (1:males, 2”females).

3 age Num 8 Age at Screening Adjudicated - Recode

4 raceeth Num 8 Race/Ethnicity – Recode

(1:Mex Am, 2:Other Hisp, 3:Non-Hisp white, 4: Non-Hisp black, 5:Other)

5 WTMEC2YR Num 8 Full Sample 2 Year Mec Exam Weight

6 psu Num 8 Masked Variance Pseudo-PSU

7 strata Num 8 Masked Variance Pseudo-Stratum

8 bmi Num 8 Body Mass Index (kg/m\*\*2)

9 waist Num 8 Waist Circumference (cm)

10 chol Num 8 Total cholesterol (mg/dL)

11 hdl Num 8 HDL-cholesterol (mg/dL)

12 scot Num 8 Cotinine (ng/mL)

13 ubpa Num 8 Urinary BPA, LOD corrected

14 SMK Num 8 smoking status (0:never, 1:former, 2:current)

15 PACKYR Num 8 pack-years of cigarettes

16 educat Num 8 Education (1:<high sch, 2:high sch, 3:some college+)

17 dmmed Num 8 Diabetes medication (0:No, 1:Yes)

18 inccat Num 8 Income: 5=unknown (including missing)

(1:<$20K, 2:$20K-$34.999K, 3:$35K-$64.999K, 4:$65K+)

19 a1c Num 8 HbA1c, standardized

20 wtmec6yr Num 8 6-yr (03-08) MEC weight

21 ucr Num 8 Creatinine, urine (mg/dL)

Using this dataset, do the following problems.

Problem 1. Create a SAS transport file and then read it in R using the function **read.xport()**or **sasxport.get()**. Also, convert bpa.sas7bdat to a **csv** file and then import it in R. Create an R data file for future use.

Problem 2. Compute the mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum of **age** and urinary bisphenol-A (**ubpa**).

**2.9. Extracting subsets**:

If you want to see summary statistics of the subset of variables, you can specify the column indexes or the variable names. In R, the **c()** function is used to combine the several into one. If you use a negative sign on an index, you will exclude those columns. Useful ways of indexing vectors in R are provided in **Table 2.2**.

summary(nhanes3[,1:5])

summary(nhanes3[,16:20])

summary(nhanes3[,c(1,6,11,16,21)])

summary(nhanes3[,c("age","sex","race","bpb","sbp")])

summary(nhanes3[,-c(1:30)])

# The **which()** function gives the TRUE indices of a logical object where the condition is true.

which(age>80)

which(bmi<18)

# Using the **which()** function, you can also run any R commands for subsets.

white<-which(race==1)

black<-which(race==2)

summ(nhanes3[white,])

summ(nhanes3[black,])

summ(nhanes3[-white,])

# Using the AND operator (&), we can make compound selections (multiple criteria). Let’s compute summary statistics for subjects who are both male and black.

malewhite<-which(nhanes3$sex==1&nhanes3$race==2)

summ(nhanes3[malewhite,])

|  |  |
| --- | --- |
| Table 2.2. Common ways of indexing vectors | |
| Indexing | Try these examples |
| By position | x<-c(chol=234, sbp=148, dbp=78, age=54) |
|  | x[c(2,3)] #positions to include |
|  | x[-c(1,3,4)] #positions to exclude |
|  | x[which(x<100)] |
| By name (if exists) | x[c(“sbp”,”dbp”)] |
| By logical | x<100 |
|  | (x<150)&(x>70) |
| Unique values | samp<-sample(1:5, 25, replace=T); samp  unique(samp) |

You can create a subset from a data frame using the **subset()** function. For example, create a dataset for **race**==1 (White).

nhanes3.w<-subset(nhanes3, race==1)

summ(nhanes3.w)

# To select only the '**age**', '**bmi**', and '**sex**' variables among white people,

nhanes3.w<-subset(nhanes3, race==1, select=c(age,bmi,sex))

summary(nhanes3.w)

# **sample()** takes a random sample of the specified size from the elements of x using either with or without replacement.

samp<-sample(seqn, 100)

samp

samp1<-sample(seqn, 100, replace=T)

samp1

**2.10. Create categorical variables**

The **cut()** function will divide age into five levels to create age quintiles, **AGE5**. The basic **cut()** function chooses five equally spaced intervals of age. The second version of the **cut()** function chooses cut points that are quintiles of age. The categories have equal numbers of subjects, but are not equally spaced on the age axis. The second version will discard the lowest values if you omit the **include.lowest** argument.

AGE5a<-cut(age, 5)

summary(AGE5a)

AGE5b<-cut(age, quantile(age, c(0,.2,.4,.6,.8,1)), include.lowest=T)

summary(AGE5b)

You can also create an age group variable with specific cut values. Note that the minimum and maximum of the arguments in **cut()** are the outer most boundaries.

AGE5c<-cut(age, breaks=c(19,40,50,60,70,90))

summary(AGE5c)

**AGE5** variables are factors or categorical variables. They can be transformed into simple numeric variables using the "**unclass()**" function. This is useful when this categorical variable is treated as ordinal (in regression analysis).

age5c<-unclass(AGE5c)

summary(age5c)

class(age5c)

This can also be done using the function **cut2()** from **Hmisc**.

AGE5d<-cut2(age, g=5)

summary(AGE5d)

AGE5e<-cut2(age, c(40,50,60,70))

summary(AGE5e)

Also try the following. First, create a copy of the numeric variable **age** and name it **agecat**. Then, replace elements of **agecat** with character strings for each age category, creating a character vector.

agecat <- age

agecat[age<30] <- "20’s"

agecat[age>=30 & age<40] <- "30’s"

agecat[age>=40 & age<50] <- "40’s"

agecat[age>=50 & age<60] <- "50’s"

agecat[age>=60 & age<70] <- "60’s"

agecat[age>=70 & age<80] <- "70’s"

agecat[age>=80] <- "80’s"

table(agecat)

Frequency and cross-tabulation: Frequency of categorical variables can be obtained using the **table()** and **tab1()** functions. Cross-tabulation can be performed by using the **tabpct()** function.

table(race)

table(sex, race)

**2.11. Save workspace image**

When you finish your work, you want to save your script/output files as well as any graphs that you want to keep. When exiting R, you will be asked if you want to save the workspace image: answer yes to save all your work (saved as "**.Rdata**").

You should see two new files in your working directory: **".Rdata"**, which is the working environment saved from the latest **R** session, and **".Rhistory"**, which recorded all the commands in the preceding **R** session.

When starting R again, you can load this saved data.

load(“.RData”)

Note that before quitting R you can also save your workspace image with the name of your file by typing

save.image(file=”epid674.RData”)

Then when you quit R you should answer “No” to the question.

When you start R, load this saved data.

load(“epid674.RData”)

Also note that the image saved from the previous session contains only objects in the '.GlobalEnv', which is the first position in the search path. The whole search path is not saved. For example, any libraries manually loaded in the previous session need to be reloaded.

Usually one would like to start R afresh every time so it is advised to always choose “No” when prompted to save the workspace.

**2.12. Exercises**

Using the dataset “**bpa.sas7bdat**”, do the following problems.

Problem 1. Compute the mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum of age and BMI by gender.

Problem 2. Complete the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BMI category | | | |
|  | Underweight  (<18.5) | Normal  (≥18.5 & <25) | Overweight  (≥25 & <30) | Obese  (≥30) |
| Total cholesterol,  mean (SD) |  |  |  |  |
| Smoking status, N (%) |  |  |  |  |
| Never |  |  |  |  |
| Former |  |  |  |  |
| Current |  |  |  |  |