**Introduction to R in Statistics and Genetics**

Neurepiomics 2023

Qiong Yang, Associate Professor of Biostatistics, Boston University

Enchanted Springs Ranch, Boerne, TX

Contents

[Scope of the Workshop 4](#_Toc147754533)

[Introduction of R 5](#_Toc147754534)

[What is R? 5](#_Toc147754535)

[Why learn R 5](#_Toc147754536)

[How to master R 5](#_Toc147754537)

[Obtain and Run R 6](#_Toc147754538)

[Basic Operations in R 7](#_Toc147754539)

[Interactive Run R in console 7](#_Toc147754540)

[Create a script to compose and save R codes 8](#_Toc147754541)

[Run Scripts 9](#_Toc147754542)

[Data import 10](#_Toc147754543)

[Set working directory 11](#_Toc147754544)

[Read Excel Document 12](#_Toc147754545)

[Read plain text file 13](#_Toc147754546)

[Function and Packages in R 14](#_Toc147754547)

[View Function Manuals 14](#_Toc147754548)

[Import datasets created by other statistical software 15](#_Toc147754549)

[View Created Objects 16](#_Toc147754550)

[Subset Data 17](#_Toc147754551)

[Example: Select subset by rows 17](#_Toc147754552)

[Export Data 21](#_Toc147754553)

[Sort and Check duplicates 22](#_Toc147754554)

[Sort data 22](#_Toc147754555)

[Check Duplicates 22](#_Toc147754556)

[Compute Frequencies of Unique Values in a Variable 23](#_Toc147754557)

[Merge Datasets 24](#_Toc147754558)

[Merge by a common column 24](#_Toc147754559)

[Merge by a common variable named differently in the two datasets 24](#_Toc147754560)

[Merge by more than one common variables 24](#_Toc147754561)

[Multiple matches on the merged-by-variables values 26](#_Toc147754562)

[Keep records of Unmatched Common Variable 28](#_Toc147754563)

[Potential Pitfalls 30](#_Toc147754564)

[Functions in R 33](#_Toc147754565)

[Missing Values 35](#_Toc147754566)

[Explorative Data Analysis with R 41](#_Toc147754567)

[Visualize Variable Relationships and Distributions in a Single Plot 42](#_Toc147754568)

[Individual Plotting Functions for Explorative Data Analyses 44](#_Toc147754569)

[Summary Statistics 45](#_Toc147754570)

[Example: Compute mean and quantiles statistic of a single variable 45](#_Toc147754571)

[Example: Create a characteristic table for publication 46](#_Toc147754572)

[Example: Characteristic table with p-value as last column 48](#_Toc147754573)

[Genetics and Omics Research Using R 50](#_Toc147754574)

[Analyzing a few genetic variants or omic measures 50](#_Toc147754575)

[Family data 51](#_Toc147754576)

[Analyze a large number of genetic variants 52](#_Toc147754577)

[Two sample mendelian randomization analyses 54](#_Toc147754578)

[Using GWAS summary statistics from MRC IEU open GWAS database to perform two-sample MR 54](#_Toc147754579)

Scope of the Workshop

In this workshop, we will talk about how to use R in

* Data Processing
* Generate Graphics
* Explorative Data Analyses
* Genetic and Omics Analyses

Introduction of R

What is R?

R is a statistical programming language available for **free** and offers comprehensive tools to perform data management, statistical modeling and inferences, and graphic presentations.

Why learn R

You can perform both classic and cutting-edge statistical computing in R. New machine learning methods are almost all available in R. Many newly developed statistical methods in omics research are only available in R, such as the two-sample Mendelian Randomization and many of its extensions and improvements.

How to master R

Use it as much as you can. Learn from other people by google searching the answers to your questions and problems online.

Obtain and Run R

It is best to install **RStudio** and run R from there. **RStudio** is an integrated development environment for R, with many convenient features.

1. **Download and Install R**

* Go to <http://www.r-project.org/>;
* Click on **‘download/CRAN’** on the left panel;
* Select a mirror site closest to your location.
* Click on an operating system (**Windows, Mac** or **Linux**) in the “download and install R” section.
* Click on **‘base’**;
* Click on the [Download R-4.3.1 for Windows](https://mirror.las.iastate.edu/CRAN/bin/windows/base/R-4.3.1-win.exe) (79 megabytes, 64 bit) to download the latest version of R setup program for Windows.
* Click on the setup program and follow the instructions to install R.

1. **Download and Install R studio**

[https://www.rstudio.com/products/rstudio/download/#download](https://www.rstudio.com/products/rstudio/download/%23download)

Basic Operations in R

Interactive Run R in console

Issue a command at the command prompt ‘>’, and press enter

3+1 #expression, print out

## [1] 4

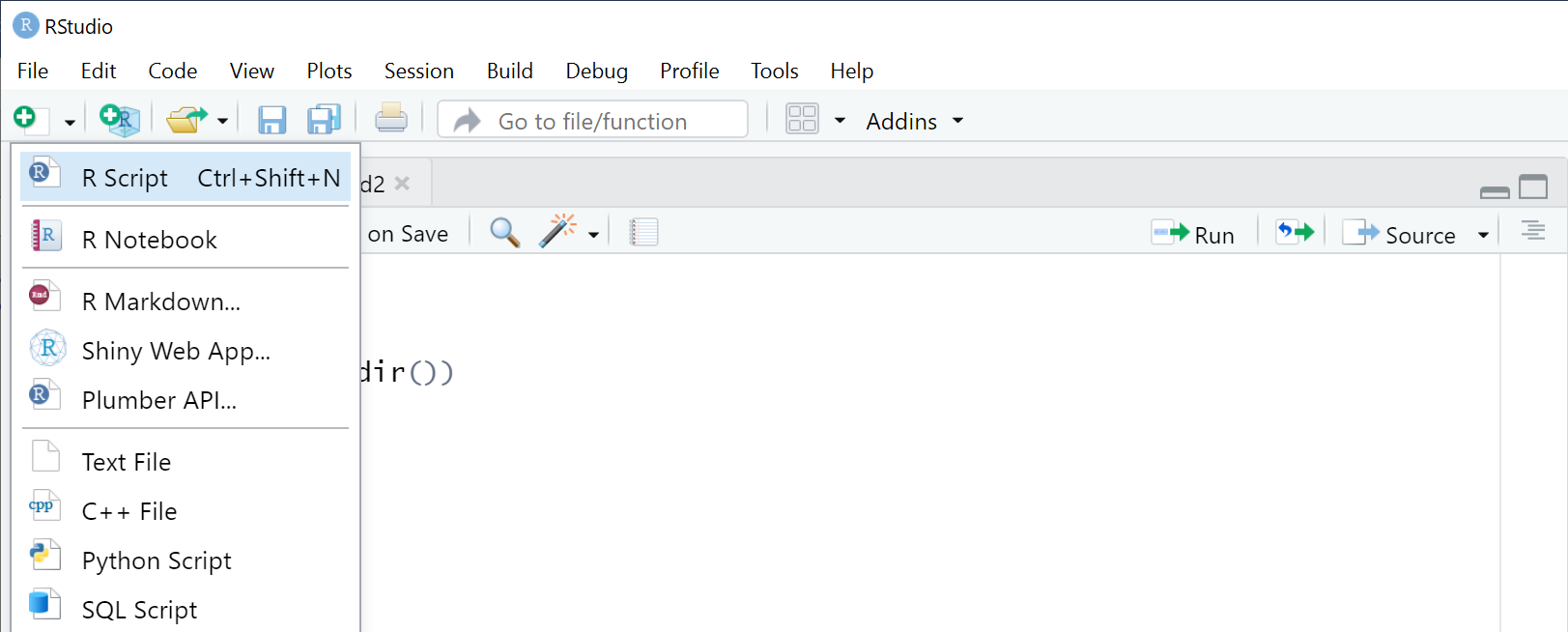
x <- 5 #Assignment, no printout  
x #print an object

## [1] 5

y <- c(1,4,7) #create a vector object

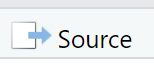
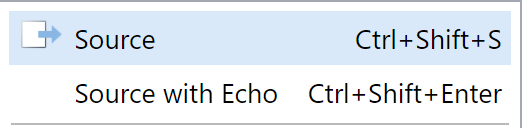
* + Assign is <- or =
  + Text after **#** is treated as comments and ignored in the execution
  + Press **Esc** key to cancel current statement

Create a script to compose and save R codes



* In the upper right source panel, click on the first  button then to open an editor.
* Click on  to save the script

Run Scripts

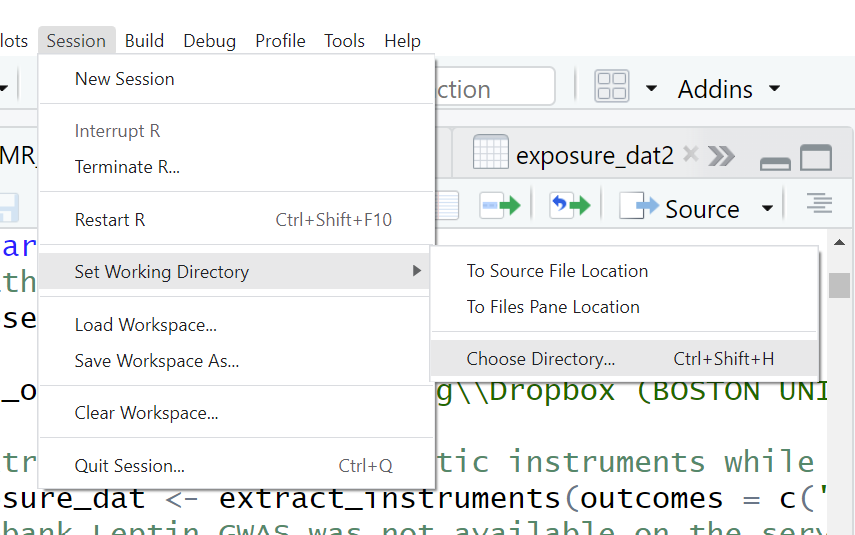
* Place the cursor at a line, or highlight multiple lines; then click on the  to submit the line(lines) to the console.
* Click on  to run whole script without showing any printout from command.
* You can choose show the on-screen printout from commands by selecting “Source with Echo”
* 

Data import

Before we learn data import, please download all files from github for this workshop

<https://github.com/qyang10/R-2023>

Set working directory

To read the files we have downloaded, let’s first set the working directory to the directory with these files.

A command like the following was executed in console. You can copy & paste it to your script for future reference.

setwd("C:/Users/qyang/Dropbox (BOSTON UNIVERSITY)/R Tutorial")

Read Excel Document

Excel documents with multiple tabs can be imported in to R one tab at a time. User contributed package **xlsx** should be installed first.

install.packages("xlsx") #download from internet  
library(xlsx) #load into R  
**pqtl <-** read.xlsx**("Folkersen2020SuppTables.xlsx",**

**sheetName ="5 eQTL",**

**startRow = 2,**

**endRow = 548)**

we are reading the supplementary table from *Folkersen L, et al. Genomic and drug target evaluation of 90 cardiovascular proteins in 30,931 individuals. Nat Metab. 2020 Oct;2(10):1135-1148.*

If you receive an error about not having enough memory when you run the above command, you may close and reopen R-studio to release some memory. If the problem persists, you may save a relevant tab to new excel files with a single tab, and read it separately.

Read plain text file

By default read.csv() reads comma delimited plain text file, but you can change sep= <char> to read, e.g. tab demilited file, by specifying sep="\t". You may also specify whether there is a header (header=T/F), and fist n lineses to skip (skip=n) in the function call.

read.csv**("Folkersen2020ST5.csv", skip=1)**

Function and Packages in R

R use functions and operators to perform various tasks. Functions can be in base packages or user contributed packages. The former is downloaded when you download R. The latter require downloading for once. An internet connection is required to download packages.

View Function Manuals

You may view the manual of a function or a operator in **Help** tab of the lower right quadrant by typing the function name in the search window.

Note for functions from user-contributed packages, you need load the package using library(<lib>) before you can access the manual of the functions.

*Tip: Run the examples at end of the function manual to learn more.*

|  |  |
| --- | --- |
| Import datasets created by other statistical software  Datasets from other statistical software can be imported into R using functions from user-contributed packages.  **Example: Import a SAS dataset** |  |

install.packages("haven")  
library(haven)  
fram <-read\_sas("mydat.sas7bdat")

The R **haven** package also provides functions

**read\_spss()** for reading a SPSS data file (.sav)

**read\_dta()** for reading a STATA data file (.dta)

View Created Objects

Note that in the “**Environment**” tab in upper right quadrat, the assigned objects appear. Click on the arrow in front to view structure of the object, or click on the object name to view the data.

We may also use the following command to check the data

head(pqtl) #show first 6 records

tail(pqtl) #show first 6 records

dim(pqtl) #show number of rows and columns

## [1] 585 24

names(pqtl) #show all column names, if any

summary(pqtl) #summarize all variables

summary(pqtl$P.value) #summarize a single variable

Subset Data

The most common data manipulation is selecting columns and rows. They can be performed using filter() and select() from user contributed package dplyr .

Example: Select subset by rows

#Install a group of packages including dplyr

library(tidyverse)

#Select the rows where Target is CD40

d <- filter(pqtl, Protein=="CD40")

#Select the rows where Target contains "CD40"

d <- filter(pqtl,grepl("CD40",Protein))

Note: "==" is to test, "=" is assign & equivalent to "<-"

#Select the rows where Target is one of several proteins

d <- filter(pqtl, Protein%in%c("CD40","VEGF-A"))

#Try to Select all nominal significant cis pqtl

filter(pqtl, Cis.Trans..1.MB.=="cis", P.value<=0.05)

#Above command didn’t return data as expected, check

#class attributes of P.value

class(pqtl$P.value)

## [1] "character"

#Convert P.value to numeric

pqtl$P.value <-as.numeric(pqtl$P.value)

#Rerun the command, only print first 5 rows/5 columns

filter(pqtl, Cis.Trans..1.MB.=="cis", P.value<=0.05)[1:5,1:5]

Example: Select subset by columns

#Select columns 1-8, then view first a few records

select(pqtl,1:8)%>%  
 head()

#select rows then select columns by names  
filter(pqtl,grepl("CD40",Protein)) %>%  
 select(Protein,MarkerName,P.value)

## Protein MarkerName P.value  
## 1 CD40 19:57835252:C\_T 1.8e-12  
## 2 CD40 20:44747947:G\_T 0.0e+00  
## 3 CD40 20:44755376:C\_G 2.6e-38  
## 4 CD40-L 6:114340014:A\_G 1.6e-08  
## 5 CD40-L 8:106590706:A\_G 1.4e-08

Note: %>% redirect output from filter() as input to the first argument of select()

Export Data

Example: Export to a plain text file

#create a new dataset with pqtl for CD40  
new <- filter(pqtl,grepl("CD40",Protein)) %>%  
 select(1:8)  
  
write.table(new, "CD40.pqtl.csv", sep=",", row.names=F)  
  
#Example: Export to an excel file  
write.xlsx(new, "stroke.pqtls.xlsx", sheetName="CD40", row.names=F)  
  
#Add another tab to the xlsx file with append=T  
filter(pqtl,grepl("VEGF",Protein)) %>%  
 select(1:8) %>%  
 write.xlsx ("stroke.pqtls.xlsx", sheetName="VEGF", row.names=F, append=T)

Sort and Check duplicates

Sort data

#Example: sort pqtl data by P.value

**pqtl <- pqtl[**order**(pqtl**$**P.value),]**

Check Duplicates

duplicated(c(1,1,2,3,1))

## [1] FALSE TRUE FALSE FALSE TRUE

#compute number of unique and duplicated proteins

table(duplicated(pqtl$Protein))

##   
## FALSE TRUE   
## 85 460

Compute Frequencies of Unique Values in a Variable

#compute number of pqtls for each gene

table(pqtl$Protein)

##   
## ADM AGRP CA-125 CASP-8 CCL20   
## 6 6 8 5 2   
## CCL3 CCL4 CD40 CD40-L CHI3L1

Merge Datasets

Merge by a common column

Suppose we have two datasets a, and b, we can merge by common column "ID".

merge(a, b, by="id",)

Merge by a common variable named differently in the two datasets

merge(a, b, by.x="id", by.y="ID")

Merge by more than one common variables

merge(a, b, by=c("famid", "id"))

Multiple matches on the merged-by-variables values

If there are more than one matches of values of the merged-by variables, by default, all possible combinations of matches are included in the merged dataset. **But all records corresponding to unmatched values of the merged by variables were removed.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Dataset 1:D1 | |  | Dataset 2:D2 | |  |  |  |  |  |
|  | V1 | V2 |  | V1 | X2 |  |  | V1 | V2 | X2 |
|  | 1 | 3 |  | 6 | a | **merge(D1,D2,by=V1)** | | 1 | 3 | f |
|  | 1 | 6 |  | 1 | f |  |  | 1 | 3 | d |
|  | 3 | 8 |  | 7 | e |  |  | 1 | 6 | f |
|  | 4 | 9 |  | 1 | d |  |  | 1 | 6 | d |

In the results, all 4 possible combinations of V1=1 between the two datasets were in merged dataset. However, records corresponding to V1=3 or 4 of first dataset, and V1=6,7 of the second datasets were not included in the final dataset because they only appear in a single dataset.

Keep records of Unmatched Common Variable

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **merge(D1,D2,by=V1,all.x=T)** | | |  | **merge(D1,D2,by=V1,all=T)** | | |
| V1 | V2 | X2 |  | V1 | V2 | X2 |
| 1 | 3 | f |  | 1 | 3 | f |
| 1 | 3 | d |  | 1 | 3 | d |
| 1 | 6 | f |  | 1 | 6 | f |
| 1 | 6 | d |  | 1 | 6 | d |
| 3 | 8 | NA |  | 3 | 8 | NA |
| 4 | 9 | NA |  | 4 | 9 | NA |
|  |  |  |  | 6 | NA | a |
|  |  |  |  | 7 | NA | e |

Potential Pitfalls

* All commands and variables are case sensitive in R
* Missing values are designated by “NA” (instead of “.” as in SAS)
* R allows dots (.) and underscores (\_) in the name of functions or variables.
* Referencing file names uses double backslashes "c:\\temp\\test.txt"(add a slash to the path copied from windows file explorer)

or using single **forward slashes as "c:/temp/test.txt"**

**Data Summary and Explorative Analyses**

**Simulate a dataset according to mean, sd and correlation of FHS brain MRI data.**

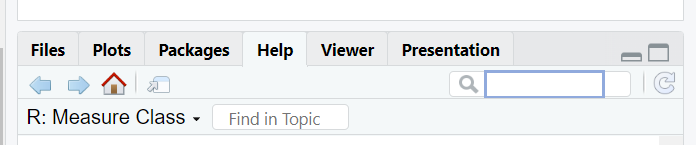
Functions in R

Tasks are carried out in R using **operators** or **functions**

* To call an existing function, type the function name followed by brackets.

For example, **ls(), rm(x,y,z)**

* To view manuals of a function, enter the function name in the search window in the Help tab in the lower right quadrant:



* To view arguments of a function, type function name followed by round bracket, then R-studio will show all arguments of this function.
* When calling a function, default values of unspecified arguments are used.

For example, **pnorm(1.65)** is equivalent to

**pnorm(q = 1.65,** **mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)**

* Argument names may be omitted when specifying their values: R will assign the values to the arguments, following the order of the arguments in the function definition, e.g. the following 2 are equivalent

**pnorm(3,** **0.5,3)**

**pnorm(q = 3,** **mean = 0.5, sd = 3)**

Out of order arguments can be added with argument names:

write.table(data, "filename", sep=",")

* To view source code of a function, type the **function name** and press **enter in console**. For example, type **lm** and press **enter** will display the source code.

Missing Values

**NA is a reserved value in R**

**is.na(X)** #test for missing values in X

**!is.na(X)** #test for NOT a missing value

**na.omit(X)** #delete missing values from a vector or column

#and rows with at least one missing value

For example:

**X <- 5**

**is.na(X) #right way to test missingness**

**[1] FALSE**

**X == NA #wrong way to test missingness**

**[1] NA**

Any logical or mathematical operation with NA will return NA, except %in%

|  |  |  |
| --- | --- | --- |
| **X > NA**  **[1] NA** | **X + NA**  **[1] NA** | **NA %in% X**  **[1] FALSE** |

In element selection, use **which()** on the logical vector to avoid NA being unintentionally selected

Example:

**x <-c(1,3,NA,4,6,NA)**

**x >1**

**[1] FALSE TRUE NA TRUE TRUE NA**

**x[x>1]**

**[1] 3 NA 4 6 NA**

Not only positions with TRUE selected, but also position with NA

However **which(x>1)** filters out missing values when returning indexes

**which(x>1)**

**[1]2 4 5**

**x[which(x>1)]**

**[1] 3 4 6**

**Many functions have an option to choose how to deal with NA.**

**sum (..., na.rm = FALSE)**; the default is **not** removing NA when performing a sum. If there is a NA in the elements, the result will be NA. To remove NAs from elements, specify na.rm = T. A partial list of such functions includes **mean(), rowSum(), colSum()**, **var()**.

For example,

**sum(c(1,2,NA,3,4))**

**[1] NA**

**sum(c(1,2,NA,3,4),na.rm=T)**

**[1] 10**

**lm(**…, **na.action=na.omit**); remove any records with missing values in the variables used in regression

Explorative Data Analysis with R

The purpose of the exploratory data analysis usually involves checking model assumptions, checking outliers and examining relationships among variables.

Visualize Variable Relationships and Distributions in a Single Plot

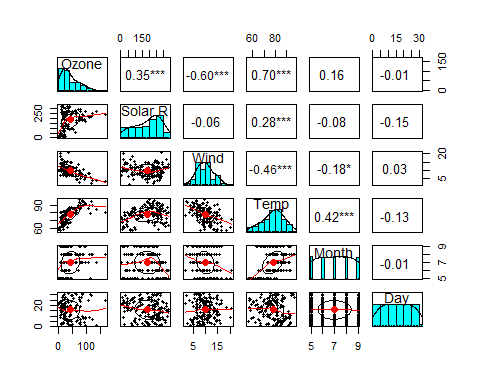
Correlation coefficient is notorious in missing non-linear relationships, therefore it is important to use scatter plots to visualize variable relationship.

The pairs.panels() in user contributed package **psych** can visualize distribution, pairwise relationship, and correlation coefficient in a single graph:

install.packages("psych")

library(psych)

pairs.panels(airquality, stars=T)



Confidence ellipses on the bivariate mean (red dot) were drawn in scatter plots as a visual test for bivariate normality and an indication of the strength of the correlation: The narrower the ellipse, the greater the correlation. The wider and more round it is, the less the correlation.

Individual Plotting Functions for Explorative Data Analyses

hist(x), histogram of a continuous variable

barplot(table(x)), barplot to visualize frequency of a factor variable x

qqnorm(x) and qqline(x), normal QQ plot for a continuous variable

plot(x,y), Scatter plot of two variables

scatter.smooth(x,y) Scatter plot of two variables with smooth curve added

boxplot(y~x), Box plot of a continuous Y vs a categorical x, x must be a character or factor variable.

Summary Statistics

In contrast to graphics, summary statistics provide quantitative evaluation of the data distribution.

Example: Compute skewness statistic of a single variable

library(moments)   
skewness(airquality$Ozone)

## [1] NA

Why I got NA? How to make it right?

Example: Compute mean and quantiles statistic of a single variable

summary(airquality$Ozone)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's## 1.00 18.00 31.50 42.13 63.25 168.00 37

Example: Create a characteristic table for publication

**install.packages("**table1"**)**

**library(**table1**)**

**library**(boot) #need this package because melanoma dataset is in it

str(melanoma)

*# create a copy of the dataset to make changes*

melanoma2 <- melanoma

*# Make the variable to stratify a factor, arrange and label the levels*

*#labels shown instead of values in operations after the formatting below*

melanoma2$status <- **factor**(melanoma2$status,

levels=**c**(2,1,3),

labels=**c**("Alive",

"Melanoma death",

"Non-melanoma death"))

*# Generate descriptive stats for age,sex,ulcer and thickness by status*

**table1**(~ **factor**(sex) + age + **factor**(ulcer) + thickness | status, data=melanoma2)

|  | **Alive (N=134)** | **Melanoma death (N=57)** | **Non-melanoma death (N=14)** | **Overall (N=205)** |
| --- | --- | --- | --- | --- |
| **factor(sex)** |  |  |  |  |
| 0 | 91 (67.9%) | 28 (49.1%) | 7 (50.0%) | 126 (61.5%) |
| 1 | 43 (32.1%) | 29 (50.9%) | 7 (50.0%) | 79 (38.5%) |
| **age** |  |  |  |  |
| Mean (SD) | 50.0 (15.9) | 55.1 (17.9) | 65.3 (10.9) | 52.5 (16.7) |
| Median [Min, Max] | 52.0 [4.00, 84.0] | 56.0 [14.0, 95.0] | 65.0 [49.0, 86.0] | 54.0 [4.00, 95.0] |
| **factor(ulcer)** |  |  |  |  |
| 0 | 92 (68.7%) | 16 (28.1%) | 7 (50.0%) | 115 (56.1%) |
| 1 | 42 (31.3%) | 41 (71.9%) | 7 (50.0%) | 90 (43.9%) |
| **thickness** |  |  |  |  |
| Mean (SD) | 2.24 (2.33) | 4.31 (3.57) | 3.72 (3.63) | 2.92 (2.96) |
| Median [Min, Max] | 1.36 [0.100, 12.9] | 3.54 [0.320, 17.4] | 2.26 [0.160, 12.6] | * 1. 0.100, 17.4] |

Example: Characteristic table with p-value as last column

# read in the user defined pvalue() in Pvalue4Table1.txt

source("Pvalue4Table1.txt")

# remove non-melanoma death from status variable

melanoma2$status2 <- factor(melanoma2$status, exclude="3",  
 levels=c(2,1),  
 labels=c("Alive", "Melanoma death" ))  
# change sex and ulcer to factor variables

melanoma2$sex <- factor(melanoma2$sex,   
 levels=c(0,1),  
 labels=c("Male", "Female" ))  
melanoma2$ulcer <- factor(melanoma2$ulcer,   
 levels=c(0,1),  
 labels=c("No", "Yes" ))  
# Generate descriptive stats by binary status   
table1(~ sex + age + ulcer + thickness | status2, data=melanoma2,  
overall=F, extra.col=list(`P-value`=pvalue))

|  | Alive (N=134) | Melanoma death (N=57) | P-value |
| --- | --- | --- | --- |
| **sex** |  |  |  |
| Male | 91 (67.9%) | 28 (49.1%) | 0.0221 |
| Female | 43 (32.1%) | 29 (50.9%) |  |
| **age** |  |  |  |
| Mean (SD) | 50.0 (15.9) | 55.1 (17.9) | 0.067 |
| Median [Min, Max] | 52.0 [4.00, 84.0] | 56.0 [14.0, 95.0] |  |
| **ulcer** |  |  |  |
| No | 92 (68.7%) | 16 (28.1%) | &lt;0.001 |
| Yes | 42 (31.3%) | 41 (71.9%) |  |
| **thickness** |  |  |  |
| Mean (SD) | 2.24 (2.33) | 4.31 (3.57) | &lt;0.001 |
| Median [Min, Max] | 1.36 [0.100, 12.9] | 3.54 [0.320, 17.4] |  |

Genetics and Omics Research Using R

Analyzing a few genetic variants or omic measures

We can use statistical modeling functions to perform the association analyses for a single variant or multiple variants or a score created using multiple variants.

lm(y~x1+x2+x1:x2), linear regression of y on x1, x2, and their interaction

glm(y~x1+x2, family=binomial), logistic regression

Family data

If you have family data, you need compute kinship matrix and fit linear mixed effects model for continuous outcomes

**library(**coxme**)**

**library(**kinship2**)**

# Compute kinship coefficient matrix

ped <- read.csv("ped.csv" )%>%

with(pedigree(id,fid,mid))

kmat <- kinship(ped)

# perform linear mixed effects model

lmekin(outcome ~gscore+age+sex+(1|id), data=final, varlist= kmat\*2)

Analyze a large number of genetic variants

Desktop R is not efficient to handle data that are too large such as GWAS data. But we can prepare data and commands in R, then call external programs such as

1. Plink to generate results for unrelated, or
2. R package gaston to generate results accounting for encrypted relatedness.

<https://cran.r-project.org/web/packages/gaston/index.html>

association.test.dosage ("extract\_chr22.vcf.gz" ",

Y= Y2$B\_VO2MX, X = COV2, method = "lmm”,

response = "quantitative",

test = "wald",

K =K, eigenK =eiK,

beg = 1,end = n\_col, p = 0,

tol = .Machine$double.eps^0.25)

Two sample mendelian randomization analyses

Two sample Mendelian randomization (2SMR) is a method to estimate the causal effect of an exposure on an outcome using only summary statistics from genome wide association studies (GWAS).

Using GWAS summary statistics from MRC IEU open GWAS database to perform two-sample MR

Developed at the MRC Integrative Epidemiology Unit (IEU) at the University of Bristol, this resource is a manually curated collection of complete GWAS summary datasets made available as open source files for download, or by querying a database of the complete data.

They developed an R package to facilitate such analyses

install.packages("remotes")

library(remotes)

install\_github("MRCIEU/TwoSampleMR")

Example: the causal effect of body mass index on coronary heart disease, looks like this:

[library](https://rdrr.io/r/base/library.html)(TwoSampleMR)

# Get instruments

exposure\_dat <- [extract\_instruments](https://mrcieu.github.io/TwoSampleMR/reference/extract_instruments.html)("ieu-a-2")

# Get effects of instruments on outcome

outcome\_dat <- [extract\_outcome\_data](https://mrcieu.github.io/TwoSampleMR/reference/extract_outcome_data.html)(snps=exposure\_dat$SNP, outcomes = "ieu-a-7")

# Harmonise the exposure and outcome data

dat <- [harmonise\_data](https://mrcieu.github.io/TwoSampleMR/reference/harmonise_data.html)(exposure\_dat, outcome\_dat)

# Perform MR

res <- [mr](https://mrcieu.github.io/TwoSampleMR/reference/mr.html)(dat)