Topics on R
Example R Codes

Konan Hara*

September 8, 2021

1 Getting Help

```
help(matrix)
example(matrix)

##
## matrix> is.matrix(as.matrix(1:10))
## [1] TRUE
##
## matrix> !is.matrix(warpbreaks)  # data.frame, NOT matrix!
## [1] TRUE
##
## matrix> warpbreaks[1:10,]
##    breaks wool tension
## 1      26    A       L
## 2      30    A       L
## 3      54    A       L
## 4      25    A       L
## 5      70    A       L
## 6      52    A       L
## 7      51    A       L
## 8      26    A       L
## 9      67    A       L
## 10     18    A       M
##
## matrix> as.matrix(warpbreaks[1:10,])  # using as.matrix.data.frame(.) method
##    breaks wool tension
## 1  "26"   "A"  "L"
## 2  "30"   "A"  "L"
## 3  "54"   "A"  "L"
## 4  "25"   "A"  "L"
## 5  "70"   "A"  "L"
## 6  "52"   "A"  "L"
## 7  "51"   "A"  "L"
## 8  "26"   "A"  "L"
## 9  "67"   "A"  "L"
```

*Department of Economics, University of Arizona. E-mail: harakonan@email.arizona.edu

```
## 10 "18"   "A"  "M"
##
## matrix> ## Example of setting row and column names
## matrix> mdat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3, byrow = TRUE,
## matrix+                dimnames = list(c("row1", "row2"),
## matrix+                                c("C.1", "C.2", "C.3")))
##
## matrix> mdat
##      C.1 C.2 C.3
## row1   1   2   3
## row2  11  12  13

help.search("linearmodels")
help(package="sandwich")
```

2 R Objects

2.1 Numerics, Characters, Logicals, and Factors

```
# numeric
n <- 100
class(n)

## [1] "numeric"

# numeric
c <- "Konan"
class(c)

## [1] "character"

# logical
class(TRUE)

## [1] "logical"

class(FALSE)

## [1] "logical"

class(T)

## [1] "logical"

class(F)

## [1] "logical"

as.numeric(TRUE)

## [1] 1
```

```r
as.numeric(FALSE)

## [1] 0

# factor
sample_num <- rbinom(10,1,0.5)
sample_num

##  [1] 0 1 1 0 0 0 0 0 1 1

sample_factor <- factor(sample_num
        , levels = c(0,1)
        , labels = c("control","treatment"))
sample_factor

##  [1] control   treatment treatment control   control   control   control
##  [8] control   treatment treatment
## Levels: control treatment

class(sample_factor)

## [1] "factor"
```

2.2 Vectors

```r
# define a vector
v <- c(seq(from = 1, to = 5, by = 1))
v

## [1] 1 2 3 4 5

class(v)

## [1] "numeric"

# recycle rule
c(seq(1,3)) + c(seq(1,4))

## Warning in c(seq(1, 3)) + c(seq(1, 4)): longer object length is not a multiple
of shorter object length

## [1] 2 4 6 5

c(seq(1,2)) + c(seq(1,4))

## [1] 2 4 4 6

# operations are vectorized
v <- 1:5
v

## [1] 1 2 3 4 5

sqrt(v)

## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

2.3 Matrices

```
# define a matrix
m <- matrix(1:4, nrow = 2, ncol = 2)
m

##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4

class(m)

## [1] "matrix" "array"

# transpose
t(m)

##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4

# dimension
dim(m)

## [1] 2 2

# length
length(m)

## [1] 4

# indexing
m[2,1]

## [1] 2

m[2,]

## [1] 2 4

m[,1]

## [1] 1 2
```

2.4 Lists

```
# define a list
l <- list(v,m,c("Tiemen","Konan"))
l

## [[1]]
```

```
## [1] 1 2 3 4 5
##
## [[2]]
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## [[3]]
## [1] "Tiemen" "Konan"
```

```r
class(l)
```

```
## [1] "list"
```

```r
# element-wise calculation
set.seed(100)
l <- list(rnorm(2), runif(4), rgamma(6,1,1))
l
```

```
## [[1]]
## [1] -0.5021924  0.1315312
##
## [[2]]
## [1] 0.4685493 0.4837707 0.8124026 0.3703205
##
## [[3]]
## [1] 0.5861317 0.7506868 0.1732320 0.8569455 0.2751689 0.5655790
```

```r
lapply(l, mean)
```

```
## [[1]]
## [1] -0.1853306
##
## [[2]]
## [1] 0.5337608
##
## [[3]]
## [1] 0.534624
```

2.5 Data Frames

```r
# define a data.frame
df <- data.frame(
          V1 = rep(c(1, 2), 5)[-10]
        , V2 = 1:9
        , V3 = c(0.5, 1.0, 1.5)
        , V4 = rep(LETTERS[1:3], 3)
        )
df
```

```
##   V1 V2  V3 V4
```

```
## 1  1  1 0.5  A
## 2  2  2 1.0  B
## 3  1  3 1.5  C
## 4  2  4 0.5  A
## 5  1  5 1.0  B
## 6  2  6 1.5  C
## 7  1  7 0.5  A
## 8  2  8 1.0  B
## 9  1  9 1.5  C
```

```
# define a data.table
# install.packages("data.table")
library(data.table)
dt <- data.table(
          V1 = rep(c(1, 2), 5)[-10]
        , V2 = 1:9
        , V3 = c(0.5, 1.0, 1.5)
        , V4 = rep(LETTERS[1:3], 3)
        )
dt
```

```
##    V1 V2  V3 V4
## 1:  1  1 0.5  A
## 2:  2  2 1.0  B
## 3:  1  3 1.5  C
## 4:  2  4 0.5  A
## 5:  1  5 1.0  B
## 6:  2  6 1.5  C
## 7:  1  7 0.5  A
## 8:  2  8 1.0  B
## 9:  1  9 1.5  C
```

```
# define a tibble
# install.packages("dplyr")
# dplyr is a part of the "tidyverse"
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:data.table':
##
##    between, first, last
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union
```

```
# V3 way doesn't work in tibble
tib <- tibble(
          V1 = rep(c(1, 2), 5)[-10]
```

```r
        , V2 = 1:9
        , V3 = c(0.5, 1.0, 1.5)
        , V4 = rep(LETTERS[1:3], 3)
        )
```

```
## Error: Tibble columns must have compatible sizes.
## * Size 9: Existing data.
## * Size 3: Column `V3`.
## i Only values of size one are recycled.
```

```r
tib <- tibble(
          V1 = rep(c(1, 2), 5)[-10]
        , V2 = 1:9
        , V3 = rep(c(0.5, 1.0, 1.5), 3)
        , V4 = rep(LETTERS[1:3], 3)
        )
tib
```

```
## # A tibble: 9 x 4
##      V1    V2    V3 V4
##   <dbl> <int> <dbl> <chr>
## 1     1     1   0.5 A
## 2     2     2   1   B
## 3     1     3   1.5 C
## 4     2     4   0.5 A
## 5     1     5   1   B
## 6     2     6   1.5 C
## 7     1     7   0.5 A
## 8     2     8   1   B
## 9     1     9   1.5 C
```

```r
# demo using the R built-in iris data set
# data.frame
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
# data.table
iris_dt <- as.data.table(iris)
iris_dt
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
##   1:          5.1         3.5          1.4         0.2    setosa
##   2:          4.9         3.0          1.4         0.2    setosa
##   3:          4.7         3.2          1.3         0.2    setosa
##   4:          4.6         3.1          1.5         0.2    setosa
##   5:          5.0         3.6          1.4         0.2    setosa
```

```
##  ---
## 146:          6.7         3.0          5.2         2.3 virginica
## 147:          6.3         2.5          5.0         1.9 virginica
## 148:          6.5         3.0          5.2         2.0 virginica
## 149:          6.2         3.4          5.4         2.3 virginica
## 150:          5.9         3.0          5.1         1.8 virginica

# tibble
iris_tib <- as_tibble(iris)
iris_tib

## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows

# summary statistics
# data.frame way
summary(iris)

##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##

# data.table way
iris_dt[, .(
          count = .N
        , mean_sep = mean(Sepal.Length)
        , mean_pet = mean(Petal.Length)
        ), by = .(Species)]
```

```
##       Species count mean_sep mean_pet
## 1:     setosa    50    5.006    1.462
## 2: versicolor    50    5.936    4.260
## 3:  virginica    50    6.588    5.552

# tibble way
iris_tib %>%
        group_by(Species) %>%
        summarise(
                  count = n()
                , mean_sep = mean(Sepal.Length)
                , mean_pet = mean(Petal.Length)
                )

## # A tibble: 3 x 4
##   Species     count mean_sep mean_pet
## * <fct>       <int>    <dbl>    <dbl>
## 1 setosa         50     5.01     1.46
## 2 versicolor     50     5.94     4.26
## 3 virginica      50     6.59     5.55
```

3 Loops

```
# for loop
x <- seq(3,7,2)
for (i in x){
        print(i^2)
}

## [1] 9
## [1] 25
## [1] 49

# while loop
i <- 3
while(i <= 7){
        print(i^2)
        i <- i+2
}

## [1] 9
## [1] 25
## [1] 49

# repeat loop
i <- 3
repeat{
        print(i^2)
        i <- i+2
        if(i > 7){
```

```
                break
        }
}

## [1] 9
## [1] 25
## [1] 49
```

4 Regressions

We replicate Keizer et al. (2008)'s dataset based on the following information:

- $N_{X=0} = N_{X=1} = 77$;
- $\bar{Y}_{X=0} = 0.33$ and $\bar{Y}_{X=1} = 0.69$.

```
# use Keizer et al. (2008)'s dataset
keizer_data <- data.table(
          x = c(rep(0,77), rep(1,77))
        , y = c(rep(1,25), rep(0,52), rep(1,53), rep(0,24))
        )

# linear model
keizer_lm <- lm(formula = y~x, data = keizer_data)

# summary
summary(keizer_lm)

##
## Call:
## lm(formula = y ~ x, data = keizer_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6883 -0.3247  0.3117  0.3117  0.6753
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.32468    0.05342   6.078 9.42e-09 ***
## x            0.36364    0.07555   4.813 3.55e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4688 on 152 degrees of freedom
## Multiple R-squared:  0.1323,Adjusted R-squared:  0.1265
## F-statistic: 23.17 on 1 and 152 DF,  p-value: 3.553e-06

# attributes
attributes(keizer_lm)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"

# two ways to get coefficients
keizer_lm$coefficients

## (Intercept)           x
##   0.3246753   0.3636364

coefficients(keizer_lm)

## (Intercept)           x
##   0.3246753   0.3636364

# heteroskedasticity robust standard errors
# install.packages("sandwich")
library(sandwich)
# White's estimator
keizer_lm_vcm <- vcovHC(keizer_lm, type = "HC0")
sqrt(diag(keizer_lm_vcm))

## (Intercept)           x
##  0.05336243  0.07505842

# Long & Ervin (2000)
keizer_lm_vcm <- vcovHC(keizer_lm, type = "HC3")
sqrt(diag(keizer_lm_vcm))

## (Intercept)           x
##  0.05406457  0.07604603

# Logit model
keizer_logit <- glm(
          y~x
        , family = binomial(link = "logit")
        , data = keizer_data
        )

# summary
summary(keizer_logit)

##
## Call:
## glm(formula = y ~ x, family = binomial(link = "logit"), data = keizer_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5269  -0.8861   0.8643   0.8643   1.4999
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.7324     0.2434  -3.009  0.00262 **
## x             1.5246     0.3461   4.405 1.06e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 213.46  on 153  degrees of freedom
## Residual deviance: 192.62  on 152  degrees of freedom
## AIC: 196.62
##
## Number of Fisher Scoring iterations: 4
```

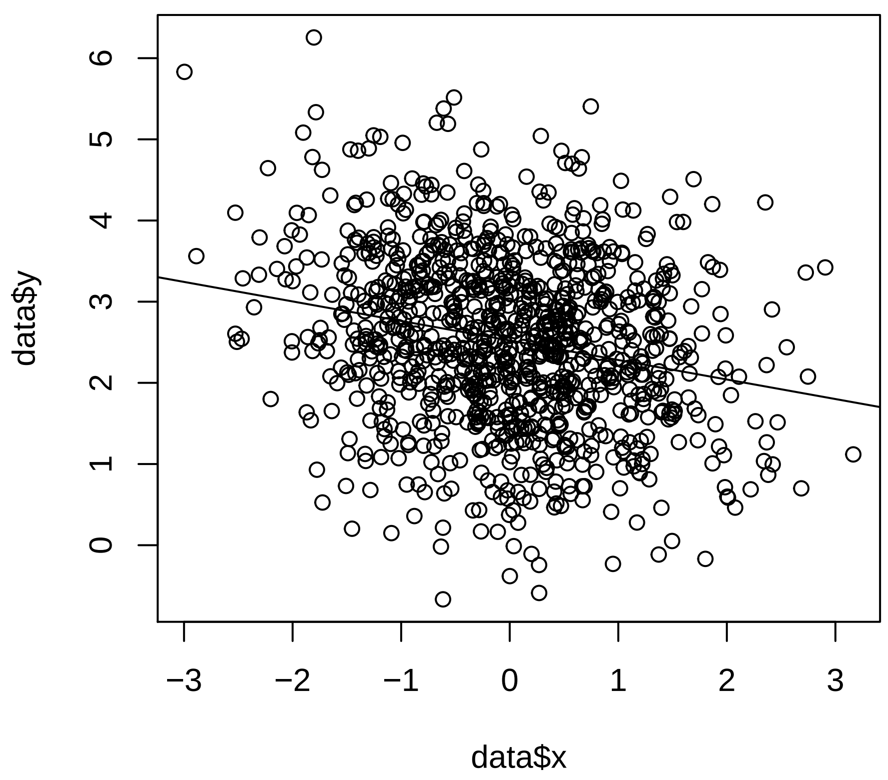
```
# attributes
attributes(keizer_logit)
```

```
## $names
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
##
## $class
## [1] "glm" "lm"
```

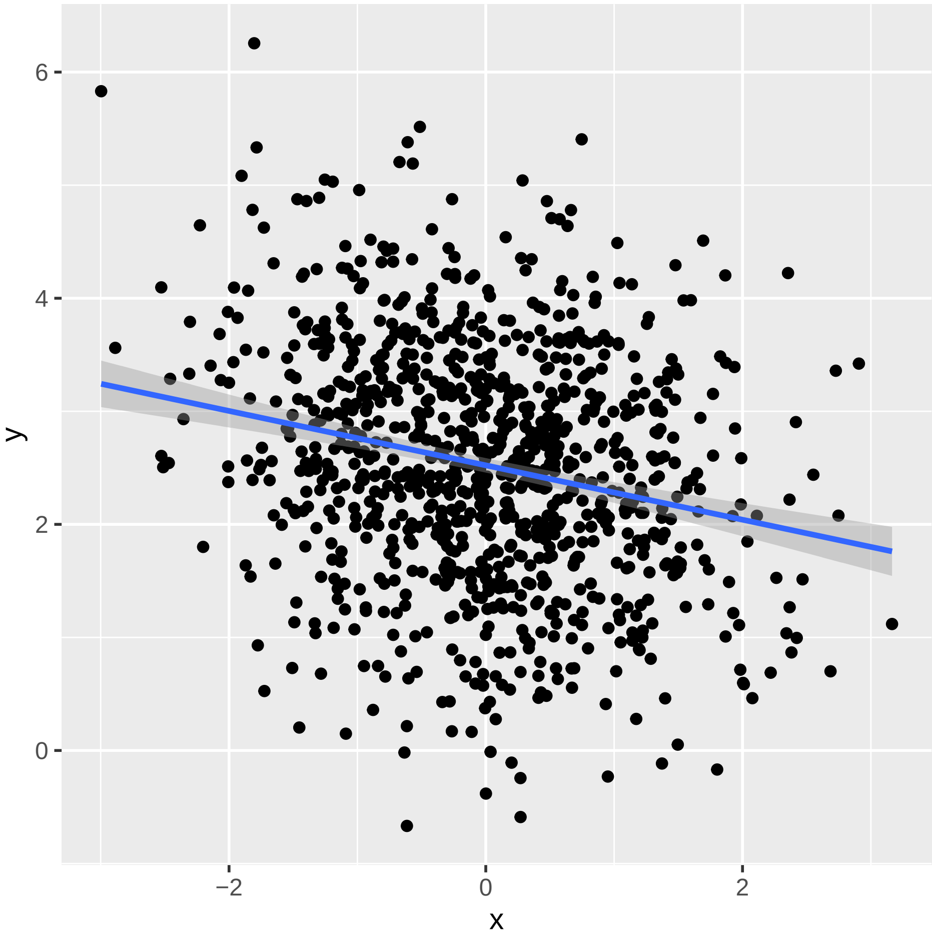
5 Plots

```
# example data
N <- 1000
data <- data.table(
          u = rnorm(N)
        , x = rnorm(N)
        )
data[, y := 2.5-0.25*x+u]

# base style
plot(data$x,data$y)
abline(lm(y~x, data=data))
```




```r
# ggplot style
# install.packages("ggplot")
library(ggplot2)
ggplot(data, aes(x=x, y=y)) +
        geom_point() +
        stat_smooth(formula=y~x, method=lm)
```



6 User-defined Functions and Optimization

We estimate the following model:

1. Data generating process is half $exp(0.5)$ and half $exp(0.75)$.
2. Our model is y_i i.i.d. $\sim exp(\theta)$.
3. Estimate θ by MLE:
   - Log-likelihood: $L(\theta; y_1, \ldots, y_N) = N \log(\theta) - \theta \sum_{i=1}^{N} y_i$
   - Estimator for s.e.: $se(\theta) = \sqrt{\frac{\theta^2}{N}}$

```r
# log-likelihood function
# inputs
# y: vector of samples
# theta: parameter
ll_func <- function(y, theta){
        N <- length(y)
        val <- N*log(theta)-theta*sum(y)
        return(val)
}

# example data
```

```r
theta1 <- 0.5
theta2 <- 0.75
N <- 1000
y1 <- rexp(0.5*N, rate = theta1)
y2 <- rexp(0.5*N, rate = theta2)
y <- c(y1, y2)

# optimization
ll_op <- optimize(
          ll_func
        , interval = c(0.1,10)
        , y = y
        , maximum = TRUE
        )
ll_op

## $maximum
## [1] 0.5789243
##
## $objective
## [1] -1546.582

# estimate for theta
theta_hat <- ll_op$maximum
theta_hat

## [1] 0.5789243

# estimate for s.e.
sqrt(theta_hat^2/N)

## [1] 0.01830719
```