The University of British Columbia

Data Science 581 Modelling and Simulation II

Lab Assignment 1

This lab is about functions, data frames and lists in R.

Functions

Functions in R are used for almost everything. They consist of the following:

1. a name
2. the keyword `function`.
3. a list of arguments surrounded by brackets.
4. statements (usually) surrounded by curly braces.

Consider the following function; It involves resampling from a sample to compute a standard error.

```
SEcalculator <- function(x, FUN, N = 1000) {
     xResample <- numeric(N)
     for (i in 1:N) {
         xResample[i] <-  FUN(sample(x, size = length(x),  replace=TRUE))
     }
     sd(xResample)
}
```

The code above created a function named "SEcalculater". The keyword `function` is used to define a function in R. This function takes three arguments, which are `x`, containing the sample of data values, `FUN`, the name of a function, such as `mean` or `median`, and `N`, the number of times we want to repeat the resampling.
As is often the case, the value of `N` has been pre-specified as 1000; this value can be over-ridden by the user, but it has been chosen, since this will often be a user's preferred choice. It is a *default* value.

An example

The data in `LakeHuron` measure the level of the lake on an annual basis. By taking successive differences, we can obtain the amounts by which the lake changes from year to year, and we may be interested in the median change, as well as an estimate of the standard error of the median; note as argument `N` has a defualt value , we can skipp including that in calling the function unless you want to use values other than 1000.

```
changes <- diff(LakeHuron)
median(changes)  # this calculates the median annual change

## [1] -0.01

SEcalculator(changes, median) # this estimates the standard error

## [1] 0.08107125
```

It appears that the median annual change is near 0 with a large standard error.

Data Frames

Data sets in R are usually stored as data frames. Data frames are a two-dimensional array-like structure. They are similar to matrices, except that the columns are named, we can store different data types (numeric, char,..) in a dataframe. (Matrix can store just one data type)

Examples

Examples of built-in data frames are `ChickWeight` and `women`. The `head()` function shows the first `n` rows:

```
head(women, n = 3)
```

```
##   height weight
## 1     58    115
## 2     59    117
## 3     60    120
```

The data.frame() function is used to construct data frames. vectors `car` and `years` would be the columns of `caryears`.

```
cars <- c("Honda", "Ford", "Buick", "Toyota", "VW")
years <- c(2018, 2012, 2017, 2012, 2013)
caryears <- data.frame(cars, years)
head(caryears, n=2)
```

```
##    cars years
## 1 Honda  2018
## 2  Ford  2012
```

We can use the `names()` function to identify the names of the columns of a data frame:

```
names(caryears)
```

```
## [1] "cars"  "years"
```

To extract the `years` column, use the `$` operator:

```
caryears$years
```

```
## [1] 2018 2012 2017 2012 2013
```

We use the `read.table()` function to read data from an external file (usually .csv) into a data frame:

```
mydata <- read.table("mydata.csv", header=TRUE)
```

The `header` argument tells R that the first row contains the names of the columns. For a data file that contains no names, use `header = FALSE`.

Lists

Lists are convenient for storing objects of a variety of types, and they are often used to return output from functions. List contents can be accessed either by index (like mylist[[1]]) or by name (like mylist$age, whenwe'rehavingtwodimenstionallist.$)

Example

We will modify the `SEcalculator` function to return the estimate and its standard error, and then apply it to the median of the Lake Huron level changes.

```
EstAndSE <- function(x, FUN, N = 1000) {
     xResample <- numeric(N)
     for (i in 1:N) {
         xResample[i] <-  FUN(sample(x, size = length(x),  replace=TRUE))
     }
     list(estimate = FUN(x), SE = sd(xResample))
}
```

```
EstAndSE(changes, median)

## $estimate
## [1] -0.01
##
## $SE
## [1] 0.0813864
```

Using `glm` function ; Poisson

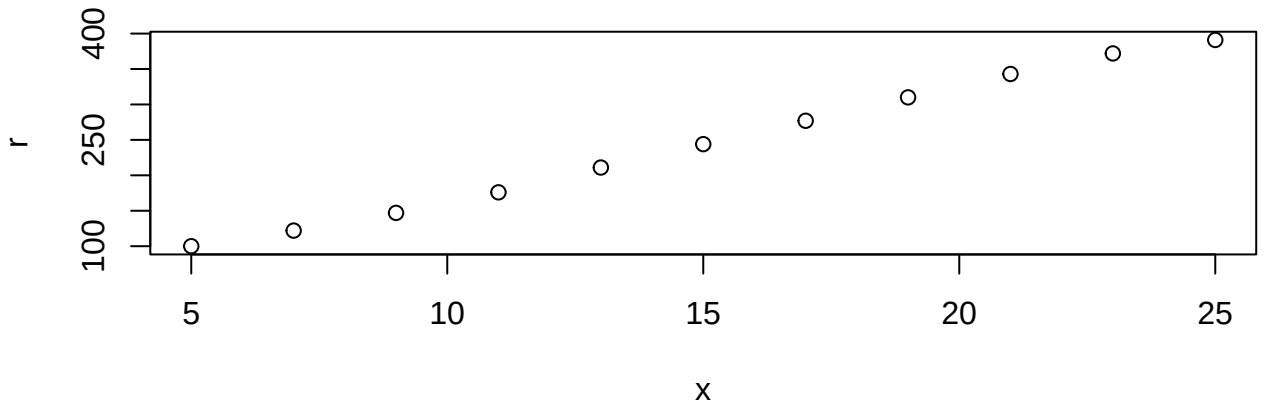
The data set in `p13.4` of the *MPV* package concerns the number r of store coupons redeemed for various discounts, x. 500 coupons of each discount value were used in this study. The counts are actually binomially distributed, but with the large value of n (500), a Poisson approximation could be quite accurate.

Plot the data first, as in the figure below, to see that the relation between count and discount percentage is very close to linear. This means that the usual `log` link for the Poisson is inappropriate. The `identity` link can be used here. It say that the Poisson rate λ is linearly related to discount value x (at least where this will be positive):

$$\lambda(x) = \beta_0 + \beta_1 x$$

for unknown β_0 and β_1.

```
plot(r ~ x, data = p13.4)
```



We can fit this model as follows:

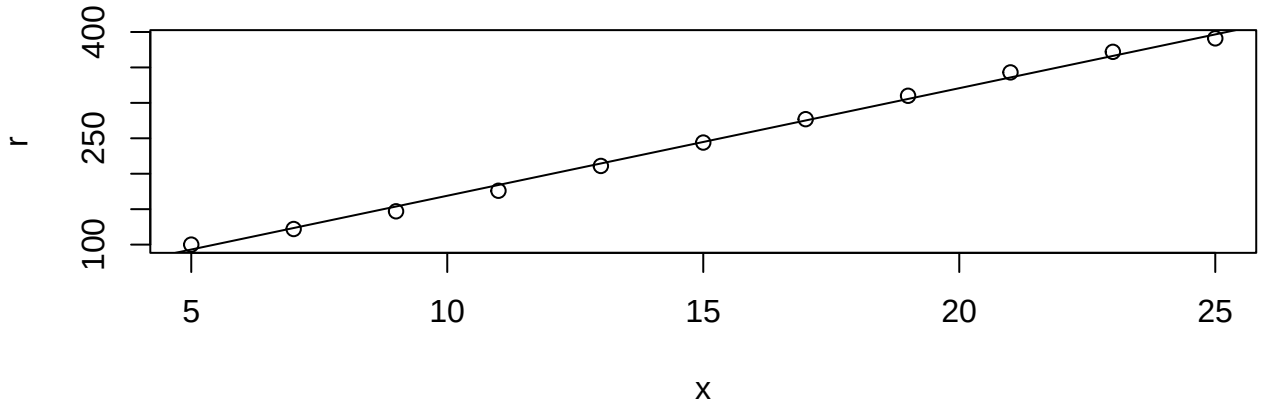
```
p13.4.glm <- glm(r ~ x, data = p13.4, family=poisson(link='identity'))
coef(p13.4.glm)

## (Intercept)           x
##    17.16308    15.17701
```

This tells us that the fitted model is

$$\widehat{\lambda} = 17.1630809 + 15.1770067x.$$

To check that this model fits the data well, consult the plot constructed using

```
plot(r ~ x, data = p13.4)
abline(coef(p13.4.glm))
```



You will learn in a different module that a residual plot is a more effective way of checking goodness of fit.

Using `glm` function ; Logistic Regression

The `anesthetic` data in the *DAAG* package contains 30 observations on the concentration of anesthetic used in a surgical setting and a record as to whether the patient moved or not.

The variable `nomove` is recorded as 1 if there is movement and 0 otherwise. The variable `conc` records the anesthetic concentration. We can fit a logistic model to these data using

We fit a model relating the tendency to not move to concentration as follows:

```
library(DAAG) # load DAAG
anesth.glm <- glm(formula = nomove ~ conc, family = binomial,
    data = anesthetic)
coef(anesth.glm)
```

```
## (Intercept)        conc
##   -6.468675    5.566762
```

The result is reassuring. Movement is less likely as concentration increases. On the logit scale, the model is

$$\text{logit}(x) = -6.47 + 5.57x$$

where x is concentration.

Exercises

Do the following exercises, and hand in #1, #2, #3, #6 and #7 for credit.

1. Enter the function `SEcalculator()` in R. Use it to obtain an estimate of the standard error of the following:
   (a) the mean annual change of level in Lake Huron. Compare with the usual estimate $s/\sqrt{n}$.
   (b) the variance of the annual change of level in Lake Huron.

2. Write a function called `q90()` which takes a single argument `x` and returns the 90th percentile of `x`, (hint: you can use `quantile()` in R to calculate the percentile.
   Then estimate the standard error of the 90th percentile estimate for the annual changes in the Lake Huron water levels.

3. Suppose X_1 and X_2 are independent Bernoulli random variables with common parameter p.
   (a) Write down the likelihood function for p.
   hint: PDF for Bernoulli distribution is:
   $$f(x,p) = p^x(1-p)^1 - x, x \in \{0,1\}$$
   (b) Suppose the observed data are $x_1 = 1$ and $x_2 = 0$. If the possible parameter values lie in the set $\{.2, .7, .9\}$, find the maximum likelihood estimate of p.
   (c) Refer to the preceding part, and suppose the possible parameter values lie in the set $\{p : 0 < p < 1\}$. Find the maximum likelihood estimate of p.

4. Identify the names of the `ChickWeight` data frame, and use the `$` operator and `hist()` to plot a histogram of the weights.

5. Enter the following table of data into a file called `demo.txt`:

   ```
   name age height
   Mary  22   171
   Bob   23   175
   Sue   21   166
   Kim   19   167
   ```

   Read the data into a data frame called `demo`, and use the `$` and `mean()` function to calculate the average of the heights.

6. Consider the `p13.2` data frame in the *MPV* package. Use the following code to obtain it:

   ```
   install.packages("MPV") # if MPV is not installed
   install.packages("boot")
   library(MPV) # loads MPV
   library(boot)
   ```

   (a) Read the help file on `p13.2`, i.e. `help(p13.2)` to obtain information on the data. Why is this an example of data for which binary logistic regression makes sense?
   (b) Fit the binary logistic model to the data, using the `glm()` function, assign the output to an object called `p13.glm`.

(c) By using the command `is.list(p13.glm)`, find out whether the output from `glm()` is a list.

(d) Write out the fitted logit model.

7. Install the package `boot()`. Read the help file for `boot()` which executes the resampling of your dataset and calculation of your statistic(s) of interest on these samples. Before calling boot, you need to define a function that will return the statistic(s) that you would like to bootstrap. The first argument passed to the function should be your dataset. The second argument can be an index vector of the observations in your dataset to use or a frequency or weight vector that informs the sampling probabilities.

   consider the following sample:

```
set.seed(123) # use this seed replicate same result
x <- c(12, 14, 14, 15, 18, 21, 25, 29, 32, 35)

#define function to return mean,
myMean <- function(x,i){mean(x[i])}
```

   We can pass `myMean` to calculate SE for the mean of the above sample.

   Create a function for calculating median and report the SE for statistic using 200 bootstrapping samples by using `boot()`.

8. Consider the `p13.7` data frame in *MPV* package.

   (a) Read the help file on `p13.7`. Why would modelling the number fractures in coal seams as a function of the other variables be an example of Poisson regression?

   (b) Fit a Poisson regression model relating the numbers of fractures to `x2`. Write out the fitted model.