Experiencing with R

Tom Landman

Question 1

What is the sequence of actions R performs when loading the program. In particular, which .R scripts are executed?

In general, according to [1], the sequence of actions R performs at startup are as follows: (1) setting environment variables, (2) running startup scripts, and (3) setting up a session. This relatively complex procedure can be simplified by focusing on two main customization files commonly referred to as *dotRfiles*: *'.Renviron'*, and *'.Rprofile'*. However, *'.Rhistory'*, and *'.RData'* are also being sets at the beginning of the R session.

(1) The first **.Renviron** file found on the R startup search path is processed. **.Renviron** contains a list of environment variables to be set in R sessions and it is not R code, and it uses a format similar to command-line shell.

(2) The first **.Rprofile** file found on the R startup search path is processed. **.Rprofile** runs after the **.Renviron** file is sourced, and it contains R code to be run when R starts up.

(3) If the **.Rprofile** file in calls startup::startup() then the following will also take place:

   a. The first **.Renviron.d** directory on the R startup search path is processed. The search path is (in order):

      (i) **paste0(Sys.getenv("R_ENVIRON_USER"), ".d")**, (ii) **./.Renviron.d**, and (iii) **~/.Renviron.d**. Also, note that the format of these files should be the same as for **.Renviron**.

   b. A set of handy R options that can be use in Step 3c are set. Their names are prefixed startup.session.

   c. The first **.Rprofile.d** directory found on the R startup search path is processed. The search path is (in order):

      (i) **paste0(Sys.getenv("R_PROFILE_USER"), ".d")**, (ii) **./.Rprofile.d**, and (iii) **~/.Rprofile.d**. The format of these files should be the same as for **.Rprofile**, that is, they must be valid R scripts.

   d. If no errors occur above, the startup package will be unloaded, leaving no trace of itself behind, except for R options startup.session.

Also, R automatically saves a **.Rhistory** file in the R working directory. It contains the history of commands entered by the user during an open R session and enables the user to push the up arrow to scan through the list of previously entered commands.

Furthermore, **.RData** is also a file created by R, and it used for storing a working environment that can be extended or modified later. Hence, **.RData** saves a workspace, which includes the function and value objects created during an open session in R.

Question 2

Write a function that generates a random matrix (see **Eq. 1** below), subtract the mean, and divide by the standard deviation to return a matrix of the z-scores of each column.

$$A <- matrix(rnorm(40), nrow = 10, ncol = 4) \ (\textbf{Eq.1})$$

The function **normal.matrix.with.standardized.columns** implemented below, generates a matrix with normally distributed random numbers and standardized each columns separately (i.e., column-wise z-scores).

Inputs:

- **n:** number of elements.
- **nrow:** number of rows.
- **ncol:** number of columns.

Output:

- **standardized__by__columns:** a matrix of the z-scores of each column.

```
normal.matrix.with.standardized.columns <- function(n=40, nrow=10, ncol=4){

  # Generate a matrix of normally distributed random numbers
  # with a dimension of (nrow,ncol)
  A <- matrix(rnorm(n), ncol=ncol, nrow=nrow)

  # Creates a matrix (same dims as the A) based on columns means of A
  means_matrix <- rep(colMeans(A), rep.int(nrow(A), ncol(A)))

  # Creates a matrix (same dims as the A) based on columns SDs of A
  stdev_matrix <- rep(apply(A, 2,sd), rep.int(nrow(A), ncol(A)))

  # Convert A to Z scores (column-wise) - calculated by subtracting column mean,
  # and divide by column's standard deviation
  standardized_by_columns <- (A - means_matrix)/ stdev_matrix

  return(standardized_by_columns)
}

# Generate a random matrix
Standardized_A <- normal.matrix.with.standardized.columns()

# Print matrix
print(Standardized_A)
```

```
##             [,1]        [,2]        [,3]        [,4]
##  [1,] -0.1396117  0.51231505 -1.25951575 -1.15510519
##  [2,] -1.1216234 -1.39145196 -0.01272829 -1.05374681
##  [3,]  1.4630562 -0.21324393 -1.79664664  0.04306859
##  [4,]  0.7798029 -0.72304834  0.28745972 -0.16167316
##  [5,]  0.1885803 -1.46385136 -0.03114493 -0.09524578
##  [6,]  0.4729262  0.01353270  1.69329599  0.98812888
##  [7,] -0.5775969  1.29680823  0.48831349 -1.15289028
##  [8,] -1.4771346 -0.06848371 -0.53135526  1.88297072
##  [9,] -0.8223944  0.61366205  0.44263250 -0.10546308
## [10,]  1.2339953  1.42376129  0.71968919  0.80995612
```

Question 3

Using the last column in (2), create an indicator for each observation being above/below the column's median.

Assumption:
In case where a given value is equal to the exact median (i.e., when the number of rows is odd), it will be marked as 1.

```
# Extract the last column of the standardized matrix generated using the function implemented in (2)
last_column = Standardized_A[,ncol(Standardized_A)]

# Calculate indicator vector based on the last column (i.e., if a given value is
# above the last column's median the indicator will be equal to 1, otherwise 0)
indicator <- sapply(last_column, function(x) ifelse(x >= median(last_column), 1, 0))

# Combine last column and indicator columns
columns <- cbind(last_column, indicator)

# Print last column's median
cat(paste(c("Last column median:\n",median(last_column))), collapse="\n")
```

```
## Last column median:
##  -0.100354433146745
```

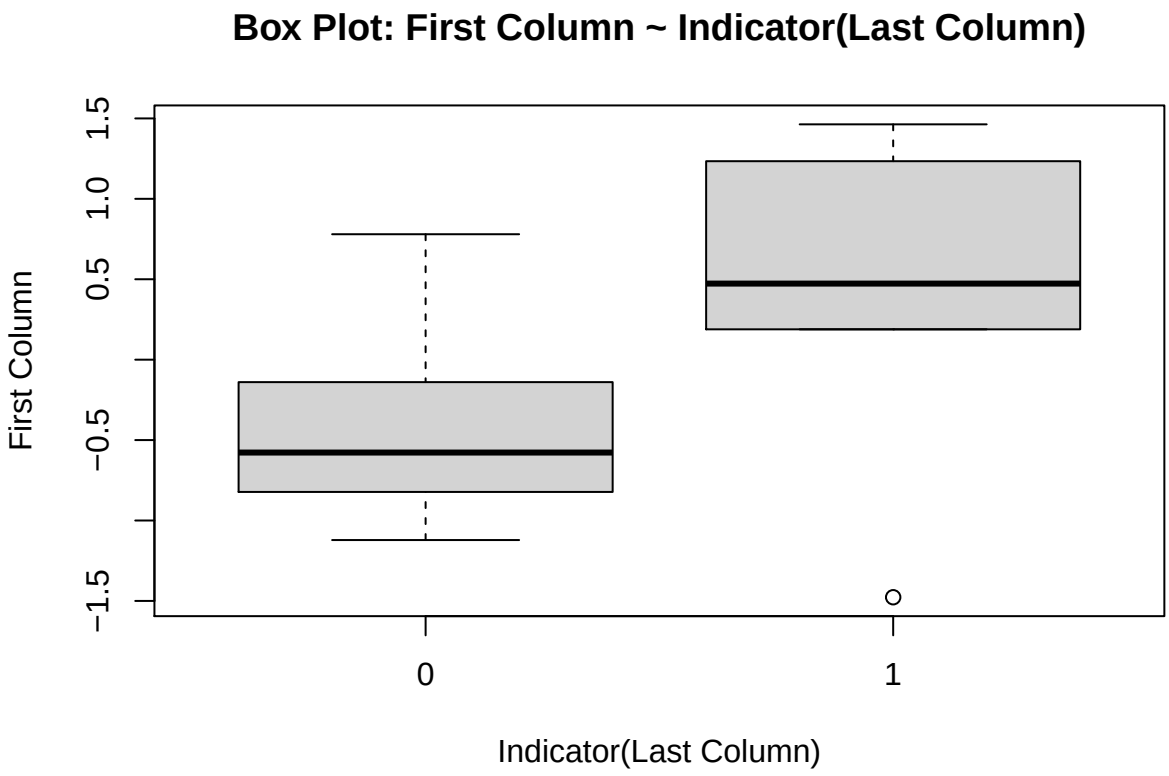
```
# Print last column and indicator columns
print(columns)
```

```
##       last_column indicator
##  [1,] -1.15510519         0
##  [2,] -1.05374681         0
##  [3,]  0.04306859         1
##  [4,] -0.16167316         0
##  [5,] -0.09524578         1
##  [6,]  0.98812888         1
##  [7,] -1.15289028         0
##  [8,]  1.88297072         1
##  [9,] -0.10546308         0
## [10,]  0.80995612         1
```

Question 4

Using the first column in (2), create a boxplot, for each level of the indicator in (3).

```
# Extract first column from Standardized_A matrix
first_column = Standardized_A[,1]

# Create a boxplot for each level of the indicator in (3)
boxplot(first_column~indicator, main="Box Plot: First Column ~ Indicator(Last Column)",
        ylab="First Column", xlab="Indicator(Last Column)")
```



Question 5

Create a matrix with independent entries from Unif[0,1] distribution. 1e2 rows, and 1e1 columns.

a. Convert it to a data.table.
b. Change the first column with a series that starts at 1, ends at your id number, and increases linearly.

```
# Import data.table library
library(data.table)

# Generate a matrix of uniformly distributed random numbers (i.e., ~U(0,1))
# with a dimension of (100,10).
U_matrix <- matrix(runif(1000), nrow=1e2, ncol=1e1)

# Convert U_matrix to a data.table object
U_marix_as_DT <- data.table(U_matrix)

# Override the 1st column with a series that starts at 1, ends at 301804639,
# and increases linearly.
U_marix_as_DT[,1] <- seq(from=1, to=301804639, length.out=nrow(U_marix_as_DT))
```

```r
# Print the first six rows and columns of the updated matrix
print(head(U_marix_as_DT[,1:6]))
```

```
##          V1        V2         V3        V4        V5         V6
## 1:        1 0.8259305 0.87870811 0.2021551 0.2607576 0.03756358
## 2:  3048533 0.4132272 0.27261396 0.4320498 0.9637548 0.62901673
## 3:  6097064 0.9527028 0.60179789 0.3558345 0.2413215 0.64983380
## 4:  9145596 0.2515999 0.13061974 0.8317734 0.8346874 0.03112017
## 5: 12194128 0.8875584 0.06845656 0.3203039 0.2706630 0.64327739
## 6: 15242659 0.2019443 0.64486301 0.7618128 0.3504198 0.72282268
```

```r
# Print the last six rows and columns of the updated matrix
print(tail(U_marix_as_DT[,1:6]))
```

```
##           V1        V2        V3        V4         V5        V6
## 1: 286561981 0.9237637 0.5434301 0.1159459 0.80664201 0.3914015
## 2: 289610512 0.9194311 0.5412344 0.5512113 0.16131324 0.5704085
## 3: 292659044 0.9183134 0.8771687 0.5068234 0.41715020 0.6300058
## 4: 295707576 0.4313991 0.3879157 0.5241558 0.01955458 0.7002255
## 5: 298756107 0.8568909 0.6667727 0.4954307 0.25162634 0.5085255
## 6: 301804639 0.9628809 0.3141139 0.9138044 0.95730603 0.4371978
```

Question 6

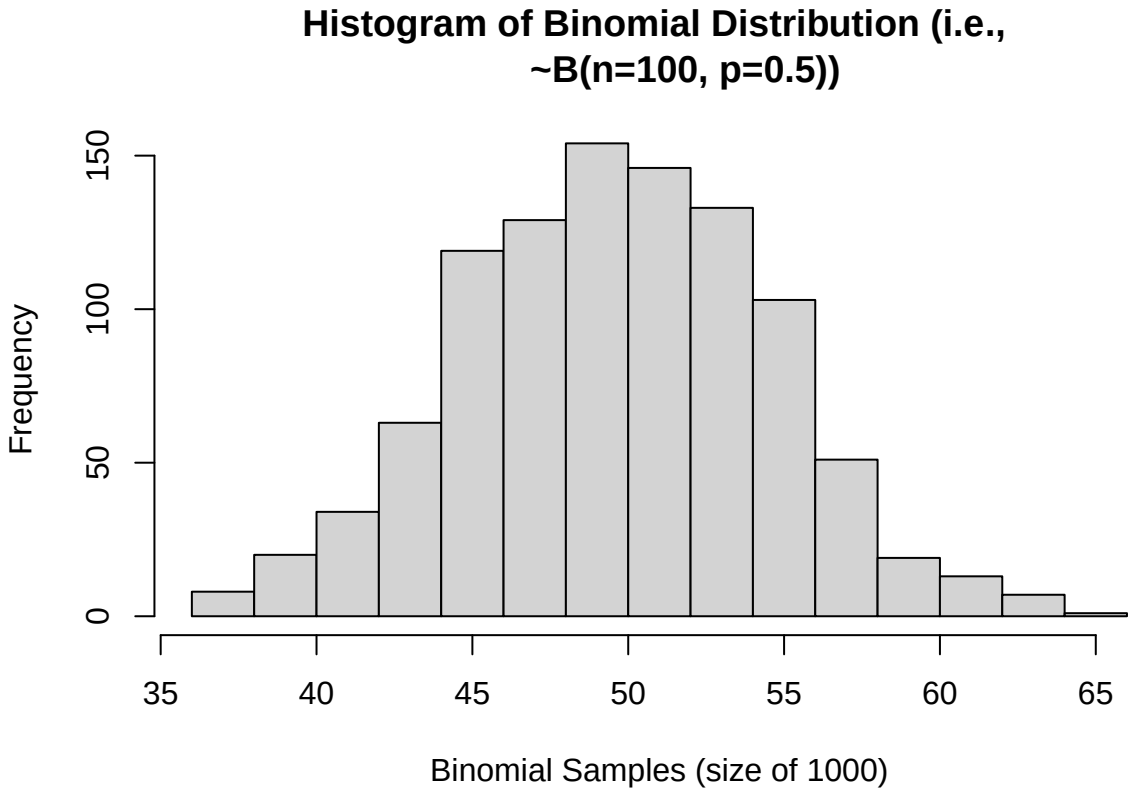
Generate 1e3 samples from Binom(n=100,p=0.5). What plot is best suited for this data?

Based on the generation of the 1e3 samples of the binomial distribution, which is a discrete distribution, it can be visualized using a histogram. As shown in the histogram below, it can be seen that the expectation μ is exactly 50 which is equal to the binomial expectation given **Eq. 2** below:

$$\mu = n \cdot p = 100 \cdot 0.5 = 50 \ (\textbf{Eq.2})$$

Also, due to a relatively high sample size consisting of 1000 samples, the binomial distribution is approximately symmetric and can be approximated to the Normal distribution.

```r
# Generate 1000 samples from Binomial(n=100,p=0.5)
binomial_samples <- rbinom(n=1e3, size=100, prob=0.5)

# Display a histogram given the binomial vector above
hist(binomial_samples, main = "Histogram of Binomial Distribution (i.e.,
     ~B(n=100, p=0.5))", xlab="Binomial Samples (size of 1000)")
```



Question 7

Read in Wikipedia on Hotelling's two-group test statistic, and implement it in your own function. Make sure to comment it nicely so I can read it.

According to Wikipedia [2], the Hotelling's two-group test statistic arises in multivariate statistics in undertaking tests of the differences between the (multivariate) means of different groups (i.e., populations), where tests for univariate problems would make use of a t-test. The distribution is named for Harold Hotelling, who developed it as a generalization of Student's t-distribution.

Here are the steps to conduct and calculate the estimators for Hotelling's two-group test statistic:

First, the sample means vectors of both multivariate matrices are being calculated using **Eq. 3**, and **Eq. 4** below:

$$\bar{x} = \frac{1}{n_x}\sum_{i=1}^{n_x} x_i \ (\textbf{Eq.3}); \quad \bar{y} = \frac{1}{n_y}\sum_{i=1}^{n_y} y_i \ (\textbf{Eq.4})$$

Second, both covariance matrices are calculated using the equations below (1st & 2nd groups) for both X (**Eq. 5**), and Y (**Eq. 6**):

$$\hat{\Sigma x} = \frac{1}{n_x - 1}\sum_{i=1}^{n_x} (x_i - \bar{x})(x_i - \bar{x})' \ (\textbf{Eq.5}); \quad \hat{\Sigma y} = \frac{1}{n_y - 1}\sum_{i=1}^{n_y} (y_i - \bar{y})(y_i - \bar{y})' \ (\textbf{Eq.6})$$

Third, given both covariance matrices above, the unbiased pooled covariance matrix estimate is calculated using **Eq. 7** below:

$$\hat{\Sigma} = \frac{(n_x - 1)\hat{\Sigma x} + (n_y - 1)\hat{\Sigma y}}{n_x + n_y - 2} \ (\textbf{Eq.7})$$

Then, the Hotelling's two-sample t-squared statistic can be calculated based on **Eq. 8**:

$$T^2 = \frac{n_x n_y}{n_x + n_y}(\bar{x} - \bar{y})' \hat{\Sigma}^{-1}(\bar{x} - \bar{y}) \sim T^2(p,\ n_x + n_y - 2) \text{ (Eq.8)}$$

Last, the non-null distribution of the Hotelling's statistic, or the noncentral F-distribution (the ratio of a non-central Chi-squared random variable and an independent central Chi-squared random variable) is calculated using **Eq. 9** below:

$$F = \frac{n_x + n_y - p - 1}{(n_x + n_y - 2) \cdot p} t^2 \sim F(p,\ n_x + n_y - 1 - p) \text{ (Eq.9)}$$

Under null hypothesis, $H_0 : \mu = \mu_0$, this will have a F distribution with p and $n - p$ degrees of freedom. We reject the null hypothesis, H_0, at level α if the test statistic F is greater than the critical value from the F-table with p and $n - p$ degrees of freedom, evaluated at level α. (i.e., $F > F_{p,\ n-p,\ \alpha}$)

Inputs:

- **x:** a vector of the first group samples.
- **y:** a vector of the second group samples.
- **alpha:** test significance level, denoted as α, default is 0.05.

Output:

- **list** of **hotteling's statistic**, **F-statistic**, **df1**, **df2** & **p-value**.

```
Hotelling.two.sample.t.squared.statistics <- function(X, Y, alpha=0.05){

  # Extract number of rows in matrix X, denoted as n_X
  n_X <- nrow(X)

  # Extract number of rows in matrix Y, denoted as n_Y
  n_Y <- nrow(Y)

  # Extract number of columns, denoted as p
  p <- ncol(X)

  # Calculate means vector for both X and Y matrices
  mean_vec_X <- colMeans(X)
  mean_vec_Y <- colMeans(Y)

  # Calculate covariance matrices for both X and Y
  sigma_X <- cov(X)
  sigma_Y <- cov(Y)

  # Calculate the pooled covariance matrix
  pooled_covariance_matrix <- ((n_X-1)*sigma_X+(n_Y-1)*sigma_Y)/(n_X + n_Y - 2)

  # Calculate the inverse of the pooled covariance matrix
  pooled_covariance_matrix_inv <- solve(pooled_covariance_matrix)

  # Calculate the Hotelling's two-sample t-squared statistic
  T2 <- round(((n_X*n_Y)/(n_X+n_Y))*(mean_vec_X-mean_vec_Y)%*%
         (pooled_covariance_matrix_inv)%*% (mean_vec_X-mean_vec_Y),6)

  # Calculate the F-statistic based on the Hotelling's statistic
  F_stat <- round(((n_X+n_Y-p-1)/((n_X+n_Y-2)*p))*T2,6)
```

```r
  # Calculate degrees of freedom (i.e., df1 & df2) to find F critic
  df1 <- p
  df2 <- n_X+n_Y-1-p

  # Calculate p-value given the F-statistic
  p.value <- round(pf(F_stat, df1, df2, lower.tail = FALSE), 6)

  # Print calculated statistics
  cat(paste("T2 = ",T2,", F = ",F_stat,", df1 = ",df1,", df2 = ",df2,",
            p-value =",p.value,"\n", sep = ""))

  # Testing hypothesis
  if(p.value > alpha){
    cat(paste("There is a true difference in mean vectors.\n",
    "There is sufficient evidence to reject the null hypothesis.","\n",
    "Significane level of ", alpha,".\n", sep=""))
  }
  else{
    cat(paste("There is no difference in mean vecors.\n","There is a lack of
              evidence to reject the null hypothesis in significane level of ",
              alpha,".\n"), sep="")
  }

  return(list(T2=T2,F=F_stat,df1=df1,df2=df2,pvalue=p.value))
}

# Import mvtnorm for multivariate matrix generation
library(mvtnorm)

# Import rrcov for comparing result as suggested in R course book (i.e., sanity check)
library(rrcov)
```

```
## Loading required package: robustbase
```

```
## Scalable Robust Estimators with High Breakdown Point (version 1.5-5)
```

```r
# Set seed for reproducibility
set.seed(1111)

# Number of observations
n <- 100

# Parameter dimension, denoted as p
p <- 25

# Vector of means, denoted as mu
mu <- rep(0,p)

# Generate two multivariate gaussian distribution matrices B and C
B <- rmvnorm(n = n, mean = mu)
C <- rmvnorm(n = n, mean = mu)
```

```r
# Test function given two multivariate gaussian distribution matrices B and C
statistics_list <- Hotelling.two.sample.t.squared.statistics(B, C)
```

```
## T2 = 21.514466, F = 0.756266, df1 = 25, df2 = 174,
##             p-value =0.791819
## There is a true difference in mean vectors.
## There is sufficient evidence to reject the null hypothesis.
## Significane level of 0.05.
```

```r
# Compare results using Hotelling's built in function of the rrcov's package
rrcov::T2.test(B, C)
```

```
##
##  Two-sample Hotelling test
##
## data:  B and C
## T2 = 21.51447, F = 0.75627, df1 = 25, df2 = 174, p-value = 0.7918
## alternative hypothesis: true difference in mean vectors is not equal to (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
## sample estimates:
##                       [,1]        [,2]       [,3]       [,4]         [,5]
## mean x-vector -0.09899625  0.13740597 0.14588086 0.03498209 -0.066646002
## mean y-vector  0.02320222 -0.04798894 0.03085515 0.06816952  0.009616913
##                       [,6]       [,7]        [,8]         [,9]       [,10]
## mean x-vector -0.02393879 0.03767514 -0.07140813 -0.001899419 -0.07026315
## mean y-vector -0.03716904 0.08394026  0.10556921 -0.050544302 -0.12425319
##                      [,11]      [,12]      [,13]       [,14]     [,15]
## mean x-vector -0.14273321 0.02975608 -0.1189100 -0.08497252 0.1042766
## mean y-vector  0.02358039 0.09394074 -0.1543268  0.06794559 0.1713054
##                      [,16]       [,17]       [,18]        [,19]       [,20]
## mean x-vector 0.16661289  0.01604197 -0.12968798 0.0003075019 -0.02334147
## mean y-vector 0.03723313 -0.28794692 -0.04254946 0.0841274603 -0.05471361
##                      [,21]      [,22]       [,23]        [,24]       [,25]
## mean x-vector -0.04144310  0.1479946 -0.12508144 -0.006178023 -0.09382899
## mean y-vector  0.05606944 -0.1121953 -0.01450819  0.004812673 -0.21915927
```

Question 8

Fit a linear model to explain the number of dead insects in the InsectSprays dataset. Do it with the lm() function, and then with the aov() function. Now explain the differences in the output of summary() on these two objects.

Based on the outputs obtained from both ANOVA models and linear regression, the following can be deduced:

1. Based on the ANOVA output it can be seen that there's a significance evidence (i.e., P-value lower than 2e-16 in significance level of $\alpha = 0.0001$) to reject the ANOVA's NULL hypothesis (i.e., H_0 : *There is no difference between groups*, H_1 : *otherwise*). Hence, there is a difference between at least 2 sprays in the InsectSprays dataset.

2. Based on the Linear regression output it can be seen that there's a significance evidence (i.e., P-value lower than 2.2e-16 in significance level lower than $\alpha = 0.0001$) to reject the Linear Regression model's NULL hypothesis (i.e., $H_0 : \beta_i = 0 \ \forall i \in [1, k]$; H_1 : *otherwise*). Also, note that the calculated F score is equal to the F score calculated in the ANOVA model.

3. The spray categorical feature is the only independent variable appears in the InsectSprays dataset. hence, by viewing the p-value for each β coefficient it can be seen that the impact of sprays C, D & E on the amount of dead insects is significantly different from the effect of the base group which is spray A (significance level lower than $\alpha = 0.0001$), whereas sprays B & F impact on the amount of dead insects is not significantly different from spray A effect (P-values of 0.604 and 0.181 respectively).
4. based on the significant coefficients' values, it can be seen that spray A is the most effective spray to destroy insects (intercept of 14.5).
5. Spray D is less effective than spray A, resulting in intercept of $14.5 - 9.5833 = 4.9167$.
6. Spray E is less effective than spray A & D, resulting in intercept of $14.5 - 11 = 3.5$.
7. Spray C is less effective than spray A, D & E, resulting in intercept of $14.5 - 12.4167 = 2.0833$.
8. Although both sprays B and F do not significantly affect the number of dead insects compared to the effect of spray A, it is not possible to deduce which one is more effective.

The outputs are presented below:

```
# Create an ANOVA model based on InsectSprays dataset.
anova_model <- aov(count ~ spray, data = InsectSprays)

# Print ANOVA model summary
cat("ANOVA Output:", collapse="\n")
```

```
## ANOVA Output:
```

```
print(summary(anova_model))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## spray        5   2669   533.8    34.7 <2e-16 ***
## Residuals   66   1015    15.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Create a Linear Regression model based on InsectSprays dataset.
lr_model <- lm(count ~ spray, data = InsectSprays)

# Print Linear Regression summary
cat(paste("\n","Linear Regression Output:"))
```

```
##
##  Linear Regression Output:
```

```
print(summary(lr_model))
```

```
##
## Call:
## lm(formula = count ~ spray, data = InsectSprays)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -8.333 -1.958 -0.500  1.667  9.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.5000     1.1322  12.807  < 2e-16 ***
## sprayB        0.8333     1.6011   0.520    0.604
## sprayC      -12.4167     1.6011  -7.755 7.27e-11 ***
## sprayD       -9.5833     1.6011  -5.985 9.82e-08 ***
## sprayE      -11.0000     1.6011  -6.870 2.75e-09 ***
## sprayF        2.1667     1.6011   1.353    0.181
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.922 on 66 degrees of freedom
## Multiple R-squared:  0.7244, Adjusted R-squared:  0.7036
## F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

Question 9

Write a function that takes an lm class object, and returns the p-value of the intercept, against a one-sided alternative. Apply your function on the model log(volume)~log(height)+log(girth) in the trees dataset.

The function **calculate.one.sided.Pval** implemented below, takes an lm class object, and returns the p-value of the intercept, against a one-sided alternative.

Assumptions:

1. User specifies which one side test to use (lower/upper tail).
2. User specifies the intercept value under the null hypothesis, default is zero (i.e., $H_0 : \beta_0 = 0$).
3. The significance level is fixed with $\alpha = 0.05$.

Inputs:

- **lm_obj:** lm class object.
- **lower_tail:** logical; if TRUE (default), probability is $P(X \leqslant x), otherwise, P(X \geqslant x)$.

Output:

- **p_value:** the p-value of the intercept, against the chosen one-sided alternative.

```
calculate.one.sided.Pval <- function(lm_obj, beta_H0=0, lower_tail = TRUE){

  # Create summary of the inputted linear regression model
  summary_lm <- summary(lm_obj)

  # Subtract the intercept's estimate by its value under H0 and divide by its stdev.
  intercept_t_value <- (coef(summary_lm)[1,1]- beta_H0) / coef(summary_lm)[1,2]

  # Extract degrees of freedom
  df <- lm_obj$df

  # Calculate the intercept's the p-value, against the chosen one-sided alternative
  p_value <- round(pt(intercept_t_value, df=df , lower.tail = lower_tail), 5)

  # Print the suitable output given the chosen one sided test
  if (lower_tail){
    cat(paste("One sided t test for intercept (Lower tail) has P value of",
```

```
                p_value, " in significance of 0.05."))
  }
  else{
  cat(paste("One sided t test for intercept (Upper tail) has P value of",
                p_value, " in significance of 0.05."))  }
  return(p_value)
}

# Create a linear regression model based on trees dataset
lm_trees <- lm(log(Volume)~log(Height)+log(Girth), data = trees)

# Execute one sided t test (upper tail)
upper <- calculate.one.sided.Pval(lm_trees, 0, FALSE)
```

```
## One sided t test for intercept (Upper tail) has P value of 1  in significance of 0.05.
```

```
# Execute one sided t test (lower tail)
lower <- calculate.one.sided.Pval(lm_trees, 0, TRUE)
```

```
## One sided t test for intercept (Lower tail) has P value of 0  in significance of 0.05.
```

Question 10

In the airquality data ignore the Temp variable. Use linear regression to impute (fill the missing observations) in all remaining columns, after ignoring Temp all the way.

```
# Load airquality dataset
airquality_mod <- airquality

# Convert Day & Month columns from integers to factors
airquality_mod[,"Day"] <- as.factor(airquality_mod[,"Day"])
airquality_mod[,"Month"] <- as.factor(airquality_mod[,"Month"])

# Slice out Temp column
airquality_no_Temp <- subset(airquality, select = -c(Temp))

# Slice out Solar.R column (in addition to Temp)
airquality_no_Temp_Solar <- subset(airquality_no_Temp, select = -c(Solar.R))

# Slice out Ozone column (in addition to Temp)
airquality_no_Temp_Ozone <- subset(airquality_no_Temp, select = -c(Ozone))

# Create Linear Regression model to predict Ozone given all features except Temp
lm_full_Ozone <- lm(Ozone~., data = airquality_no_Temp)

# Create Linear Regression model to predict Ozone given all features except Temp
# & Solar.R (relevant in cases where both Ozone and Solar.R has Missing values)
lm_partial_Ozone <- lm(Ozone~., data = airquality_no_Temp_Solar)

# Create Linear Regression model to predict Solar.R given all features except Temp
lm_full_Solar.R <- lm(Solar.R~., data = airquality_no_Temp)
```

```r
# Create Linear Regression model to predict Solar.R given all features except Temp
# & Ozone (relevant in cases where both Solar.R and Ozone has Missing values)
lm_partial_Solar.R <- lm(Solar.R~., data = airquality_no_Temp_Ozone)

# Iterate columns with missing values (i.e., Ozone & Solar.R)
for (column in c("Ozone", "Solar.R")){

  # Extract current column's missing values indices
  missing_indices <- c(which(is.na(airquality_mod[,column])))

  # Differentiate between each case, since different linear regression models
  # are used based on the column to be predicted
  if(column == "Ozone"){

    # Complete missing values in cases of records where only Ozone is missing
    # Note that values are rounded since Ozone is a column of integers
    airquality_mod[missing_indices, column] <-
      round(predict(lm_full_Ozone, airquality_no_Temp[missing_indices,]))

    # Extract missing values indices after first values completion step
    missing_indices <- c(which(is.na(airquality_mod[,column])))

    # Checks if there are still missing values in Ozone column
    if(length(missing_indices)>0){

      # Complete missing values in cases where both Solar.R and Ozone columns
      # have Missing values
      airquality_mod[missing_indices, column] <-
      round(predict(lm_partial_Ozone, airquality_no_Temp_Solar[missing_indices,]))
    }
  }

  else{

    # Complete missing values in cases of records where only Solar.R is missing
    # Note that values are rounded since Solar.R is a column of integers
    airquality_mod[missing_indices, column] <-
      round(predict(lm_full_Solar.R, airquality_no_Temp[missing_indices,]))

    # Extract missing values indices after first values completion step
    missing_indices <- c(which(is.na(airquality_mod[,column])))

    # Checks if there are still missing values in Ozone column
    if(length(missing_indices)>0){

      # Complete missing values in cases where both Solar.R and Ozone columns
      # have Missing values
      airquality_mod[missing_indices, column] <-
      round(predict(lm_partial_Solar.R, airquality_no_Temp_Ozone[missing_indices,]))

    }
  }
}
```

The first twenty rows of the **airquality** dataset before missing values completion:

```
# print airquality dataset after missing values completion
print(head(airquality, 20))
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
## 11     7      NA  6.9   74     5  11
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
## 15    18      65 13.2   58     5  15
## 16    14     334 11.5   64     5  16
## 17    34     307 12.0   66     5  17
## 18     6      78 18.4   57     5  18
## 19    30     322 11.5   68     5  19
## 20    11      44  9.7   62     5  20
```

The first twenty rows of the **airquality** dataset after missing values completion:

```
# print airquality dataset after missing values completion
print(head(airquality_mod, 20))
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     15     207 14.3   56     5   5
## 6     28     206 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    47     194  8.6   69     5  10
## 11     7     152  6.9   74     5  11
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
## 15    18      65 13.2   58     5  15
## 16    14     334 11.5   64     5  16
## 17    34     307 12.0   66     5  17
## 18     6      78 18.4   57     5  18
## 19    30     322 11.5   68     5  19
## 20    11      44  9.7   62     5  20
```

Question 11

Create synthetic data to be analyzed using a Poisson regression. Guidelines: 1e4 observations; two covariates (independent), and one dependent (outcome); each of the covariates has mean 0; effects (beta) are 2 and -1.

```
# Setting seed for reproducibility
set.seed(301804639)

# Create a vector of ones for the intercept
ones <- rep(1, 1e4)

# Generate the first independent variable x1 using standard normal distribution
x1 <- rnorm(1e4)

# Generate the second independent variable x1 using standard normal distribution
x2 <- rnorm(1e4)

# Combine all three columns into a single matrix
X <- cbind(x1, x2)

# Create a vector of coefficients (i.e., betas) based on the given instructions
coefficients<- c(2,-1)

# Generate the dependent variable y using the poisson distribution where
# lambda equal to the exponent of the multiplication between matrix X and the
# respective coefficients.
y <- rpois(n=1e4, lambda = exp(X %*% coefficients))
```

Question 12

Fit a Poisson regression model to the data from item (11). Are the estimated coefficients close to the true ones?

As shown in the poisson regression output below, it seems that the estimated coefficients are close to the true ones with a slight change, where the first slope coefficient, β_1, has an additional $\varepsilon_1 = 0.004267$, whereas the second slope coefficient, β_2 has an additional $\varepsilon_2 = 2.98 \cdot 10^{-4}$.

```
# Fit a Poisson regression model to the data from item (11)
poisson_regression <- glm(y ~ x1 + x2, family = poisson)

# Print a summary based on the Poisson regression model
summary(poisson_regression)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8440  -0.7499  -0.3035   0.4080   3.7278
##
## Coefficients:
##             Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -0.006511   0.007739   -0.841      0.4
```

```
## x1           2.004267   0.003276  611.754   <2e-16 ***
## x2          -0.999702   0.003007 -332.504   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 548955.9  on 9999  degrees of freedom
## Residual deviance:   8795.6  on 9997  degrees of freedom
## AIC: 28746
##
## Number of Fisher Scoring iterations: 4
```

Question 13

Read about the ChickWeight dataset. Which model fits the data/story?

The ChickWeight data frame [2] has 578 rows and 4 columns (i.e., weight, time, chick, & diet) captured from an experiment regarding the effect of diet on early growth of chicks. The body weights of the chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups on chicks on different protein diets. Based on the description of the given experiment, it seems that the appropriate model to apply here is Linear Mixture Model (LMM), since each subject's (i.e., chick) weight was sampled every second day, from the moment of birth until day 21 (including). Hence, this DOE is within-subject where each chick has 12 observations. As a result, by using LMM the random effect can be also taken into consideration.

Question 14

Create data (given the code below) and make a scree plot of the explained variance of each principal component.

```
  set.seed(your id number)
  mu <- rexp(50,6)
  multi <- rmvnorm(n = 100, mean = mu)
```

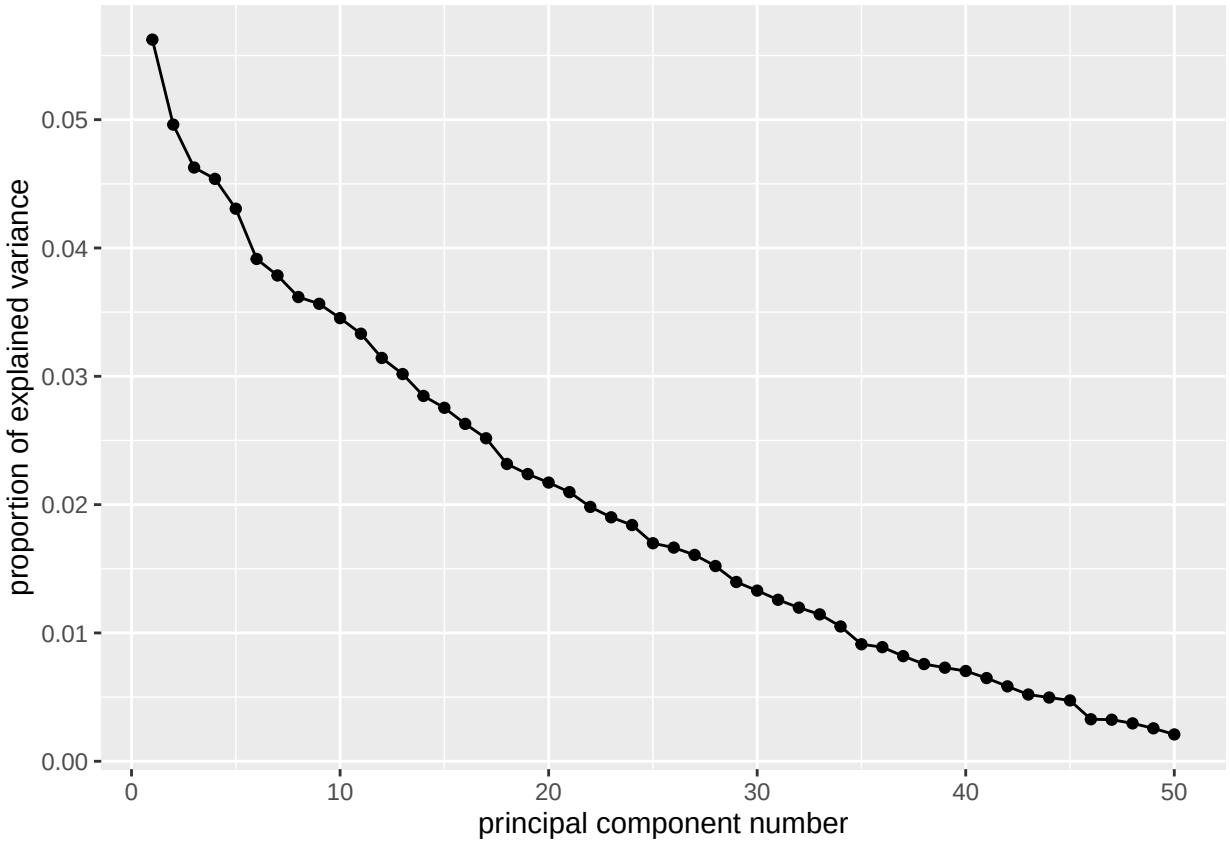
```
library(mvtnorm)
library(tinytex)
library(devtools)
```

```
## Loading required package: usethis
```

```
library(ggbiplot)
```

```
## Loading required package: ggplot2

## Loading required package: plyr

## Loading required package: scales

## Loading required package: grid
```

```r
# Set seeds for reproducibility
set.seed(301804639)
mu <- rexp(50,6)
multi <- rmvnorm(n = 100, mean = mu)

# Execute Principal Component Analysis (PCA)
pca_result <- prcomp(multi, scale = TRUE)

# Create a scree plot for all 50 PCA's components
ggscreeplot(pca_result, type = c("pev"))
```

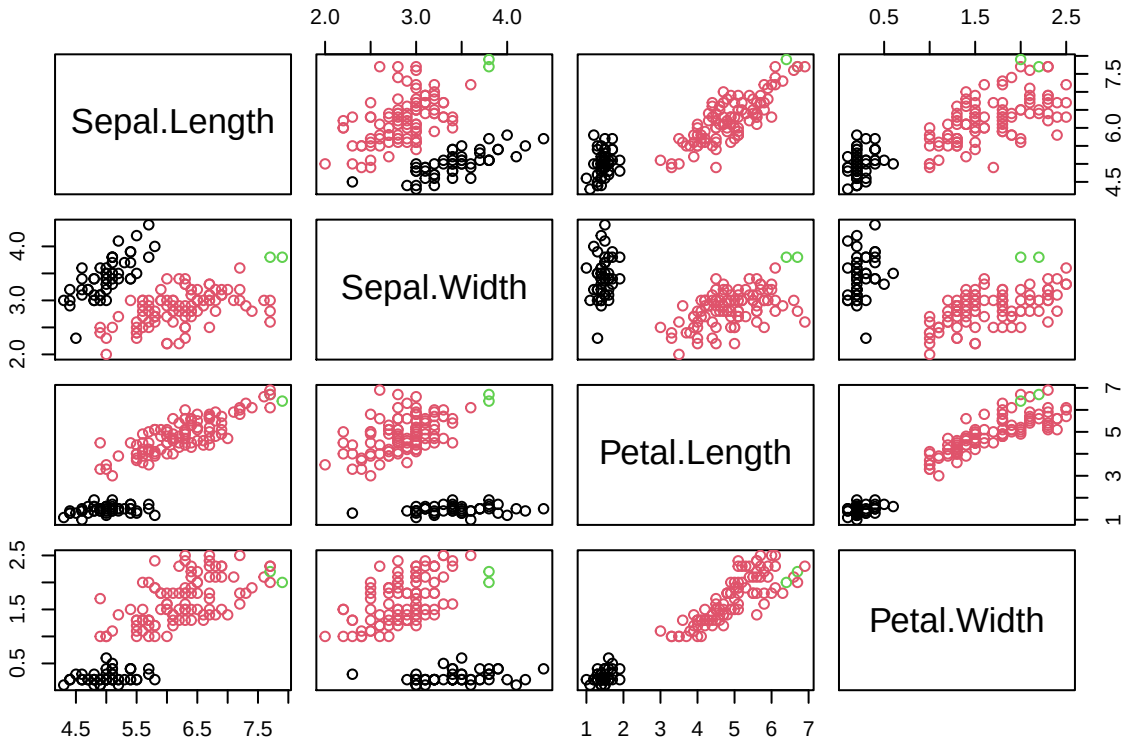


Question 15

Use the iris dataset (without the species variable), and prepare clusters using agglomerative clustering (single linkage). Plot the clusters using colors, in 6 scatter plots: one for each pair of variables.

```r
# Slice ouut Specias column from iris data frame
sliced_iris <- subset(iris, select = -c(Species))

# Compute distances matrix
iris.disimilarity <- dist(sliced_iris)

# Execute agglomerative clustering using single-linkage
HCA_result <- hclust(iris.disimilarity, method='single')
```

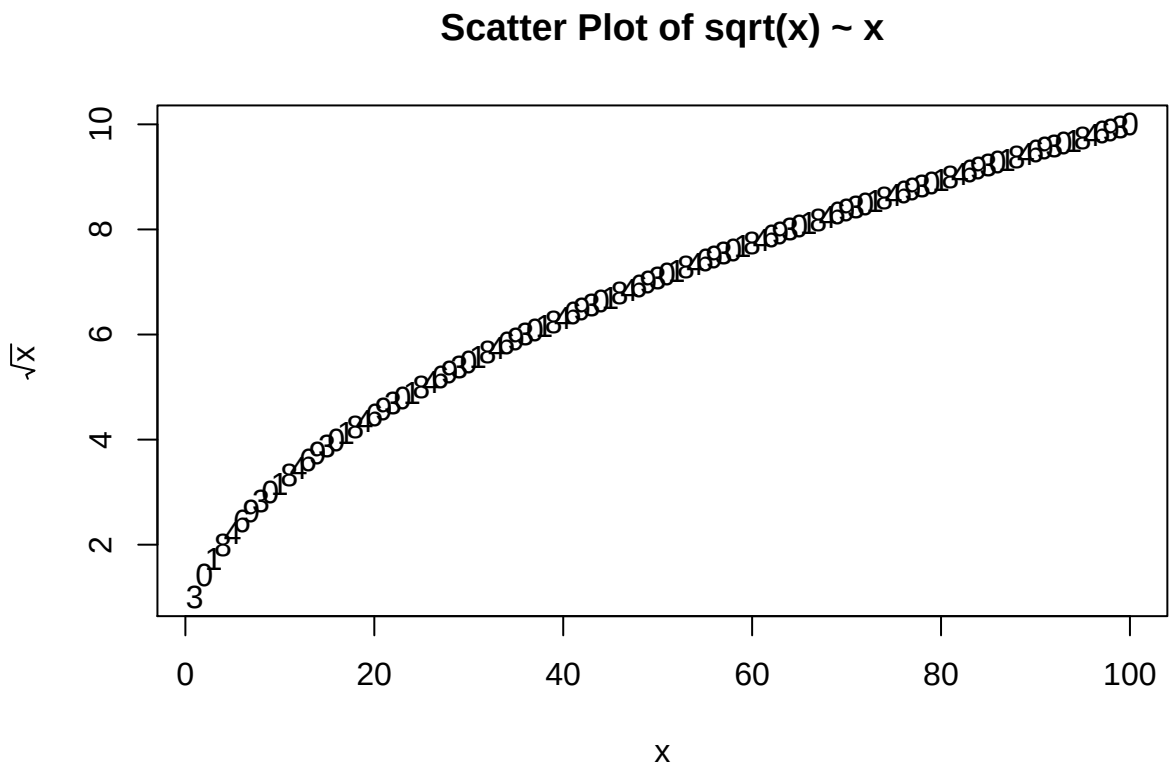
```r
# Get the class assignments given 3 clusters (i.e., setosa, versicolor & virginica)
HCA_result_cut <- cutree(HCA_result, k=3)

# Plot clusters using colors, in 6 scatter plots (i.e., one for each pair of variables)
pairs(sliced_iris, col=HCA_result_cut)
```



Question 16

Using the graphics package. Make a scatter plot where the x-axis is the numbers 1:100, y is sqrt(x), and the dot is a digit in your id. Recycle digits as necessary.

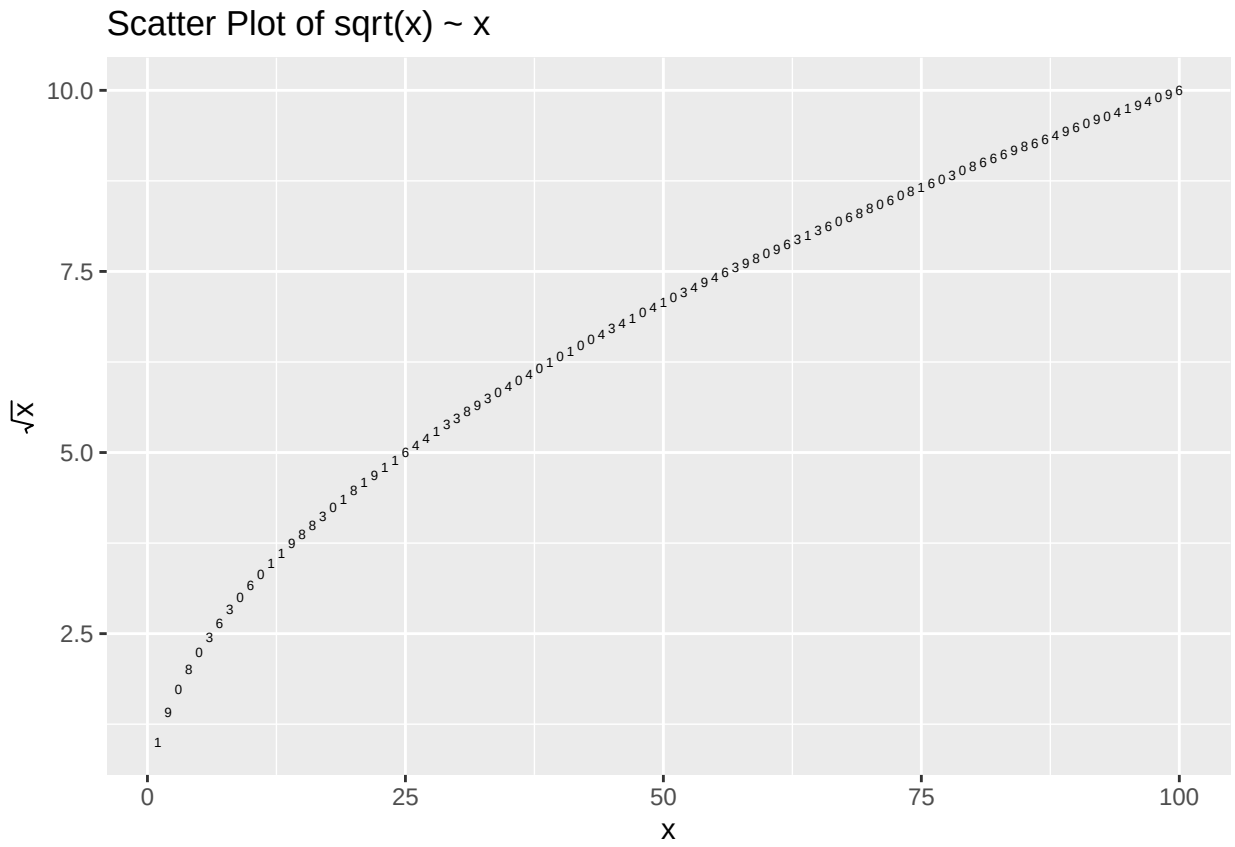
```r
# Create x-axis where the numbers are between [1, 100]
x <- seq(1,100,1)

# Create y-axis which equal to square root of x (i.e., values between [1, 10])
y <- sqrt(x)

# Make a scatter plot based on x & y axes and the dot as a digit in my id
plot(y~x, pch=c('3','0','1','8','4','6','9'), ylab=expression(sqrt(x)),
     main="Scatter Plot of sqrt(x) ~ x")
```



Question 17

Repeat the visualization in (16) using the ggplot2 package.

```
# Import ggplot2 package
library(ggplot2)

# Create x-axis where the numbers are between [1, 100]
x <- seq(1,100,1)

# Create y-axis which equal to square root of x (i.e., values between [1, 10])
y <- sqrt(x)

# Sample 100 numbers taken from my personal ID (301804639)
dots <- as.character(sample(c(3,0,1,8,4,6,9), length(x), replace = TRUE))

# Create data frame object based on the combination of both axes: x & y
df <- as.data.frame(cbind(x,y))

# Make a scatter plot based on x & y axes and the dot as a digit in my id
ggplot(aes(x,y), data = df) + ggtitle("Scatter Plot of sqrt(x) ~ x") +
  ylab(expression(sqrt(x)))+ geom_point(shape = dots)
```



Question 18

Generate 1e5 random letters in English. Make a dummy variable for each letter, and store all dummies in a sparse matrix. How much memory does this object consume?

The sparse matrix object consumes 7.604832 MB of memory.

```
# Import relevant packages
library('magrittr')
library('Matrix')
library('pryr')

# Generate 100,000 random letters in English and convert to factor
random_letters_vec <- letters %>% sample(1e5, replace=TRUE) %>% factor

# Stores all dummies in a sparse matrix
sparse_matrix <- sparse.model.matrix(~random_letters_vec-1)

# Calculate sparse object memory size
sparse_mat_obj_size <- pryr::object_size(sparse_matrix)/1e6

# Print sparse object memory size message
cat(paste("The sparse matrix object consumes",sparse_mat_obj_size ,
          "MB of memory.", sep = " ", collapse = "\n"))
```

```
## The sparse matrix object consumes 7.604832 MB of memory.
```

Question 19

Name 3 differences between open source R and Microsoft's R.

According to [4], **Open source R**, as the name suggest, is open source and can be downloaded for free from the Comprehensive R Archive Network (CRAN). **Open source R**, also known as **Base-R**, compiles and runs on a wide variety of UNIX platforms and similar systems such as (e.g., FreeBSD and Linux), Windows and MacOS and it's limited by the local computer memory. However, instead of downloading Base-R, one might download **Microsoft R Open** from Microsoft R Application Network (MRAN), an enhanced distribution of open source R, which is also free and open source, and extends open source R with an additional set of specialized packages released by Microsoft corporation and a version control mechanism [5].

According to [5], there are several differences between **open source R** and **Microsoft R Open**:

1. **Multithreaded Performance:** in **Microsoft R Open** multi-threaded math libraries are used to improve the performance of R (i.e., utilize multiple CPUs cores), unlike **open source R** that was designed to use only a single thread (i.e., CPU) at a time, unless linked with multi-threaded BLAS/LAPACK libraries. Also, performance benchmark for operations such as matrix multiplication, SVD, PCA, etc. can be found in [6].

2. **Number of Packages:** in **open source R** many packages can be installed include all base packages and more that can be downloaded from the CRAN's repository. In the case of **Microsoft R Open**, in addition to the last mentioned packages, a set of specialized packages released by Microsoft corporation can be installed to further enhance the **Microsoft R Open** experience.

3. **Reproducibility:** in the case of **open source R**, CRAN provides access to the latest versions of the vast majority of R packages. Although, it may seems like an advantage,the ever-changing packages present significant challenges. For instance, a package used yesterday may have been updated overnight and the script no longer works as expected. In order to address this issue, **Microsoft R Open** offers a reproducibility-based solution, in which during its installation, the CRAN repository is configured to point to a specific CRAN repository snapshot using the *checkpoint* package. As a result, when using install.packages in **Microsoft R Open** packages by default will be retrieved as they were at midnight UTC on the date in which the snapshot was originally taken.

References

[1] How the R startup process works?.
[2] Hotelling's T-squared two-sample statistic.
[3] ChickWeight Dataset.
[4] Introduction to R.
[5] About Microsoft R Open: The Enhanced R Distribution.
[6] The Benefits of Multithreaded Performance with Microsoft R Open.