week4b_lab

Naga Soundari Balamurugan

10/18/2018

Collaborators: Jayashree Raman

Instructions:

1. Download the `week4b_lab.Rmd` file from Canvas. Open `week4b_lab.Rmd` in RStudio and supply your solutions to the assignment by editing `week4b_lab.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text. If you are using more than just a standard function that you found from another source, please credit the source in the comments.
4. Collaboration on labs is encouraged, but students must turn in an individual assignments. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF` or `Knit Word`, rename the R Markdown file to `YourLastName_YourFirstName_ps3.Rmd`, knit a PDF or DOC and submit both the PDF/DOC and the Rmd file on Canvas.

Setup:

In this lab you will need, at minimum, the following R packages.

```r
# Load standard libraries
library(tidyverse)
library(readxl)
```

In-class Walk-Thru

Problem 1: Seinfeld

First we will recreate the "Seinfeld" dataset that was used in lecture 4a. One way to create this dataframe is seen in the code chunk below.

(a) Your first task is to add comments to the code chunk. Describe what the code is accomplishing at each step by replacing "Describe here" with your own words.

```r
# First Step: Creates an empty data frame with 5 columns and 10 rows
Seinfeld <- data.frame(matrix(ncol = 5, nrow = 10))
# Second Step: Assign column names for the created data frame
colnames(Seinfeld) <- c("Test_Number", "Jerry", "Newman", "George", "Cosmo")
# Third Step: Assign values(1 to 10) to the column Test_Number
Seinfeld$Test_Number <- c(1:10)
```

```r
# Fourth Step: Assign values to all other columns using their column name
Seinfeld$Jerry <- c(5,5,5,5,5,5,5,5,5,5)
Seinfeld$Newman <- c(0,0,0,0,0,10,10,10,10,10)
Seinfeld$George <- c(4,4,4,9,4,4,4,4,9,4)
Seinfeld$Cosmo <- c(5,7,2,4,8,2,7,8,1,6)
```

(b) This dataframe does not meet Hadley Wickham's definition of tidy. Tidy the data so that each observation forms a row and each variable forms a column. (Hint, you will be creating a new column, name this column "Score".)

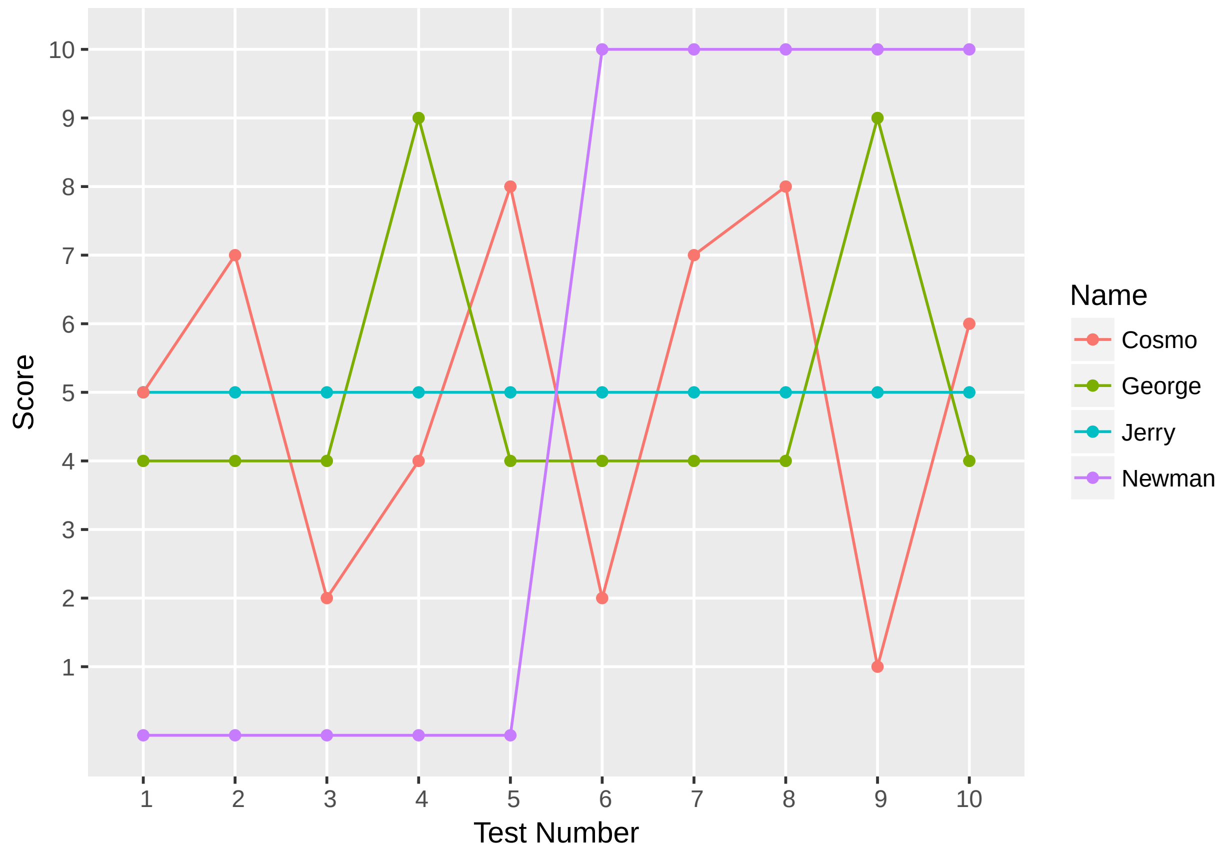
```r
#Tidying the data
Seinfeld <- gather(Seinfeld, Name, Score, -Test_Number)
Seinfeld
```

```
##    Test_Number   Name Score
## 1            1  Jerry     5
## 2            2  Jerry     5
## 3            3  Jerry     5
## 4            4  Jerry     5
## 5            5  Jerry     5
## 6            6  Jerry     5
## 7            7  Jerry     5
## 8            8  Jerry     5
## 9            9  Jerry     5
## 10          10  Jerry     5
## 11           1 Newman     0
## 12           2 Newman     0
## 13           3 Newman     0
## 14           4 Newman     0
## 15           5 Newman     0
## 16           6 Newman    10
## 17           7 Newman    10
## 18           8 Newman    10
## 19           9 Newman    10
## 20          10 Newman    10
## 21           1 George     4
## 22           2 George     4
## 23           3 George     4
## 24           4 George     9
## 25           5 George     4
## 26           6 George     4
## 27           7 George     4
## 28           8 George     4
## 29           9 George     9
## 30          10 George     4
## 31           1  Cosmo     5
## 32           2  Cosmo     7
## 33           3  Cosmo     2
## 34           4  Cosmo     4
## 35           5  Cosmo     8
## 36           6  Cosmo     2
## 37           7  Cosmo     7
## 38           8  Cosmo     8
## 39           9  Cosmo     1
## 40          10  Cosmo     6
```

```
#Seinfeld <- spread(Seinfeld, key = Test_Number, value = Score)
```

(c) Create a line plot that shows each student and their test scores over the 10 tests (Test_Number on the x-axis, Score on the y-axis) and use a different color for each student's name. (See lecture 4a slides for example.)

```
#Creating a line plot using ggplot function
ggplot(Seinfeld, aes(x = Test_Number, y = Score, group = Name)) +
  geom_line(aes(color = Name)) +
  geom_point(aes(color = Name)) +
  scale_x_discrete(name = "Test Number", limits = c(1:10)) +
  scale_y_discrete(name = "Score", limits = c(1:10))
```



Problem 2: T-tests

Load the built-in `sleep` dataset

> T-test is done to compare two groups with numerical values. It help us to know compare the sleep distribution among both groups

```
#Loading the sleep data
sleepData <- sleep
```

Suppose the two groups are independently sampled; well ignore the ID variable for the purposes here. The t.test function can operate on long-format (tidy) data like sleep, where one column (extra) records the measurement, and the other column (group) specifies the grouping; or it can operate on two separate vectors.

(a) Perform a t-test

Null hypothesis: No difference in sleeping pattern between groups, Alternate hypothesis: There is difference in sleeping pattern between groups

```
#T-test
t.test(extra ~ group, sleepData)
```

```
##
##  Welch Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

(b) You'll note that by default, the t.test does not assume equal variances; instead of Students t-test, it uses the Welch t-test by default. Note that in the Welch t-test, df=17.776, because of the adjustment for unequal variances. To use Students t-test, set var.equal=TRUE. Try a Student's t-test:

```
# Edit me
t.test(extra ~ group, sleepData, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 18, p-value = 0.07919
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.363874  0.203874
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

Since the p-value is 0.7(greater than 0.05), we cannot reject the null hypothesis. There is no difference in the means of group 1 and 2

Problem 3: Slot Machine

First we will re-create the dataset from lecture 4a regarding the randomness of a slot machine. Please add commenting.

```
# Step 1:
slotTest <- data.frame(matrix(ncol = 3, nrow = 6))
# Step 2:
colnames(slotTest) <- c("Fruit", "Count_Obs", "Count_Exp")
# Step 3:
slotTest$Fruit <- c("Banana", "Cherry", "Orange", "Peach", "Lemon", "Pear")
```

```r
# Step 4:
slotTest$Count_Obs <- c(13, 33, 14, 7, 36, 17)
slotTest$Count_Exp <- c(15, 28, 20, 5, 30, 22)
```

Look at the documentation for the chi-square test in R. Run the code `?chisq.test`. You can run a chi-square test with one input (called x in the help documentation) and the function assumes that the expected frequencies contain the same proportion of values (i.e. it will assume in our case that the expected frequency of each fruit is 6/120 or 20). Because our expected values are not a simple 1/6th proportion, we will have to specify a second parameter *p*.

Note: Since we have set pf expected values, we use mention the p parameter in the calculation. It should be in probability and hence we calculate it. Null hypothesis: The machine is not tampered with and the selection is random(No difference)

(a) In order to perform a one-sided chi-square in R, with defined expected values, we need to specify *p* (a vector of probabilities). So let's create a column that is each value in `Count_Exp` by the divided by the total of that column and let's call it `Prob_Exp`.

```r
# Edit me
total <- sum(slotTest$Count_Exp)
slotTest$Prob_Exp <- slotTest$Count_Exp/total
```

(b) Now run a chi-squared test, specifying x and p.

```r
# Edit me
chisq.test(slotTest$Count_Obs, p = slotTest$Prob_Exp)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  slotTest$Count_Obs
## X-squared = 6.0959, df = 5, p-value = 0.297
```

> As the p-value is greater than 0.05 we cannot reject the null hypothesis and say that the values are picked random.

Problem 4: Shift Types

Is there a relationship between turnover rates and shift worked among employees on a 24 hour Help Desk? We can perform a two-way chi-squared test.

First let's build the dataframe (we'll use a different method than above that achieves the same results).

```r
# Step 1:
Shift_Type = c("Day", "Swing", "Night")
# Step 2:
Stayed = c(148, 98, 128)
Resigned = c(12, 11, 29)
# Step 3:
shifts <- data.frame(Shift_Type, Stayed, Resigned)
```

(a) Now run a chi-squared test but be sure you're only passing in the numeric values from the dataframe.

> Null hypothesis: No difference between people working on different shifts

```r
# Edit me
chisq.test(shifts[,2:3])
```

```
##
##  Pearson's Chi-squared test
##
## data:  shifts[, 2:3]
## X-squared = 9.512, df = 2, p-value = 0.0086
```

> We reject the null hypothesis as the p-value is 0.008(greater than 0.05)

In-class Lab

Run `install.packages("HSAUR3")` in your console.

```r
# Load the HSAUR3 library
library(HSAUR3)
```

```
## Loading required package: tools
```

Problem 5: Pistonrings

(a) Load the `pistonrings` data.

```r
# Edit me
pistonrings_data <- pistonrings
pistonrings_data
```

```
##           leg
## compressor North Centre South
##         C1    17     17    12
##         C2    11      9    13
##         C3    11      8    19
##         C4    14      7    28
```

(a) The data are given in form of a table. The table gives the number of piston-ring failures in each of three legs of four steam-driven compressors located in the same building. The compressors have identical design and are oriented in the same way. The question of interest is whether there may be a relationship/association between the leg location variable and the compressors variable. State the null hypothesis and the alternative hypothesis:

> Null hypothesis: There is no relationship between leg location and compressor failures.

(b) Choose and state a significance level (p-value) threshhold:

> I have selected the significance level to be a 0.05.

(c) Perform a chi-square test of independence to on the data and describe whether you will or will not reject the null hypothesis.

```r
# Edit me
chisq.test(pistonrings)
```

```
##
##  Pearson's Chi-squared test
##
## data:  pistonrings
## X-squared = 11.722, df = 6, p-value = 0.06846
```

As the p-value is 0.06(>0.05 the threshold), we cannot reject the null hypothesis. Thus there is no relationship between leg location and compressor failures.

Problem 6: Linear Regression

If you still have time, start working through the example of linear regression found here: https://www.datacamp.com/community/tutorials/linear-regression-R. The dataset is available here: https://drive.google.com/drive/folders/1TcO4eVmpAn7gt4DbZ31HCz0t_gkca1Fp. Read through and perform the functions listed in the tutorial here in your .Rmd.

```r
#Read the files
ageAndHeightData <- read_excel("ageandheight.xls", sheet = "Hoja2")

#Run the linear regression
linearmodel_ageHeight <- lm(height ~ age, data = ageAndHeightData)
summary(linearmodel_ageHeight)
```

```
##
## Call:
## lm(formula = height ~ age, data = ageAndHeightData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.059   5.193   5.810   6.617   7.533
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  64.8445    43.3815   1.495    0.166
## age           0.3835     1.8264   0.210    0.838
##
## Residual standard error: 21.84 on 10 degrees of freedom
## Multiple R-squared:  0.00439,    Adjusted R-squared:  -0.09517
## F-statistic: 0.0441 on 1 and 10 DF,  p-value: 0.8379
```

```r
linearmodel_ageHeightSib <- lm(height ~ age + no_siblings, data = ageAndHeightData)
summary(linearmodel_ageHeightSib)
```

```
##
## Call:
## lm(formula = height ~ age + no_siblings, data = ageAndHeightData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.808  -4.521   1.002  10.025  21.186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   56.597     39.166   1.445   0.1823
## age            1.303      1.712   0.761   0.4660
## no_siblings   -6.415      3.464  -1.852   0.0971 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.59 on 9 degrees of freedom
## Multiple R-squared:  0.2791, Adjusted R-squared:  0.1189
## F-statistic: 1.742 on 2 and 9 DF,  p-value: 0.2293
```

Both the models are insignificant as the p-vales are higher and the R-squared values are lower. Comparatively, inclusing the no_sibling variable makes it better.