Problem Set 1
Basic Programming Concepts

Christoph Riedl

Business Analytics

---

1 Vectors and Computations [*Applied*]

Calculate the following sum by implementing it as a `for()` loop in `R`:

$$\sum_{i=10}^{99} (i^2 + i^3) \tag{1}$$

Is there another way of computing it that does not use a `for()` loop?

2 Flipping a Fair Coin (aka: Basic Random Numbers) [*Applied*]

- Use `R` to simulate a single fair coin toss. Your code should simply print "Heads" or "Tails" to the screen once every time you run it, each time with a probability $p = 1/2$ for a head or a tail.
- Using a for loop, simulate 100 tosses of a fair coin and count how many heads you get in a variable called total which you print out after the loop.
- *Optional/advanced* Now repeat without resorting to a for loop. Can you do it in just two lines of code?

3 Simulating Random Numbers [*Applied*]

A test is graded from 0 to 50, with an average score of 35 and a standard deviation of 10.

- Simulate data for a class of 25 students who took the test in `R`. Assume scores are drawn from a normal distribution. Use the `rnorm()` function to create random numbers drawn from a normal distribution – but make sure to adjust the mean and standard deviation accordingly. Use `?rnorm` to find out how.
- Do you note any issues with the approach using `rnorm` to generate the numbers?
- Bonus: Tests are not usually graded as full precision floting point numbers. Can you figure out how to simulate test scores that are rounded to the nearest `.5`?

4 Working with Character Vectors [*Applied*]

Use the function `paste()` to create the following character vectors of length 30:

1. `("label 1", "label 2", ....., "label 30")`.

Note that there is a single space between label and the number following.

2. `("fn1", "fn2", ..., "fn30")`.

In this case, there is no space between fn and the number following.

5 Random Numbers and Histograms [*Applied*]

Let $x = x_1 + ... + x_{20}$, the sum of 20 independent Uniform(0,1) random variables. In `R`, create 1,000 simulations of x and plot their histogram. On the histogram, overlay a graph of the normal density function with the same mean as x. Comment on any differences between the histogram and the curve.

Hint 1: To plot a histogram in `R` you can build on the following code:

```
library(ggplot2)
df <- data.frame( x = rnorm(1000) )
ggplot(df, aes(x)) + geom_histogram(aes(y=..density..))
```

Hint 2: In `ggplot` you cannot plot the normal density as a function. But you can easily simulate lots of values drawn from a normal distribution (say 1,000) and plot these data as a `geom_density()`. The main challenge for you to figure out is how to use two `ggplot` layers, each with its own `data.frame`: The first layer uses the data from x the second uses data from the normal distribution.

6 Understanding Vectorized Instructions and Quirkyness of R [*Applied*]

Execute the following line on the `R` console. What do you get? Can you explain, step by step, how `R` is interpreting the command?

```
1:10 > 5
```

7 Exploratory Data Analysis (EDA) [*Applied*]

A common business problem that we discussed in class is customer churn: existing customers that leave for the competition. You can download a dataset of churn data on Blackboard in the Datasets menu.

- Load the data set into `R`. You can use the below lines of code to help.
- Explore the general structure of the data: How many rows and columns does the data have?
- Create a new variable *IncomeGroup* that characterizes users with as `<$35k; $35k-$45k; $45k-$65k; $65k-$100k; $100k`. Use the function `cut()` to divide the data into intervals.
- Plot the distribution of $OVERAGE$ for each of the income groups.
- The dataset has a variable that captures whether a certain customer canceled his/her cellphone contract (variable $LEAVE$). Explore the data with regard to $LEAVE$ decisions and make at least *one* visual and quantitavie comparisons across income groups and other variables to find out who is most likely to leave.
- Create one or two metrics/measures/statistics that summarize the data. Examples of potentials metrics include min, max, mean, variance (standard deviation) and these can be calculated across various user segments. Be selective. Think about what might be most important to track.
- You should aim to include 2-3 plots or tables in your submission. Describe and interpret any patterns you find with a short paragraph.

Template code to help you load the *churn* dataset:

```
# Load the data and assign nicer variable names
churn <- read.csv("churn.arff", skip=15, header=FALSE)
names(churn) <- c("COLLEGE", "INCOME", "OVERAGE", "LEFTOVER", "HOUSE",
"HANDSET_PRICE", "OVER_15MINS_CALLS_PER_MONTH", "AVERAGE_CALL_DURATION",
"REPORTED_SATISFACTION", "REPORTED_USAGE_LEVEL",
"CONSIDERING_CHANGE_OF_PLAN", "LEAVE")
```