STOCKHOLM UNIVERSITY
Department of Statistics
Ellinor Fackle Fornius

EXAM IN R PROGRAMMING
October 27, 2022

Time: 8.00-13.00

Results will be announced no later than November 11.

At the start of the exam, go to the web page http://tenta.stat.su.se/tenta.html and select the option 'Download SEB-config File for R programming'. Run the configuration file to start the Safe Examination Browser (SEB). At the homepage in SEB you find the option to download your supporting file. Fill out your name and your anonymous code to get access to your file.

The exam should be written as a R Markdown report. You should submit the R markdown file (.Rmd) as well as an output file in .html format. You are free to use either R Base packages or Tidyverse packages to solve the tasks, unless stated otherwise in the instructions.

Notify the computer exam staff when you are ready to hand in your exam. Before you submit your exam, check that you have NOT written your name anywhere in your files (which would break the anonymity of the exam). Upload your exam files at the homepage in SEB.

Note: Don't forget to save often! In the event of R crashing, any unsaved changes could be lost.

Task 1. The following object `p` is a list with information about two persons; Ann and Ben.

```
p <- list("Ann" = list(location = "Stockholm",
                       kids = NULL,
                       job = "Programmer",
                       salary = 120000),
          "Ben" = list(location = "London",
                       kids = c("Lisa", "Paul")))
```

a) Use two sub-setting methods to get Ben's location.

b) Try typing `p["Ben"]$location`. Explain the result.

c) How many kids does Ben have (use code to get the answer)?

d) Ben got a newborn baby called Malena. Add her name to the kids vector.

e) Add a third person called Catherine. She is located in Sydney, has one child called Lea and works as a teacher.

f) Transform the list `p` to a vector which contains all the components in `p`.

g) Explain the concept coercion. Comment on whether it was needed in (f) or not.

Task 2. Create a function `skewness(x)`, that calculates the skewness of a sample in the vector `x` as

$$\texttt{skewness(x)} = \frac{\frac{1}{n}\sum_{i=1}^{n}(\texttt{x}_i - \bar{\texttt{x}})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(\texttt{x}_i - \bar{\texttt{x}})^2\right)^{3/2}}$$

If `x` is not numeric, the function should return an error message, see the test cases below. Note that it is not allowed to use any R built-in functions for calculating skewness to solve the task.

```
test1 <- skewness(x = c(12, 42, 2, 3, 12, 100))
test1

[1] 1.3006

skewness(x = "hi")

Error in skewness(x = "hi"): Not numeric!

data("trees")
skewness(x = trees[ ,1])

[1] 0.52632

skewness(x = trees[ ,2])

[1] -0.37487
```

Task 3. Create a function `poly(intercept, coef, x)` that computes the polynomial

$$\texttt{y} = \texttt{intercept} + \sum_{i=1}^{N} \texttt{coef}_i \; \texttt{x}^i$$

The function argument `coef` is a numeric vector of length N, `x` is a numeric vector of arbitrary length and `intercept` is a numerical variable. Note that the vectors `x` and `coef` does not have to be the same length and that `y` will be of the same length as `x`.

For example, given the arguments `intercept = 10` and `coef = (2, -0.5, 3)` the resulting polynomial is

$$\texttt{y} = 10 + 2\texttt{x} - 0.5\texttt{x}^2 + 3\texttt{x}^3$$

The function should return a list with `y`, `intercept`, `coef` and `x` as in the testcases below.

```
x1 <- -5:5

res1 <- poly(intercept = 3, coef = c(1, -2), x = x1)
res1

$y
 [1] -52 -33 -18  -7   0   3   2  -3 -12 -25 -42

$intercept
[1] 3

$coef
[1]  1 -2

$x
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5

names(res1)

[1] "y"         "intercept" "coef"      "x"

res2 <- poly(intercept = 3, coef = c(2, 0.1, -0.3), x = x1)
res2$y

 [1]  33.0  15.8   6.0   1.8   1.4   3.0   4.8   5.0   1.8  -6.6 -22.0

res3 <- poly(intercept = 3, coef = 5:1, x = c(5, -2, pi))
res3$y

[1] 4878.00  -15.00  652.04
```

Task 4. This data set consists of weight and length measurements taken on 8 babies at different timepoints; when they were 3, 6 and 9 months old.

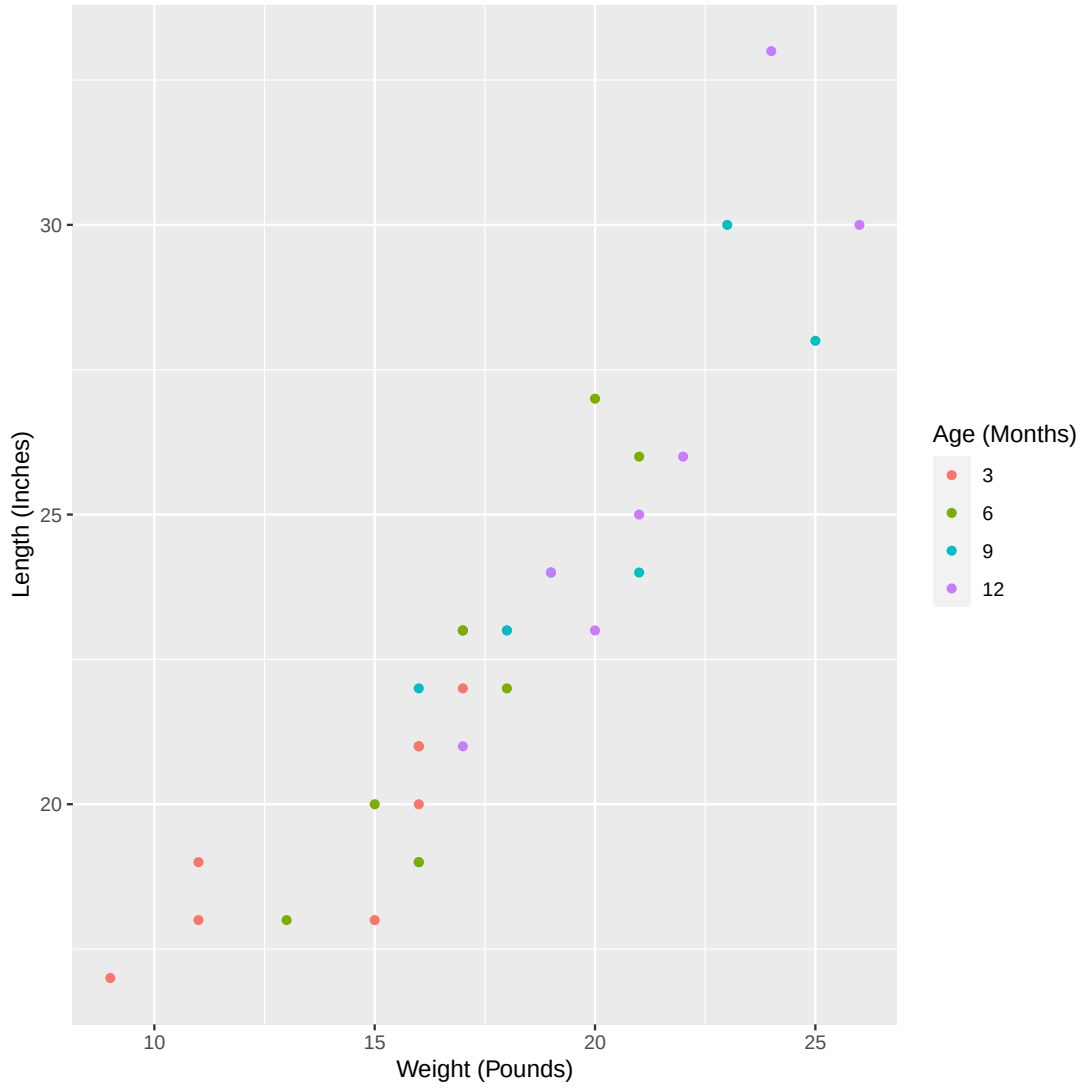
```
set.seed(123)
babies <- tibble(
  id       = 1001:1008,
  sex      = c("F", "F", "M", "F", "M", "M", "M", "F"),
  weight_3  = c(9, 11, 17, 16, 11, 17, 16, 15),
  weight_6  = c(13, 16, 20, 18, 15, 21, 17, 16),
  weight_9  = c(16, 17, 23, 21, 16, 25, 19, 18),
  weight_12 = c(17, 20, 24, 22, 18, 26, 21, 19),
  length_3  = c(17, 19, 23, 20, 18, 22, 21, 18),
  length_6  = round(length_3 + rnorm(8, 2, 1)),
  length_9  = round(length_6 + rnorm(8, 2, 1)),
  length_12 = round(length_9 + rnorm(8, 2, 1)),
)
babies


# A tibble: 8 x 10
     id sex   weight_3 weight_6 weight_9 weight_12 length_3 length_6 length_9
  <int> <chr>    <dbl>    <dbl>    <dbl>     <dbl>    <dbl>    <dbl>    <dbl>
1  1001 F            9       13       16        17       17       18       19
2  1002 F           11       16       17        20       19       21       23
3  1003 M           17       20       23        24       23       27       30
4  1004 F           16       18       21        22       20       22       24
5  1005 M           11       15       16        18       18       20       22
6  1006 M           17       21       25        26       22       26       28
7  1007 M           16       17       19        21       21       23       24
8  1008 F           15       16       18        19       18       19       23
# ... with 1 more variable: length_12 <dbl>
```

a) Transform the data into the following tidy structure.

```
# A tibble: 32 x 5
      id sex   months weight length
   <int> <chr> <chr>   <dbl>  <dbl>
 1  1001 F     3           9     17
 2  1001 F     6          13     18
 3  1001 F     9          16     19
 4  1001 F     12         17     21
 5  1002 F     3          11     19
 6  1002 F     6          16     21
 7  1002 F     9          17     23
 8  1002 F     12         20     23
 9  1003 M     3          17     23
10  1003 M     6          20     27
# ... with 22 more rows
```

b) Restructure the data back to the original format.

c) Reproduce the following graph.



Task 5. We have a character vector which contains a number of strings, where some of the elements includes a phone number. The phone numbers share a common structure although being stored in different formats.

```
strings <- c(" 219 733 8965", "329-293-8753 ", "banana", "595 794 7569",
             "387 287 6718", "apple", "233.398.9187  ", "482 952 3315", "239 923 8115",
             "842 566 4692", "Work: 579-499-7527", "$1000", "Home: 543.355.3679")

strings
```

```
 [1] " 219 733 8965"      "329-293-8753 "      "banana"
 [4] "595 794 7569"       "387 287 6718"       "apple"
```

```
 [7] "233.398.9187  "     "482 952 3315"       "239 923 8115"
[10] "842 566 4692"       "Work: 579-499-7527" "$1000"
[13] "Home: 543.355.3679"
```

(a) The phone numbers have the following pattern structure
   - one digit between 2-9
   - two digits between 0-9
   - a separator that can be either a whitespace, hyphen or dot
   - three digits between 0-9
   - a separator that can be either a whitespace, hyphen or dot
   - four digits between 0-9

   Write this pattern as a *Regular Expression*.

(b) Use the pattern defined as a regular expression in (a) to extract the strings containing a phone number. You should get the following result.

```
 [1] " 219 733 8965"      "329-293-8753 "      "595 794 7569"
 [4] "387 287 6718"       "233.398.9187  "     "482 952 3315"
 [7] "239 923 8115"       "842 566 4692"       "Work: 579-499-7527"
[10] "Home: 543.355.3679"
```

(c) Now extract just the phone numbers. This is the expected result.

```
 [1] " 219 733 8965"      "329-293-8753 "      "595 794 7569"
 [4] "387 287 6718"       "233.398.9187  "     "482 952 3315"
 [7] "239 923 8115"       "842 566 4692"       "Work: 579-499-7527"
[10] "Home: 543.355.3679"
```

(d) Anonymise the pone numbers in the data by replacing them with `XXX-XXX-XXXX`.

Task 6. When combining two factor variables using the `c()` function some information is lost because R only combines the two underlying integer vectors, as in the following example.

```
degree  <- factor(c("master", "bachelor", "bachelor"))
degree
```

```
[1] master   bachelor bachelor
Levels: bachelor master
```

```
degree2 <- factor(c("phd", "bachelor", "bachelor", "phd"))
degree2
```

```
[1] phd      bachelor bachelor phd
Levels: bachelor phd
```

```
c(degree, degree2)
```

```
[1] 2 1 1 2 1 1 2
```

Define a new method for the generic function `c()` for objects of class `factor`. The new method should take two factor variables as input arguments and create a combined factor variable that preserves the factor levels. Test `c()` with the two factor variables given above ands check that you get the following output.

```
c(degree, degree2)

[1] master   bachelor bachelor phd      bachelor bachelor phd
Levels: bachelor master phd
```

Task 7. Give a brief description of a couple of good coding practices, e.g. as described in the article "Best Practices for Scientific Computing".

Good Luck!