2022 PIPS Exam

2022 Programming in Psychological Science Exam

Exam instructions

Place your ID clearly visible on your right side. Read these exam instructions carefully.

This exam must be completed individually. You **are** allowed to use your notes, R, Python, and websites **that do not have interactive chat features**. You can only use your laptops, pens, and paper. Cellphone usage is not allowed. **Watching videos (e.g. YouTube) is not allowed.** Your computer speakers must be muted and no headphones are allowed. **You cannot communicate with anyone in any form during the exam** (e.g. no Slack, posting on Stack Overflow, morse code, etc.). If you do so, you will automatically fail this exam.

Pick any 12 of the following 20 problems to submit on Canvas. You can attempt more than 12 problems, but **you may only submit 12 answers**. Note for the first 16 questions you must use **R** and for the last 4 questions you must use **Python**. You may only use the R libraries and Python modules specified. Some questions have more specific rules about what libraries/modules to use. R questions and Python questions are labeled with a "R" and "P" respectively.

You must include the name of each problem (e.g "E.R2") in the Canvas submission before your answer. Your answer can be copied and pasted from a R or Python script. But you should **not** submit an entire .R or .py script. **You must also return these 2 front-back printed exam papers with your name and student ID.** Please enter the provided exam number on Canvas. Submit the Canvas Exam and return these papers by 16:00.

On request of the examiner (or his/her representative) you should be able to identify yourself with a valid ID card. If you have any technical problems or questions about the exam, raise your hand.

Write your name:

Write your student ID:

Place this exam number on Canvas:

```
## [1] 685656827
```

You may only use these libraries for the R questions starting on the next page:

```
library(titanic)
library(ggplot2)
library(dplyr)
```

E.R1

What are the names of the first two arguments used by the function median()?

E.R2

Define a function named `is_tensor()` that takes one argument, this function returns true if the input array has more than 2 dimensions and false if the input array has 2 dimensions or less. The function should also produce an error that says "`Input must be an array`" if the input is not an array. You may only use base R functions.

E.R3

Using the `titanic_train` data set from the library `titanic`, create a function `was_there_a` that takes one argument `search_term`. This function should return a logical value indicating whether there was a male passenger on the Titanic with the `search_term` being part of their name. You may only use base R functions.

E.R4

Create a array of size 132 by 5 by 360 with random draws from the standard normal distribution in every element using `rnorm()` with default arguments. Now index the 101st to 343rd element of the last dimension of your array to return 243 matrices within a 3 dimensional array. Do everything without using any loops and only base R functions.

E.R5

Load the `titanic_train` data set from the `titanic` package into your workspace. What is the percentage of the passengers with a *known* age was under 30 years old?

E.R6

In terms of **R style** what are two things wrong with the following code? Rewrite the code to fix the problems you identified with the **R style**. You may not use lintr, nor Styler in R Studio.

```r
# This line samples 1000 random normal variables and then adds that to 1000 beta
# variables with shape1 = 1 and shape2 = 2 parameters. Then it takes the
# absolute value of that sum. Then it takes the square root of the sum. Then it
# takes the mean of that sum. Then assigns it to the variable "mean_data"
mean_data <- mean(sqrt(abs(rnorm(1000) + rbeta(1000, shape1 = 1, shape2 = 2))))
```

E.R7

Using the built-in `ChickWeight` data set, display the weights of the chickens at time point 21 in a violin plot. Note that you may (but do not have to) use a function from the `ggplot2` library. How many chickens do not have a value for this time point?

E.R8

Create the following `data.frame` without typing out the full vector of any of the columns.

```
##   year people government
## 1  -60  Roman   republic
## 2  -40  Roman   republic
## 3  -20  Roman     empire
## 4    0  Roman     empire
```

E.R9

Load the `titanic_train` data set from the `titanic` package into your workspace. Add a new logical variable called `leo` to the data frame. This variable has the value `TRUE` when the person was in third class and did not survive. For all other cases it has the value `FALSE`. Print how many passengers have a `TRUE` value the `leo` column. You may use base R and/or `dplyr` functions.

E.R10

Only using base R functions, make the following vector without writing all the texts/numbers: `c("Student1", "Student2", "Student3", "Student4", "Student5")` Then use a base R function to change the substring `"Student"` to `"Master"` for each element in the vector.

E.R11

```
hist(sqrt(abs(rnorm(1000))))
```

Convert this R code into something that uses at least 4 pipes (either `dplyr` pipes or base R pipes).

E.R12

Your friend is writing code to calculate the amount of times throwing 3 dice will contain only 5s. Your friend is using 1000 simulated dice rolls to do so. However their code always returns 0 rolls as 5-5-5.

What is wrong with their code below? Can you fix their code to better simulate the number of 5-5-5 throws? You should be able to share the code with your friend and get the exact same answer.

```
num_rolls <- 1000
all_five <- 0
for (index in 1:num_rolls) {
  set.seed(1234)
  dice_roll <- sample(1:6, 3, replace = TRUE)
  if (all(dice_roll == 5)) {
    all_five <- all_five + 1
  }
}
```

E.R13

Load the `titanic_train` data set from the `titanic` package into your workspace. Using `dplyr` functions (also known as `dplyr` "verbs"), return a data frame with only 3rd class passengers that lists passenger fares in descending order.

E.R14

```
set.seed(11)
x = rnorm(20)
r = rnorm(20)
y = 1 + x + r
```

Make a scatter plot **only using base R** that plots variable `x` on the x-axis and `y` on the y-axis. Label your x-axis "Independent variable" and your y-axis "Dependent variable". **`ggplot2` functions are not allowed for this question**.

E.R15

Create a while loop in which you draw 100 new random normal distribution draws (using the standard normal distributions, e.g. default parameters) until the minimum is less than -5. Print `Positive outlier found!` if the maximum is greater than 5 in the loop. Use only base R functions. You must use an "explicit loop".

E.R16

In terms of **R style** what are two things wrong with the following code? Rewrite the code to fix the problems you identified with the **R style**. You may not use lintr, nor Styler in R Studio.

```
set.seed(11)
x = rnorm(20)
r = rnorm(20)
y = 1 + x + r
```

You may only use these modules for the following Python questions:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

E.P1

Define a **Python function** that takes two vectors as inputs. The **function** then plots a scatter plot of the first input on the x-axis and the second input on the y-axis. Use the `matplotlib` library to make your plots.

E.P2

Using **Python**, make the following `pandas.DataFrame` without typing out the full vector of any of the columns.

```
##    year people government
## 0   -60  Roman   republic
## 1   -40  Roman   republic
## 2   -20  Roman     empire
## 3     0  Roman     empire
```

E.P3

Write code that reads the csv file at this location using `pandas` in **Python**:

https://tinyurl.com/pipsTitanic

Write a for loop over the rows (passengers) of the data file's original order. Whenever the current passenger is female and survived, **and also** the previous passenger was male, print: "`I'll never let go, Jack. I'll never let go. I promise.`"

E.P4

Write a function in **Python** with one argument `start_num`. This function counts up by 0.1 when given a number `start_num` greater than 0, and counts down by one whole number when given a number `start_num` less than 0. The function should stop counting when it reaches 100 or -100. The function should also print "`I'm not good at counting backwards`" before it counts backwards, but not when it counts forwards.