✖

Week 4 Quiz

✓ 10/10 questions correct

Quiz passed!

Continue Course (/learn/r-programming/supplement/IVvRu/practical-r-exercises-in-swirl-part-4)

Back to Week 4 (/learn/r-programming/home/week/4)

✓ 1.

What is produced at the end of this snippet of R code?

```
set.seed(1)
rpois(5, 2)
```

○ It is impossible to tell because the result is random

○ A vector with the numbers 1, 1, 2, 4, 1

Well done!
Because the `set.seed()' function is used, `rpois()' will always output the same vector in this code.

○ A vector with the numbers 1, 4, 1, 1, 5

○ A vector with the numbers 3.3, 2.5, 0.5, 1.1, 1.7

✓ 2.

What R function can be used to generate standard Normal random variables?

○ rnorm

Well done!
Functions beginning with the `r' prefix are used to simulate random variates.

○ pnorm

○ dnorm

○ qnorm

✓ 3.

When simulating data, why is using the set.seed() function important? Select all that apply.

□ It ensures that the random numbers generated are within specified boundaries.

Well done!

□ It can be used to specify which random number generating algorithm R should use, ensuring consistency and reproducibility.

Well done!

□ It ensures that the sequence of random numbers is truly random.

Well done!

□ It can be used to generate non-uniform random numbers.

Well done!

✓ 4.

Which function can be used to evaluate the inverse cumulative distribution function for the Poisson distribution?

○ dpois

○ rpois

○ ppois

○ qpois

> Well done!
> Probability distribution functions beginning with the `q' prefix are used to evaluate the quantile (inverse cumulative distribution) function.

✓ 5.

What does the following code do?

```
set.seed(10)
x <- rep(0:1, each = 5)
e <- rnorm(10, 0, 20)
y <- 0.5 + 2 * x + e
```

○ Generate data from a Normal linear model

> Well done!

○ Generate uniformly distributed random data

○ Generate data from a Poisson generalized linear model

○ Generate random exponentially distributed data

✓ 6.

What R function can be used to generate Binomial random variables?

○ qbinom

○ rbinom

Well done!

○ dbinom

○ pbinom

✓ 7.

What aspect of the R runtime does the profiler keep track of when an R expression is evaluated?

○ the function call stack

Well done!

○ the package search list

○ the working directory

○ the global environment

✓ 8.

Consider the following R code

```
library(datasets)
Rprof()
fit <- lm(y ~ x1 + x2)
Rprof(NULL)
```

(Assume that y, x1, and x2 are present in the workspace.) Without running the code, what percentage of the run time is spent in the 'lm' function, based on the 'by.total' method of normalization shown in 'summaryRprof()'?

○ It is not possible to tell

○ 23%

○ 50%

○ 100%

Well done!

When using `by.total' normalization, the top-level function (in this case, `lm()') always takes 100% of the time.

✓ 9.

When using 'system.time()', what is the user time?

○ It is the "wall-clock" time it takes to evaluate an expression

○ It is the time spent by the CPU waiting for other tasks to finish

○ It is a measure of network latency

○ It is the time spent by the CPU evaluating an expression

Well done!

✓ 10.

If a computer has more than one available processor and R is able to take advantage of that, then which of the following is true when using 'system.time()'?

○ user time is always smaller than elapsed time

○ elapsed time is 0

○ user time is 0

○ elapsed time may be smaller than user time

Well done!