

2017-02-13

*Rick O. Gilmore*

*2017-02-12*

## Contents

Goals . . . . .	1
Advanced simulation . . . . .	1
Let's make it a function . . . . .	1
Doing a series of simulations . . . . .	3
How about a factorial design? . . . . .	5
Put this in a function . . . . .	6
Create code to a set of participants and create one merged data frame . . . . .	7
Now, want to see what we have? . . . . .	7
And, just for fun . . . . .	8
Or . . . . .	9
Practice with visualization . . . . .	9
Learning by example . . . . .	9
A syntax I like . . . . .	9
Assignment (due Monday, February 20) . . . . .	10

## Goals

- Advanced simulation
- Practice with visualization

## Advanced simulation

- Putting code into a function
- A factorial design?

## Let's make it a function

- Functions help automate routines
- Parts of a function:
  - Input parameters
    - \* Defaults or not
  - Output(s)
- `My.function.name <- function(my.param1, my.param2 = "cool")`

---

```
# define global constants from prior simulation
sample.n = 200
beta0 = 36
beta1 = .33
sigma = 10
min.x = 80
max.x = 250
```

---

```

Height.weight.sim <- function(sample.n = 200, beta0 = 36, beta1 = .33, sigma = 10, min.x = 80, max.x = 180) {
  # Calculates correlation, intercept, slope estimates for
  # linear relation between two variables

  # Args:
  #   sample.n: Number of sample points, default is 200
  #   beta0: Intercept, default is 36 (inches)
  #   beta1: Slope, default is .33
  #   sigma: Standard deviation of error
  #   min.x: Minimum value for x (weight in lbs)
  #   max.x: Maximum value for x (weight in lbs)
  #
  # Returns:
  #   Named array with values
  #   beta0
  #   beta1
  #   beta0.lo: 2.5% quantile for intercept
  #   beta0.hi 97.5% quantile for intercept
  #   beta1.lo 2.5% quantile for slope
  #   beta1.hi 97.5% quantile for slope

  w <- runif(n = sample.n, min = min.x, max = max.x)

  h.pred <- rep(x = beta0, n = sample.n) + beta1 * w
  h <- h.pred + rnorm(n = sample.n, mean = 0, sd = sigma)
  height.weight <- data.frame(inches = h, lbs = w)

  fit <- lm(formula = inches ~ lbs, data = height.weight)
  ci <- confint(fit)

  # Create output vector with named values
  (results <- c("beta0" = beta0,
    "beta1" = beta1,
    "beta0.lo" = ci[1,1],
    "beta0.hi" = ci[1,2],
    "beta1.lo" = ci[2,1],
    "beta1.hi" = ci[2,2]))
}

```

---

*# Defaults only*

```
Height.weight.sim()
```

```
##      beta0      beta1  beta0.lo  beta0.hi  beta1.lo  beta1.hi
## 36.0000000  0.3300000 23.0440300 33.2490909  0.3404709  0.4004694
```

*# Larger sample size*

```
Height.weight.sim(sample.n = 500)
```

```
##      beta0      beta1  beta0.lo  beta0.hi  beta1.lo  beta1.hi
## 36.0000000  0.3300000 35.6895453 41.9194455  0.3030368  0.3394303
```

## Doing a series of simulations

- Goal: run our function a number of times, collect the results

```
n.simulations = 100
n.vars = 6 # variables Height.weight.sim() outputs

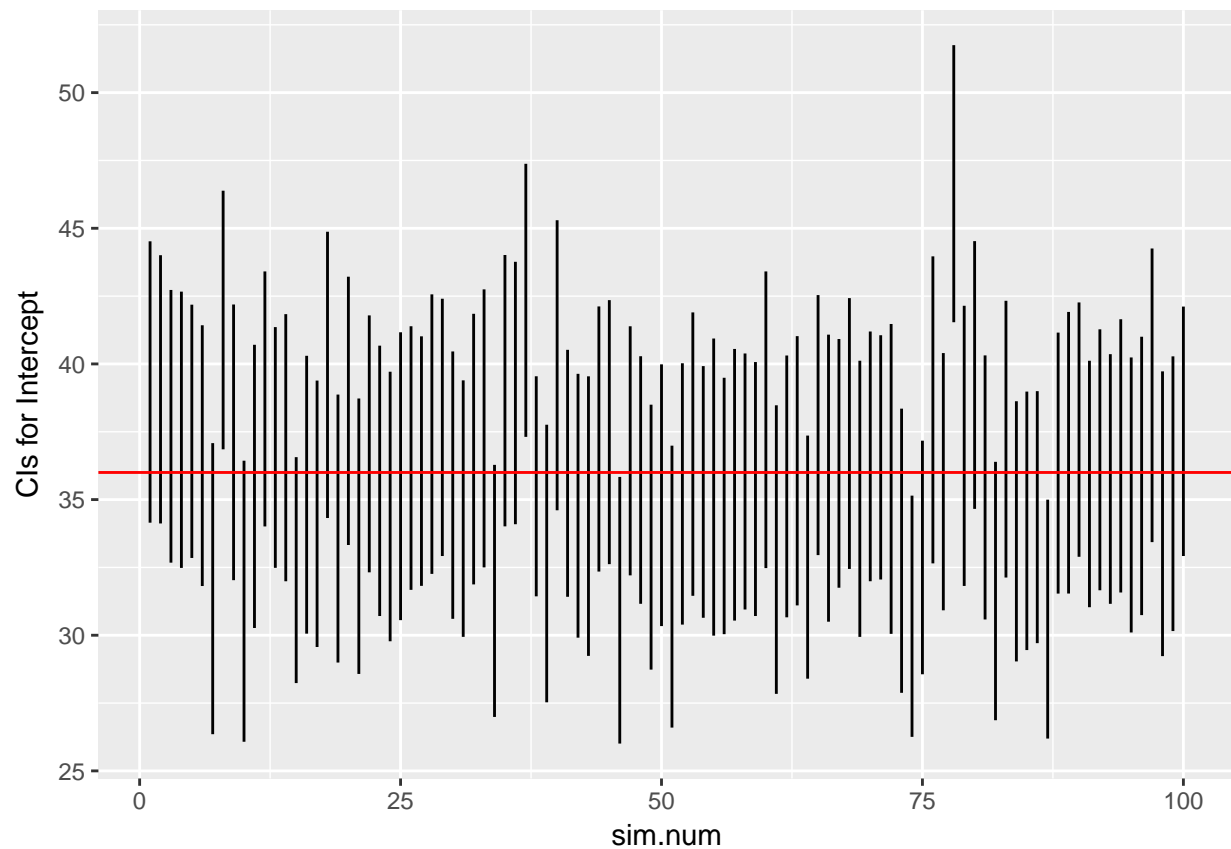
# initialize output array
height.weight.sim.data <- array(0, dim=c(n.simulations, n.vars))

# Repeat Height.weight.sim() n.simulations times
for (i in 1:n.simulations) {
  height.weight.sim.data[i,] <- Height.weight.sim()
}

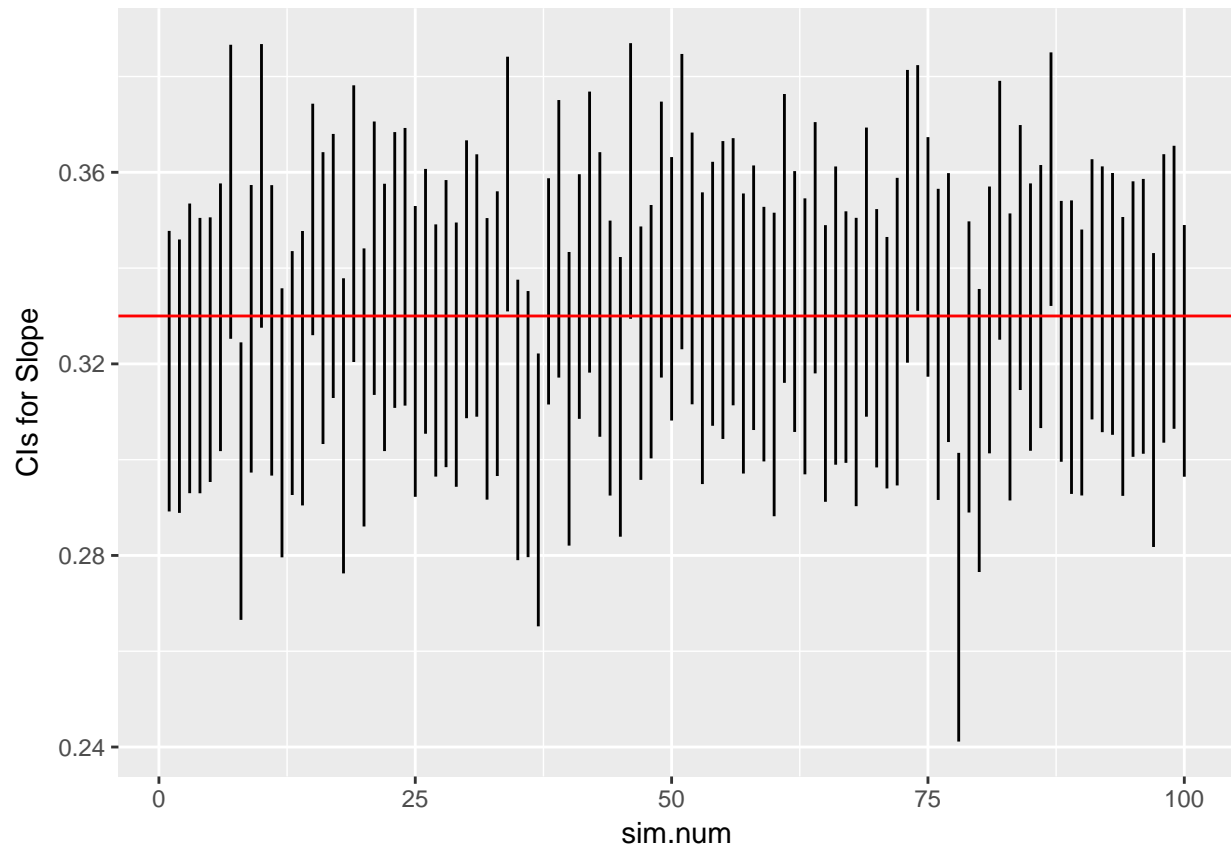
# Easier to make simulation data a data frame
ht.wt.sims <- as.data.frame(sim.num = 1:n.simulations, height.weight.sim.data)

# Add a variable to index the simulation number
ht.wt.sims$sim.num <- 1:n.simulations

# Plot beta0 min and max (now called V3, V4)
ggplot(data = ht.wt.sims) +
  aes(x = sim.num) +
  geom_linerange(mapping = aes(ymin=V3,
                                ymax=V4)) +
  ylab("CIs for Intercept") +
  geom_hline(yintercept = beta0, color = "red")
```



```
# Plot beta1 min and max (now called V5, V6)
ggplot(data = ht.wt.sims) +
  aes(x = sim.num) +
  geom_linerange(mapping = aes(ymin=V5,
                              ymax=V6)) +
  ylab("CIs for Slope") +
  geom_hline(yintercept = beta1, color = "red")
```



## How about a factorial design?

- Dependent variable: RT
- Independent variables
  - Fixed
    - \* Symbol type: {letter, number}
  - Random
    - \* Subject mean RT
- Hypothesis:
  - RT to detect numbers is lower than for letters
  - There is no order effect
- Rick wishes he'd found this first: <https://www.r-bloggers.com/design-of-experiments-%E2%80%93-93-full-factorial-designs/>

---

```
# Simulation parameters
n.subs = 30
trials.per.cond = 100

letters.numbers.rt.diff = 50 # ms
rt.mean.across.subs = 35
sigma = 50
cond.labels = c("letter", "number")
cond.rts <- c("letter" = 0, "number" = letters.numbers.rt.diff)

stim.types <- factor(x = rep(x = c(1,2), trials.per.cond), labels = cond.labels)
```

```

#sample(factor(x=rep(c("letter", "number"), 100)), 200)
random.stim.types <- sample(stim.types, trials.per.cond*length(cond.labels))

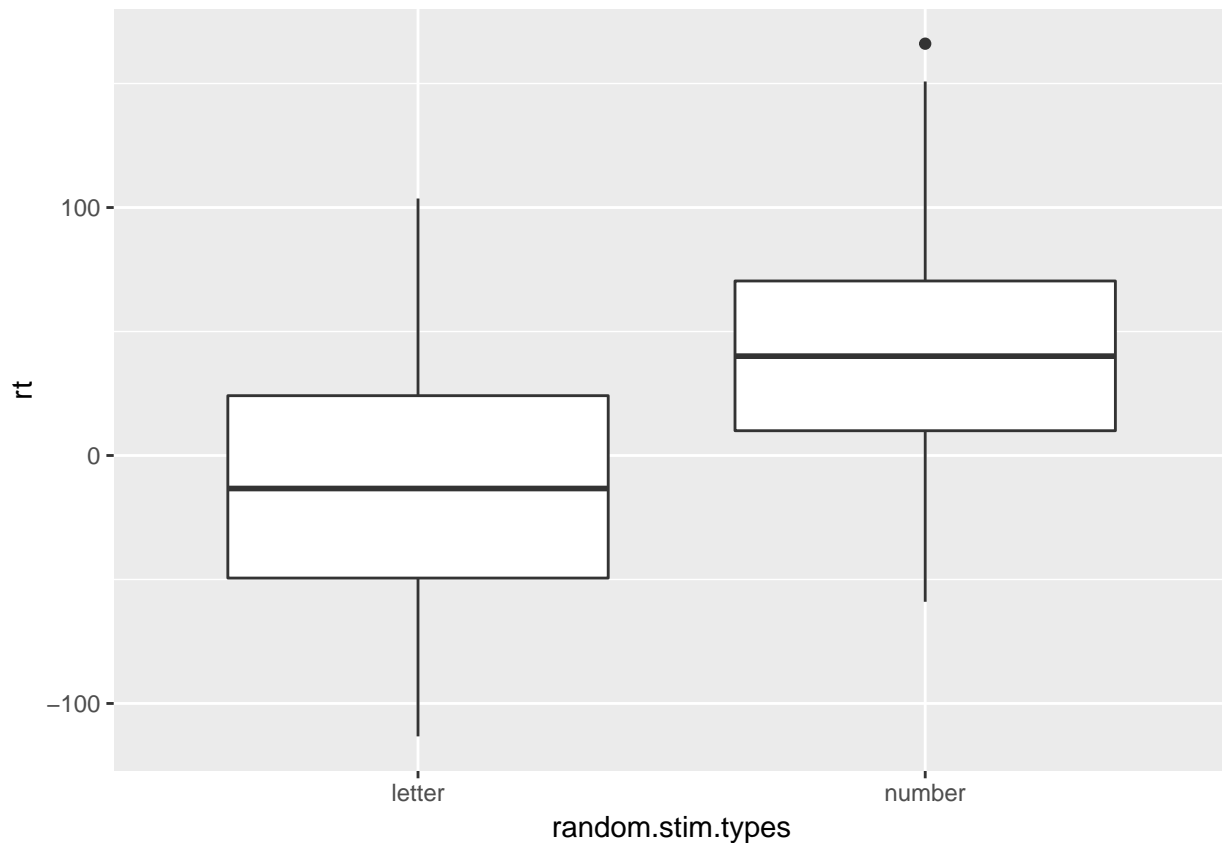
mean.sub.rt <- rnorm(n = 1, mean = rt.mean.across.subs, sd = sigma)
trial.rt <- array(0, dim = length(random.stim.types))

# Generate RTs based on trial, condition
for (t in 1:length(random.stim.types)) {
  trial.rt[t] <- mean.sub.rt + cond.rts[random.stim.types[t]] + rnorm(n = 1, mean = 0, sd = sigma)
}

# Make data frame
letter.number.df <- data.frame(trial = 1:length(random.stim.types), stim = random.stim.types, rt = trial.rt)

ggplot(data = letter.number.df, aes(x = random.stim.types, y = rt)) + geom_boxplot()

```



Put this in a function

```

Simulate.sub.rt <- function(trials.per.cond = 100,
                           letters.numbers.rt.diff = 50,
                           rt.mean.across.subs = 350,
                           sigma = 50) {

  cond.rts <- c("letter" = 0, "number" = letters.numbers.rt.diff)
  stim.types <- factor(x = rep(x = c(1,2), trials.per.cond), labels = c("letter", "number"))

```

```

random.stim.types <- sample(stim.types, 200)

mean.sub.rt <- rnorm(n = 1, mean = rt.mean.across.subs, sd = sigma)
trial.rt <- array(0, dim = length(random.stim.types))

# Generate RTs based on trial, condition
for (t in 1:length(random.stim.types)) {
  trial.rt[t] <- mean.sub.rt + cond.rts[random.stim.types[t]] +
    rnorm(n = 1, mean = 0, sd = sigma)
}

# Make data frame
letter.number.df <- data.frame(trial = 1:length(random.stim.types),
                               stim = random.stim.types,
                               rt = trial.rt)
}

```

Create code to a set of participants and create one merged data frame

```

Make.sub.rt.df <- function(sub.id) {
  sub.rt.df <- Simulate.sub.rt()
  sub.rt.df$sub.id <- sub.id
  sub.rt.df
}

# Use lapply to make separate data frames for all subs
sub.rt.df.list <- lapply(1:n.subs, Make.sub.rt.df)

# Use Reduce() with the merge function to make one big file
sub.rt.df.merged <- Reduce(function(x, y) merge(x, y, all=TRUE), sub.rt.df.list)

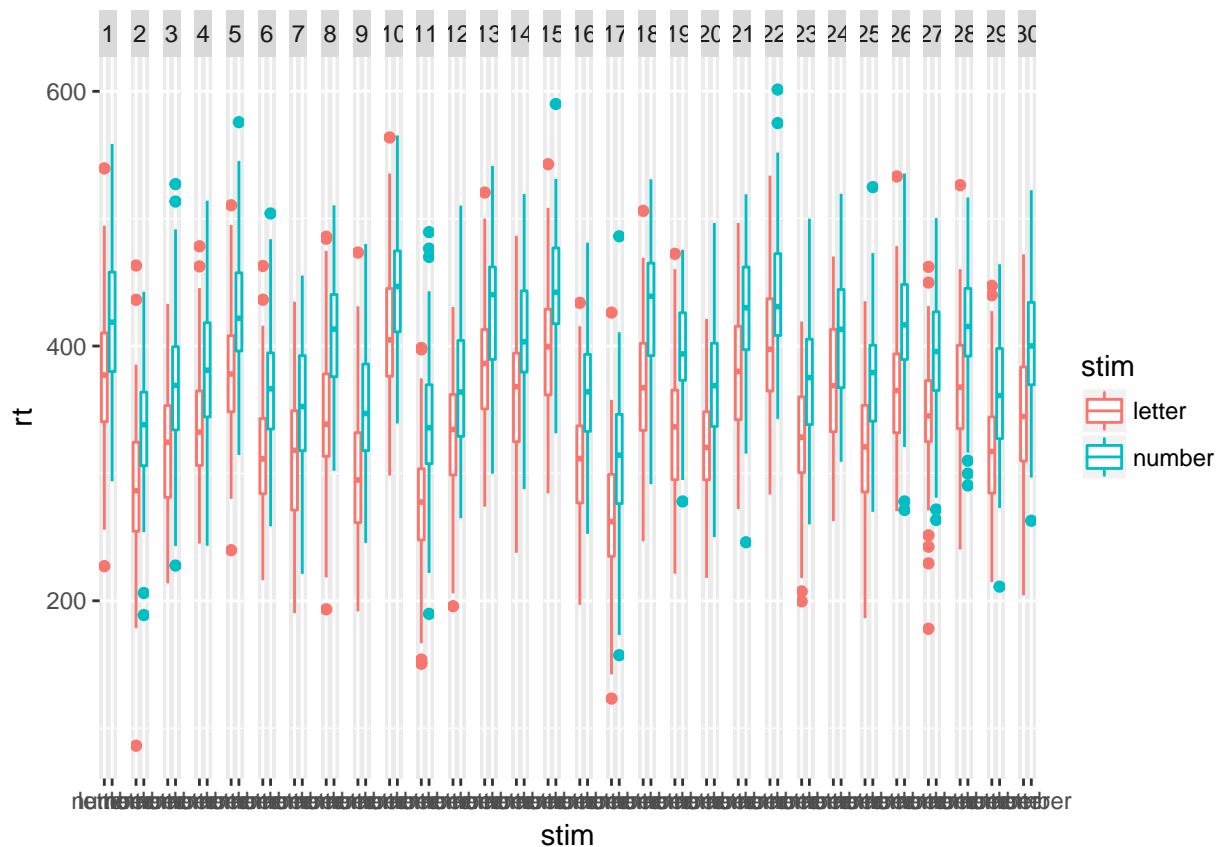
```

Now, want to see what we have?

```

ggplot(data = sub.rt.df.merged) +
  aes(x=stim, y=rt, color=stim) +
  geom_boxplot() +
  facet_grid(facets = . ~ as.factor(sub.id))

```



And, just for fun

```
library(lme4)

## Loading required package: Matrix

fit1 <- lmer(formula = rt ~ stim + (1|sub.id), data = sub.rt.df.merged)
summary(fit1)

## Linear mixed model fit by REML ['lmerMod']
## Formula: rt ~ stim + (1 | sub.id)
## Data: sub.rt.df.merged
##
## REML criterion at convergence: 64090
##
## Scaled residuals:
## Min      1Q  Median      3Q      Max
## -4.0682 -0.6683  0.0002  0.6546  3.4785
##
## Random effects:
## Groups Name Variance Std.Dev.
## sub.id (Intercept) 1278 35.75
## Residual 2496 49.96
## Number of obs: 6000, groups: sub.id, 30
##
## Fixed effects:
```



```
##               Estimate Std. Error t value
## (Intercept)   342.34      6.59    51.95
## stimnumber    48.89      1.29    37.91
##
## Correlation of Fixed Effects:
##           (Intr)
## stimnumber -0.098
```

Or

```
fit2 <- aov(formula = rt ~ stim + Error(as.factor(sub.id)), data = sub.rt.df.merged)
summary(fit2)
```

```
##
## Error: as.factor(sub.id)
##               Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  29 7483681  258058
##
## Error: Within
##               Df Sum Sq Mean Sq F value Pr(>F)
## stim           1 3585648 3585648    1437 <2e-16 ***
## Residuals 5969 14895728    2496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Practice with visualization

- Work through the Wickham & Grolemund chapter on your own.
  - <http://r4ds.had.co.nz/data-visualisation.html>

## Learning by example

```
# Installing packages
list.of.packages <- c("xlsx", "ggplot2", "nlme", "foreign", "gplots", "stats", "psych", "ISwR", "ggm", "ca")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

# Loading packages
lapply(list.of.packages, require, character.only = TRUE)

https://github.com/psu-psych-511-2017-spring/liu\_yushuang/blob/master/output.Rmd
```

## A syntax I like

```
my.data <- data.frame(x.var = ..., y.var = ...)

my.data %>%
  ggplot() +      # pipe %>% operator feeds it my.data
  aes(x=x.var, y=y.var) + # plus + operator 'adds' to plot
  geom_point() -> # assign -> operator saves output
my.scatter.plot # variable containing plot
```

```
my.scatter.plot # prints it
```

- Uses 'pipe' %>% operator and 'assign' -> operator
- Easy to follow flow of what's going on

## Assignment (due Monday, February 20)

- Create your own simulated data set for a real or proposed study.
  - You may adapt or build upon the examples used in class.
- Plot the results of your simulation using ggplot2 commands.