GEM500: Simulating Data

Jen Baron | University of British Columbia, Department of Forest and Conservation Sciences [jenbaron@mail.ubc.ca](mailto:jenbaron@mail.ubc.ca)

Sepetember 16, 2021

Table of Contents

[Load Packages 1](#_Toc97736187)

[Scatter Plot 1](#_Toc97736188)

[Boxplot 4](#_Toc97736189)

[Export Results 6](#_Toc97736190)

[Reproducibility 6](#_Toc97736191)

Here is a simple (optional) workflow to generate and visualize some data for research proposals. This is not required for GEM500 but is one way to simulate data based on predictions.

# Load Packages

library(ggplot2)  
library(truncnorm)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)

# Scatter Plot

(Two continuous variables)

Imagine a hypothetical relationship between elevation and timber harvesting. As elevation increases, harvesting decreases. I want to generate some data to demonstrate this relationship.

Let’s generate a variable (elevation) based on potentially realistic values. Here I generate 20 elevation values with minimum 0m, maximum 1500m, mean 200m, and standard deviation 200m.

Next, I generate harvesting values based on my prediction of the negative relationship between elevation and harvesting. To do this, I played around with different slopes and y-intercepts to generate harvesting values that make sense.

For a negative relationship, the slope should be negative. I add a positive intercept because negative harvesting values wouldn’t make sense for my data.You could also add another step that converts negative values back to zero if needed.

To create variation in the data, I add some random variability (residuals) from a normal distribution (-50, 50) with mean 0, standard deviation 20.

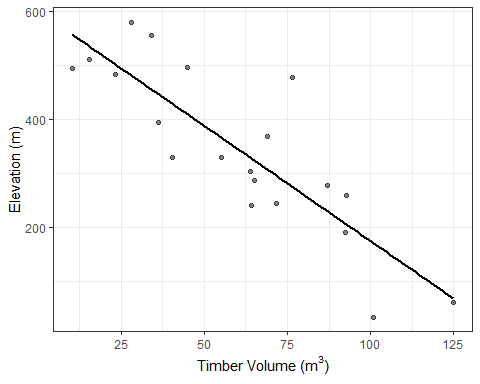
Finally, I join these two simulated vectors in a data frame and use ggplot2 to create a simple plot of my predicted results.

set.seed(110) #Set the seed of R's random number generator, which is useful for creating simulations or random objects that can be reproduced. You only need to do this once per session.  
  
#Sample from truncated normal distribution with limits (a,b), number of samples, mean, and standard deviation  
elevation <- rtruncnorm(a=0, b=1500, n=20, mean = 200, sd = 200)  
  
#y = slope(x) + intercept + random variation  
harvesting <- (elevation\*(-1/6) + 120 + rtruncnorm(a=-50, b=50, n=20, mean = 0, sd = 20))  
  
# Join observations in a data frame  
data1 <-   
 data.frame(  
 "elevation" = elevation,  
 "harvesting" = harvesting)  
head(data1)

## elevation harvesting  
## 1 258.2390 92.66399  
## 2 477.7726 76.38487  
## 3 329.8020 40.36908  
## 4 495.5752 10.39591  
## 5 287.7440 65.15053  
## 6 304.4636 63.88480

p1 <- ggplot(data1, aes(x=harvesting, y=elevation)) +  
 geom\_point(alpha=0.5) + #add point data  
 geom\_smooth(method = "lm", se = FALSE, col = "black") + #add linear regression line  
 labs(y = "Elevation (m)", x = expression(paste("Timber Volume (",m^3, ")"))) + #add axis labels  
 theme\_bw()  
p1

## `geom\_smooth()` using formula 'y ~ x'

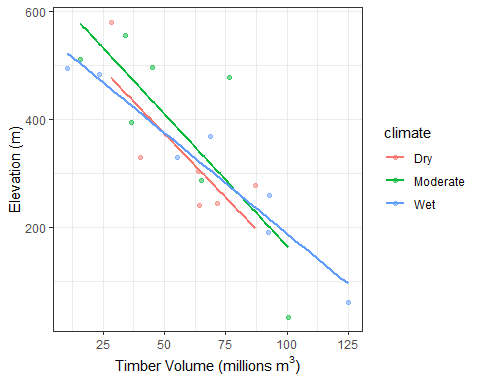


Let’s try adding in another (discrete) factor and visualizing the results.

Here, I’m adding a random climate column with three categories (wet, moderate, dry). If you wanted this column to fit a specific prediction, you could specify which cases (e.g., elevation < 200, timber volume > 50 = wet) you want assigned to which climate class using if else statements or dplyr::mutate and dplyr::case\_when.

#Create 3 climate classes, replicate in a vector of length 20, make a factor  
climate <- as.factor(rep(c("Wet", "Moderate", "Dry"), length.out = 20))  
  
#Join with data from before  
data1b <- cbind(data1, climate)  
  
p2 <- ggplot(data1, aes(x=harvesting, y=elevation, col=climate)) + #Assign colour based on climate class  
 geom\_point(alpha=0.5) + #add point data  
 geom\_smooth(method = "lm", se = FALSE) + #add linear regression line (no error estimates)  
 labs(y = "Elevation (m)", x = expression(paste("Timber Volume (millions ", m^3, ")"))) + #add axis labels  
 theme\_bw()  
p2

## `geom\_smooth()` using formula 'y ~ x'



# Boxplot

(One continuous, one discrete variable)

This time, let’s image the density of birds nesting in different tree species. I hypothesize that bird density is greatest in Douglas-fir, and lower in Cedar and Hemlock.

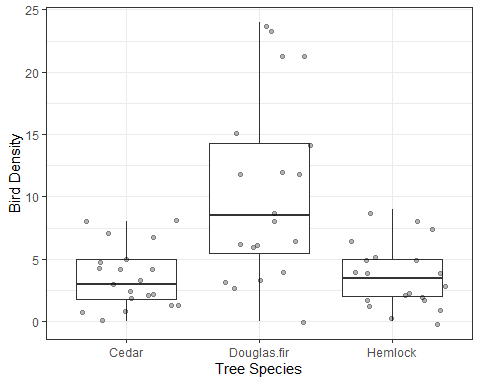
First, I generate the data for one species (cedar) from a truncated normal distribution. Next, I simulate data for douglas-fir, relative to cedar (3x the values of cedar, plus some random variation). Finally, I set the values for hemlock to be the same is ceder plus some random variation.

Next, I join the observations in a data frame and restructure it using tidyr::pivot\_longer to plot it in ggplot2 using a boxplot.

#Simulate some data  
cedar <- rtruncnorm(a=0, b=20, n=20, mean = 2, sd = 5) %>% round(0) #rounded to 0 decimal places because these should be integers  
fir <- cedar\*3 + rtruncnorm(a=-1, b=1, n=20, mean = 0, sd = 1) %>% round(0)  
hemlock <- cedar\*1 + rtruncnorm(a=-1, b=2, n=20, mean = 0, sd = 1) %>% round(0)  
  
# Join observations in a data frame  
data2 <-   
 data.frame(  
 "Cedar" = cedar,  
 "Douglas.fir" = fir,  
 "Hemlock" = hemlock) %>%  
 pivot\_longer(1:3, names\_to = "tree", values\_to = "birds") #Convert from wide to long format  
head(data2)

## # A tibble: 6 x 2  
## tree birds  
## <chr> <dbl>  
## 1 Cedar 2  
## 2 Douglas.fir 6  
## 3 Hemlock 3  
## 4 Cedar 4  
## 5 Douglas.fir 12  
## 6 Hemlock 4

p3 <- ggplot(data2, aes(x=tree, y=birds)) +  
 geom\_boxplot(outlier.shape=NA) + #create boxplots, remove outliers since I'll plot the raw data on top  
 geom\_jitter(alpha=0.3) + #add the raw data, alpha specifies transparency  
 labs(y = "Bird Density", x = "Tree Species") + #add axis labels  
 theme\_bw()  
p3



# Export Results

ggsave("figure1.jpg", p1) #you can specify dpi, width, height, device (png, dpf, etc.) here

## Saving 5 x 4 in image

## `geom\_smooth()` using formula 'y ~ x'

ggsave("figure2.jpg", p2)

## Saving 5 x 4 in image  
## `geom\_smooth()` using formula 'y ~ x'

ggsave('figure3.jpg', p3)

## Saving 5 x 4 in image

# Reproducibility

This tells us when we ran our analysis, under what operating system, and what packages we used.

Sys.time()

## [1] "2022-03-09 16:36:08 PST"

sessionInfo()

## R version 4.1.2 (2021-11-01)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 22000)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=English\_United States.1252   
## [2] LC\_CTYPE=English\_United States.1252   
## [3] LC\_MONETARY=English\_United States.1252  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=English\_United States.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] tidyr\_1.1.4 dplyr\_1.0.7 truncnorm\_1.0-8 ggplot2\_3.3.5   
##   
## loaded via a namespace (and not attached):  
## [1] highr\_0.9 pillar\_1.6.4 compiler\_4.1.2 tools\_4.1.2   
## [5] digest\_0.6.28 lattice\_0.20-45 nlme\_3.1-153 evaluate\_0.14   
## [9] lifecycle\_1.0.1 tibble\_3.1.6 gtable\_0.3.0 mgcv\_1.8-38   
## [13] pkgconfig\_2.0.3 rlang\_0.4.12 Matrix\_1.3-4 rstudioapi\_0.13   
## [17] cli\_3.1.0 DBI\_1.1.1 yaml\_2.2.1 xfun\_0.28   
## [21] fastmap\_1.1.0 withr\_2.4.2 stringr\_1.4.0 knitr\_1.36   
## [25] generics\_0.1.1 vctrs\_0.3.8 grid\_4.1.2 tidyselect\_1.1.1  
## [29] glue\_1.5.0 R6\_2.5.1 fansi\_0.5.0 rmarkdown\_2.11   
## [33] farver\_2.1.0 purrr\_0.3.4 magrittr\_2.0.1 splines\_4.1.2   
## [37] scales\_1.1.1 ellipsis\_0.3.2 htmltools\_0.5.2 assertthat\_0.2.1  
## [41] colorspace\_2.0-2 labeling\_0.4.2 utf8\_1.2.2 stringi\_1.7.5   
## [45] munsell\_0.5.0 crayon\_1.4.2