Intro to R: Live Coding Sessions

Oct 16 Randomization

```
v=1:10
v

#randomly sample 5 out of 10 numbers
#sample without replacement
sample(v, 5, replace=F)


#if I sample without replacement the length of the vector (i.e., all numbers), then I'm just shuffling the order of the numbers. This is called "permutation"
sample(v, 10, replace=F)

#I can't sample without replacement MORE numbers than there are.
sample(v, 11, replace=F)

#make the random sampling replicable by setting a "seed" number
set.seed(5)
sample(v, 5, replace=F)


##now sample WITH replacement
set.seed(2)
sample(v, 10, replace=T)

##Let's fiddle with the prob= argument
#simulate 100 coin flips
coin=sample(c(0,1), 100, replace=T)
table(coin)

#simulate 100 coin flips but with biased probabilities
coin=sample(c(0,1), 100, replace=T, prob=c(0.25, 0.75))
table(coin)


### sample from distributions

#how to sample numbers from a normal distribution
rn=rnorm(100000, mean=0, sd=1)
hist(rn)

#illustrate how the distribution is different by specifying sd
```

Intro to R: Live Coding Sessions

```
rn1=rnorm(100000, mean=0, sd=1)
rn10=rnorm(100000, mean=0, sd=10)
hist(rn1, xlim=c(-50,50))
hist(rn10, xlim=c(-50,50))

#sample from a uniform distribution
ru=runif(1000, min=0, max=1)
hist(ru)

#simulate coin flips with binomial distribution
rbinom(n=100, size=1, prob=0.5)

### try out randomization/bootstrap tests
trees
?trees

plot(Height~Girth, data=trees)

#correlation test
cor.test(trees$Girth, trees$Height)

#correlation test
cor.test(trees$Girth, trees$Height)

#just get the correlation coefficient and save it.
obs.cor=cor(trees$Girth, trees$Height)
obs.cor

sample(trees$Height, length(trees$Height), replace=F)

#use a for loop to conduct the randomization 10,000x
rep=10000 #set the number of replication
rand.cor=vector(length=rep) #set up empty vector

for(i in 1:rep){
  rand.cor[i]=cor(trees$Girth, sample(trees$Height, length(trees$Height), replace=F))
}


#plot distribution of the randomized correlation coefficient
hist(rand.cor)

#ask: what is the probability that I got the observed correlation coefficient by chance?
```

```
rand.cor>=obs.cor #True or False: the correlation coefficient from the randomization is greater or equal to the observed correlation

which(rand.cor>=obs.cor) #Which ones are greater or equal?

length(which(rand.cor>=obs.cor))/rep #the probability that the randomization produced a correlation coefficient greater or equal to the observed.


###################################
### Oct 14: Custom functions and if-else statements

4 %% 2 #%% operator gives you the remainder of a division

5 %% 2
6%%2
7%%2

#let's ask R if a number "num" is even or odd
num=4
if(num %% 2 == 0) {print("the number is even")} #you need the double-equals sign "==" which asks "are these things the same"?

## if you just use if(), you will get a result if the condition is met, but nothing will happen if the condition is NOT met

num=10
if(num %% 2 == 0) {print("the number is even")} else {print("the number is odd")}

#this works fine as long as the number is an integer because it has to be EITHER even or odd.

num=10.6
if(num %% 2 == 0) {print("the number is even")} else {print("the number is odd")}

# so let's add on another criterion for if the number is not whole.
num=11.2
if(num %% 2 == 0) {print("the number is even")} else {
  if(num %% 2==1) print("the number is odd")}
#but this one will end in a dead-end


num=11.2
if(num %% 2 == 0) {print("the number is even")} else {
  if(num %% 2==1) print("the number is odd") else {
```

```
   print("the number is neither odd nor even")
   }}

######################
### Oct 7

## speed test: apply vs. loop

dat=data.frame(n1=1:10000, n2=1:10000)
dat

dat$n1+dat$n2

#a very roundabout way to get sum of each row
# x will take values from 1 to 10,000.
# for each value of x, take the x-th row, first column and add x-th row, second column
result_apply=sapply(1:10000, function(x) sum(dat[x,]))

#how fast?
system.time({
  result_apply=sapply(1:10000, function(x) sum(dat[x,])) })

#same thing as above, but use loop
#first, create an empty vector of length=10,000 where the result will go
# then, for each value of i from 1 to 10,000, calculate sum of i-th row, first column and i-th row,
second column
result_loop=vector(length=10000)
for(i in 1:10000){
  result_loop[i]=sum(dat[i,])
}
result_loop

#how long?
system.time({
  result_loop=vector(length=10000)
  for(i in 1:10000){
    result_loop[i]=sum(dat[i,])
  }
})


## do the same thing, but with apply()
```

Intro to R: Live Coding Sessions

```
system.time({
  result_apply=apply(dat, 1,  function(x) sum(x)) })


#demonstrate for loop using population growth model

#define parameters
N0=100 #initial pop size
lambda=1.5 #pop growth rate
t=0:10 #take time steps 0 through 10

N=N0*lambda^t
N

plot(t, N, type="b")


#using these different lambda values, project population growth
N0=100 #initial population size
lambdas=rlnorm(100, meanlog=0) #100 different values for lambda
times=100 #100 time steps

N=vector(length=times) #create empty vector

#set population at time 1 = N0
N[1]=N0

#for each time step, take the N at last time step, then multiply by the lambda value
for(i in 2:times){
  N[i]=N[i-1]*lambda[i]
}


N # see the results of the simulation

plot(1:100, N, type="b") #plot the population size across time. This will look different each time
you run the for loop.
```

```
######################
## Oct 2
```

```
##fit linear regression line to scatterplot, the old fashion way:
lm.mod=lm(mpg~wt, data=mtcars) # fit the linear model
plot(mpg~wt, data=mtcars, pch=19, xlim=c(0,6), ylim=c(0,40))
abline(a=37.285, b=-5.344)

##fit linear regression line, slightly more elegant way:

lm.mod=lm(mpg~wt, data=mtcars) # fit the linear model
xv=seq(min(mtcars$wt), max(mtcars$wt), length=2) # set a series of values along the x-axis
yv=predict(lm.mod, data.frame(wt=xv)) # use predict() to get what the y-axis value should be
along each of those x-axis values

plot(mpg~wt, data=mtcars, pch=19,  xlim=c(0,6), ylim=c(0,40)) #plot the scatterplot
lines(xv, yv) #add the fit line

lm.fit=lm(mpg~wt, data=mtcars)
glm.fit=glm(mpg~wt, data=mtcars, family="gaussian")

summary(lm.fit)

summary(glm.fit)

#ANOVA
aov.mod=aov(mpg~factor(vs), data=mtcars)
summary(aov.mod)

#in this case, it actually doesn't matter if you use factor(vs)
aov.mod2=aov(mpg~vs, data=mtcars)
summary(aov.mod2)

## You can run this as a linear model too
lm.mod2=lm(mpg~factor(vs), data=mtcars)
summary(lm.mod2)

## model with multiple variables

#additive model with both weight and engine shape (using "+")
lm.mod3=lm(mpg~wt + factor(vs), data=mtcars)
summary(lm.mod3)

#interactive effect between weight and engine shape (using "*")
lm.mod4=lm(mpg~wt * factor(vs), data=mtcars)
summary(lm.mod4)
```

```
## use three categories
lm.mod5=lm(mpg~wt + factor(gear), data=mtcars)
summary(lm.mod5) #this gives you p-values for comparisons between "3 gears" vs. "4 gears"
and "3 gears" vs. "5 gears".
anova(lm.mod5) # this give you overall p-value for "gear type"

## syntax for bulding a custom function

custom.mean=function(x){sum(x)/length(x)}

custom.mean(1:10)



#########################
### September 30


#load libraries
library(tidyverse)
library(cowplot)

#import data

boloria=read.csv("data/boloria.csv")
colias=read.csv("data/colias.csv")

## create summary data for snowmelt and temperature by year
year.dat=boloria %>%
  group_by(year) %>%
  summarize(mean.snow=mean(snow))

year.dat


##run a linear model to test if snowmelt data changes linearly with year

#lm is the function for "linear model"
#stats functions typically have a y~x format for describing the model you want to fit
fit.snow=lm(mean.snow~year, data=year.dat)

# this output will just give you the intercept and slope of the fit
fit.snow
```

```
# to look at the stats summary:
summary(fit.snow)

# go deeper into looking at stats summary
str(summary(fit.snow))

summary(fit.snow)$residuals



################
### September 23

install.packages("tidyverse")
library(tidyverse)

#dataframe
iris
head(iris)

#tibble
iris_tbl=tibble(iris)
iris_tbl
("young", "old", "young", "old")
age
old = age=="old"  #ask, is the value in "age" = old?
old

#now plot this as dots and errorbars
ggplot(iris_summary, aes(x=Species, y=mean_sl)) +
  geom_point(size=5) +
  geom_errorbar(aes(ymin=mean_sl-sd_sl, ymax=mean_sl+sd_sl), width=0.2)

##merging datasets
iris
iris_summary

iris %>%
  tibble() %>%
  left_join(iris_summary, by=c("Species"="Species")) #take iris, then merge the iris_summary
data, matching by species

#use the merged dataset to calculate z-score based on each species means and sd.
```

Intro to R: Live Coding Sessions

```
iris %>%
  tibble() %>%
  left_join(iris_summary) %>%
  mutate(z_score=(Sepal.Length-mean_sl)/sd_sl)

#########################
# September 16
Go to the ggplot2 extensions website
https://exts.ggplot2.tidyverse.org/

install.packages("cowplot")
library(cowplot)
library(ggplot2)

ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_boxplot() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width")


ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_violin() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width")


#make multipanel figure with cowplot

p1=ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_boxplot() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width")


p2=ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_violin() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width")

p1
p2

#cowplot function to put together two plots in one figure
plot_grid(p1, p2)
```

```
#use cowplot theme

p1=ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_boxplot() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width") +
  theme_cowplot()


p2=ggplot(iris, aes(x= Species, y=Sepal.Width, fill=Species)) +
  geom_violin() +
  scale_fill_brewer(palette="RdYlBu") +
  ylab("Sepal Width")+
  theme_cowplot()

p1
p2

#cowplot function to put together two plots in one figure
plot_grid(p1, p2, labels="AUTO")

#let's try plotting with imported data

dat=read.csv("data/SampleData.csv")

ggplot(dat, aes(x=sex, y=weight)) +
  geom_boxplot(aes(color=sex)) +
  geom_point()

ggplot(dat, aes(x=sex, y=weight)) +
  geom_boxplot() +
  geom_jitter(aes(color=age), width=0.2)



########################
#September 9th, 2025

# practice plotting with base R using imported data.

eggs=read.csv("data/EggMeasurements2006_clean.csv")

plot(eggs$Length, eggs$Mass) #basic syntax: plot(x,y)
```

Intro to R: Live Coding Sessions

```
plot(eggs$Mass~eggs$Length) #formula syntax: plot(y~x)

plot(Mass~Length, data=eggs) #you can use the formula syntax where you just input the column names as formula, and then use "data=" to tell R which dataframe you're using

egg_volume=eggs$Length * eggs$Width


plot(eggs$Mass~egg_volume) #you can plot two things that are not in the same dataframe, as long as they have the same number of elements.


plot(Mass~Nest, data=eggs)

#R is treating nest number as a numeric. I want it to treat it like a category
eggs$Nest=as.factor(eggs$Nest)

#once it knows Nest is a category (or factor), it plots this data as a boxplot
plot(Mass~Nest, data=eggs)


##zoom into x values 43 to 55
plot(Mass~Length, data=eggs, xlim=c(43,55))

plot(Mass~Length, data=eggs, xlim=c(43,55), pch=19, col="tomato")

#pch = 21 allows you to have different outline color vs. fill color
plot(Mass~Length, data=eggs, xlim=c(43,55), pch=21, bg="tomato")



plot(Mass~Length, data=eggs, xlim=c(43,55), pch=21, bg="#DAB785", col="#04395E")

#using rgb
plot(Mass~Length, data=eggs, xlim=c(43,55), pch=21, bg=rgb(1,0,0,0.5), col="black")

def.par=par(no.readonly = TRUE) #save the default parameters

par(bty="l")
plot(Mass~Length, data=eggs, xlim=c(43,55), pch=21, bg=rgb(1,0,0,0.5), col="black")

par(def.par)
```

Intro to R: Live Coding Sessions

```
#let's use the par() functions to make a two-panel plot

par(mfrow=c(1,2)) #make panels of 1 row, 2 columns
plot(Mass~Length, data=eggs, pch=21, bg="red", col="black")

plot(Mass~Width, data=eggs, pch=21, bg="blue", col="black")
```

```
##### Sept4,2025

dat=read.csv("data/SampleData.csv")
dat


head(dat) #first 6 rows of data
tail(dat) #last 6 rows of data

str(dat) #structure of an object

#use $ to get a specific column

dat$sex
dat$weight

names(dat) #column names as character string

dat_error=read.csv("data/SampleData_w_errors.csv", na.strings="#N/A!")
dat_error

str(dat_error)

dat_error$size
```

```
## Sep 2

getwd() #get working directory

list.files() # list all files and folders in working directory
```

Intro to R: Live Coding Sessions

```
list.files("Documents") #look inside a folder

list.files("Documents/GitHub") #go into subfolder

list.files("../")

##to import data using absolute pathname
dat=read.csv("/Users/dshizuka2/Downloads/SampleData.csv")
dat


install.packages("readxl")
library(readxl)

dat3=read_excel("data/SampleData.xlsx")
dat3

#load an .rdata file that has 3 different data frames.
load("data/coot_data.rdata")
eggs
chicks




## Aug 28, 2025

install.packages("vioplot")

library(vioplot)

library(help="vioplot")

NA
"NA"

#concatenate
c(1, 4, 6, 9)

#make a matrix
# I can use NA in place of a number
mat=matrix(c(1, 3, 5, NA, 9, 11), nrow=2)
mat
```

Intro to R: Live Coding Sessions

```
a=c(TRUE,FALSE)
a
a+0
```