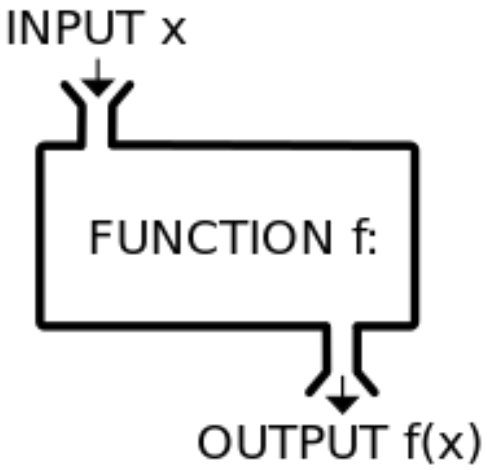
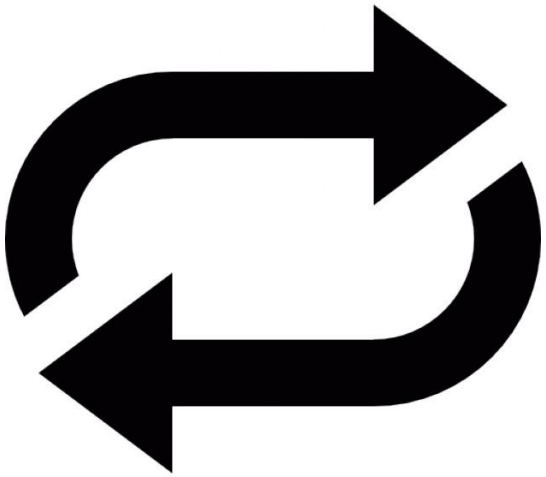
Getting **R** to do your dirty work: automation using functions, apply, and loops

Alex Filazzola





For code, bigger isn't better

```r
## need to code categorical data as ordinal (1/0 for shrub and open)
micros <- factor(data$Microsite)
micros <- as.numeric(micros)-1
consumers <- factor(data$Exclosure)
consumers <- as.numeric(consumers)-1
data[,"micros"] <- micros
data[,"consumer"] <- consumers

brome.dom <- data$brome/data$abundance ## a measurement of brome dominance
data[,"brome.dom"] <- brome.dom


## specify model. Predictors, responses and latent variables
mymodel <- '
# latent variable
community =~ Biomass + abundance + brome.dom

# regressions
community ~ consumer
community ~ micros
community ~ SWC.initial
'

## fit model
fit1 <- sem(mymodel, data = subset(data, Year==2014), estimator = "ML", std.lv=TRUE, se="bootstrap", test="bootstrap")
fit2 <- sem(mymodel, data = subset(data, Year==2016), estimator = "ML", std.lv=TRUE , se="bootstrap", test="bootstrap")


##summarize results
summary(fit1)
summary(fit2)

summary(fit1, standardized=TRUE, rsq=T)
summary(fit2, standardized=TRUE, rsq=T)
```

General rule

Repetition of tasks between studies = functions

Repetition of tasks within studies = apply family / loops

Functions

Function name

Function content

```
function ( arglist )  {body}
```

List of arguments, variables, etc

Functions

$$\overline{X} = \frac{\sum X}{N}$$

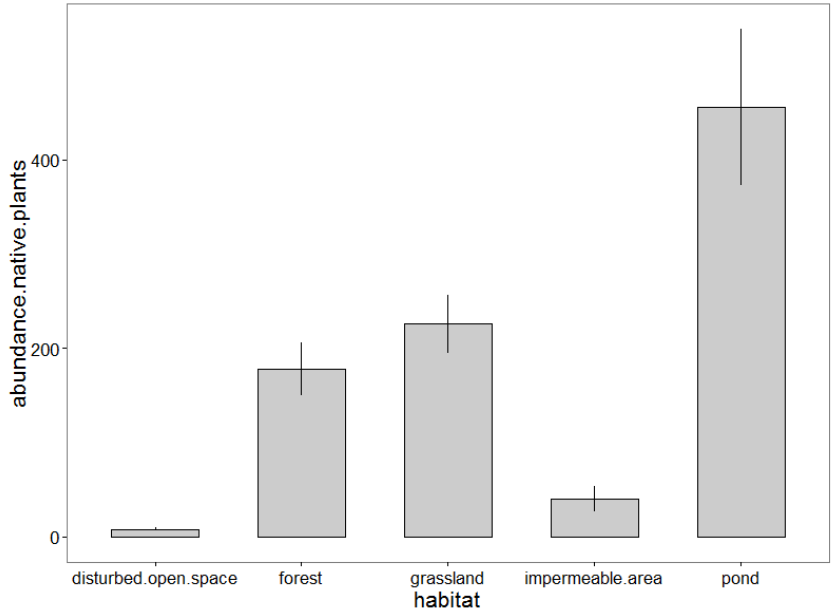
```r
mean <- function(x)
{
  sum(x)/length(x)
}
```

Functions

$$\text{RII} = \frac{(\text{shrub} - \text{open})}{(\text{shrub} + \text{open})}$$

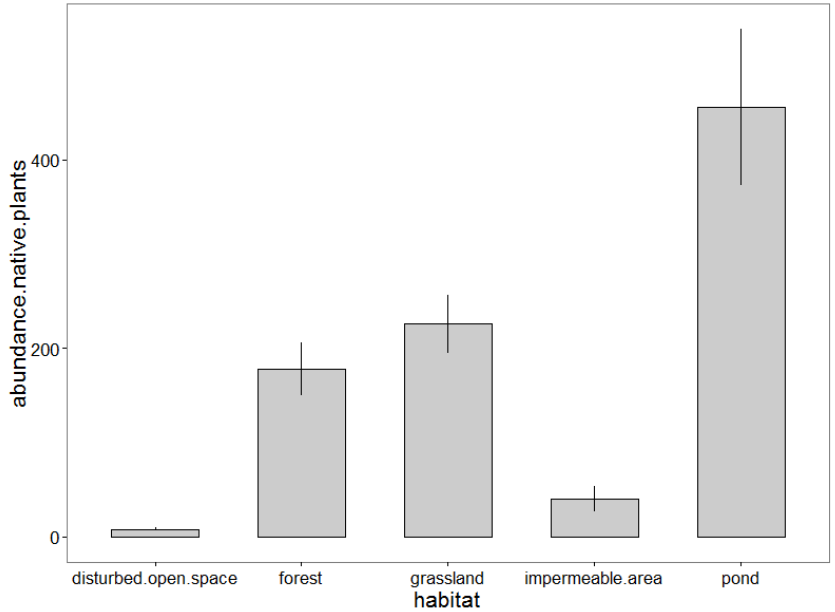
```r
rii <- function(x, j, var)
{
s1 <- subset(x, Microsite == "shrub", select=var)
o1 <- subset(x, Microsite == "open", select=var)
return1 <- (s1 - o1) / (s1+o1)
x1 <- x[seq(1, nrow(x), by = 2),]
return2 <- cbind(x1[j], return1)
return2[is.na(return2)] <- 0
print(return2)
}
```

Functions




```
bar.plot <- function(data, predictor, response)
{
ggplot(summary.data, aes(x=summary.data[,predictor], y=summary.data[,response]))
+ geom_bar(stat="identity", fill="grey80", color="black", width=0.6)+ theme_bw()+
 theme(panel.grid.minor=element_blank(),panel.grid.major=element_blank())+geom_er
rorbar(aes(ymin=y.min,ymax=y.max, width=0))+ xlab(predictor) + ylab(response)+the
me(text=element_text(size=16))
}
```

Functions



data

predictor – x axis

response – y axis

…

```
bar.plot <- function(data, predictor, response)
{
ggplot(summary.data, aes(x=summary.data[,predictor], y=summary.data[,response]))
+ geom_bar(stat="identity", fill="grey80", color="black", width=0.6)+ theme_bw()+
 theme(panel.grid.minor=element_blank(),panel.grid.major=element_blank())+geom_er
rorbar(aes(ymin=y.min,ymax=y.max, width=0))+ xlab(predictor) + ylab(response)+the
me(text=element_text(size=16))
}
```

Functions

Indices

```
rii(data, j, var)
LRR(data, treatment, control)
simpsons(data)
SE(x)
```

Plots

```
bar.plot(data, predictor, response)
stackedbar.plot(data, predictor1,  predictor2, response)
pie.chart(data, predictor, response)
```

Functions

source(“ecofunctions.r”)

Reduces bloated code

Increases efficiency

Reduces mistakes