Empirical Reasoning Center
R Workshop (Summer 2017)
Session 3

1 User-defined functions

One advantage of R is its ability to incorporate user-defined functions. Like the built-in functions above, user-defined functions can perform complex (or, not so complex) sets of operations many times (e.g., calculating means or estimating a parameter that requires you to hand-code an estimator). Functions allow you to write code once and implement it multiple times by calling the function. User-defined functions are especially useful because they save coding time and mitigate the risk of coding errors or bugs. You can use a function over and over again, so be sure to verify that your code works properly and correctly provides the output you want.

User-defined functions are composed of three key elements:

1. input objects, known as arguments
2. an operation or set of operations
3. a returned object or set of objects

Note that objects in the function are local to the function. Functions can take or return objects of any data type.

1.1 Defining functions

Defining functions is relatively straightforward. Use the assignment operator to name your function, add the arguments and the operations, and return the output object(s). You can use built-in functions within your user-defined functions. Indeed, many functions in R actually are functions of functions. Basic function structure and syntax are provided below. Note the use of curly braces.

```
# some_function <- function(arg1, arg2, ...){ statements / operations
# return(object) }
```

For example, you can define a function that adds two vectors.

```
rm(list = ls(all = TRUE))
```

```
addVectors <- function(a, b) {

    out.vec <- a + b

    return(out.vec)

}
```

1.2 Invoking user-defined functions

Once defined, your function will work like built-in R functions. Note that the objects you pass through a user-defined function do not have to have the same name as the argument—see the example below.

```
a <- 1:10
b <- -1:-10

addVectors(a, b)

##  [1] 0 0 0 0 0 0 0 0 0 0

x <- 1:10
y <- -1:-10

addVectors(x, y)

##  [1] 0 0 0 0 0 0 0 0 0 0

z <- addVectors(a = x, b = y)
```

Below is a much more complicated function for estimating cluster-robust standard errors. Many, many R users rely on this user-defined function (which they probably found on the internet).

```
cluster_se <- function(dat, fm, cluster) {
    require(sandwich, quietly = TRUE)
    require(lmtest, quietly = TRUE)
    M <- length(unique(cluster))
    N <- length(cluster)
    K <- fm$rank
    dfc <- (M/(M - 1)) * ((N - 1)/(N - K))
    uj <- apply(estfun(fm), 2, function(x) tapply(x, cluster, sum))
    vcovCL <- dfc * sandwich(fm, meat = crossprod(uj)/N)
    coeftest(fm, vcovCL)
}
```

An example with the municipal finance data...

First, load the data and fit an OLS model

```
setwd("/Users/patriciakirkland/Dropbox/Empiprical Reasoning Center/R Workshop")
load("muni_finance_data_cleaned.RData")

fit_3 <- lm(Total.Expenditure.PC ~ Total.Taxes.PC + Population + Census.Region,
    data = COG.fips)
summary(fit_3)

##
## Call:
## lm(formula = Total.Expenditure.PC ~ Total.Taxes.PC + Population +
##     Census.Region, data = COG.fips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8993 -0.5034 -0.2145  0.2300  8.6835
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.931e-01  2.244e-02  26.435  < 2e-16 ***
## Total.Taxes.PC          1.650e+00  2.407e-02  68.530  < 2e-16 ***
## Population              4.173e-07  3.096e-08  13.478  < 2e-16 ***
## Census.RegionNortheast  3.747e-01  3.688e-02  10.158  < 2e-16 ***
## Census.RegionSouth      3.094e-01  2.681e-02  11.543  < 2e-16 ***
## Census.RegionWest      -1.303e-01  2.688e-02  -4.849 1.27e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8901 on 7707 degrees of freedom
## Multiple R-squared:  0.4898,Adjusted R-squared:  0.4895
## F-statistic:  1480 on 5 and 7707 DF,  p-value: < 2.2e-16
```

To get heteroskedasticity-robust standard errors, you can use the built-in `coeftest()` function.

```
library(lmtest)
library(sandwich)

# heteroskedasticity-robust standard errors
coeftest(fit_3, vcov = vcovHC(fit_3, type = "HC1"))

##
## t test of coefficients:
##
##                           Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)             5.9308e-01  2.1572e-02 27.4937 < 2.2e-16 ***
## Total.Taxes.PC          1.6497e+00  3.4732e-02 47.4977 < 2.2e-16 ***
## Population              4.1734e-07  3.7661e-08 11.0814 < 2.2e-16 ***
## Census.RegionNortheast  3.7468e-01  4.0670e-02  9.2125 < 2.2e-16 ***
## Census.RegionSouth      3.0940e-01  2.6809e-02 11.5408 < 2.2e-16 ***
## Census.RegionWest      -1.3031e-01  2.0485e-02 -6.3611 2.118e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

However, you could also define your own function to estimate robust standard errors.

```
robust_se <- function(regmodel) {
    require(sandwich, quietly = TRUE)
    require(lmtest, quietly = TRUE)
    coeftest(regmodel, vcov = vcovHC(regmodel, type = "HC1"))
}
```

Examples

```
# robust SEs with our user-defined function
robust_se(fit_3)

##
## t test of coefficients:
##
##                           Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)             5.9308e-01  2.1572e-02 27.4937 < 2.2e-16 ***
## Total.Taxes.PC          1.6497e+00  3.4732e-02 47.4977 < 2.2e-16 ***
## Population              4.1734e-07  3.7661e-08 11.0814 < 2.2e-16 ***
## Census.RegionNortheast  3.7468e-01  4.0670e-02  9.2125 < 2.2e-16 ***
## Census.RegionSouth      3.0940e-01  2.6809e-02 11.5408 < 2.2e-16 ***
## Census.RegionWest      -1.3031e-01  2.0485e-02 -6.3611 2.118e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# cluster-robust standard errors
cluster_se(COG.fips, fit_3, COG.fips$fipsid)

##
## t test of coefficients:
##
##                           Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)             5.9308e-01  5.0197e-02 11.8152 < 2.2e-16 ***
## Total.Taxes.PC          1.6497e+00  7.8029e-02 21.1417 < 2.2e-16 ***
## Population              4.1734e-07  9.9522e-08  4.1934 2.779e-05 ***
## Census.RegionNortheast  3.7468e-01  9.9119e-02  3.7801  0.000158 ***
## Census.RegionSouth      3.0940e-01  6.5415e-02  4.7298 2.287e-06 ***
## Census.RegionWest      -1.3031e-01  5.0369e-02 -2.5870  0.009699 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.3 Source files

One very convenient feature of R is that you can use `source()` files so that you do not have to include every user-defined function you want to use in a script. If there are user-defined functions that you use often, you can create an R script that includes one or more user-defined function(s), and call the functions using the `source()` function. Many R users make one source file that they use in every script, while others make a source file for each project, assignment, or class.

Below is an example to illustrate. Start by clearing the workspace to remove the functions we defined above. You can use the `source()` function at the start of a script when you clear the workspace, set your working directory, and load packages.

```
## clear the workspace
rm(list = ls(all = TRUE))
```

```
setwd("/your/file/path")
```

```
source("ERC R Workshop Source.R")
```

For this example, start by loading data and running a regression model.

```
load("muni_finance_data_cleaned.RData")

fit_3 <- lm(Total.Expenditure.PC ~ Total.Taxes.PC + Population + Census.Region,
    data = COG.fips)
summary(fit_3)
```

```
##
## Call:
## lm(formula = Total.Expenditure.PC ~ Total.Taxes.PC + Population +
##     Census.Region, data = COG.fips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8993 -0.5034 -0.2145  0.2300  8.6835
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.931e-01  2.244e-02  26.435  < 2e-16 ***
## Total.Taxes.PC            1.650e+00  2.407e-02  68.530  < 2e-16 ***
## Population                4.173e-07  3.096e-08  13.478  < 2e-16 ***
## Census.RegionNortheast    3.747e-01  3.688e-02  10.158  < 2e-16 ***
## Census.RegionSouth        3.094e-01  2.681e-02  11.543  < 2e-16 ***
## Census.RegionWest        -1.303e-01  2.688e-02  -4.849 1.27e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8901 on 7707 degrees of freedom
## Multiple R-squared:  0.4898,Adjusted R-squared:  0.4895
## F-statistic:  1480 on 5 and 7707 DF,  p-value: < 2.2e-16
```

Call the user-defined function `robust_se` to get robust standard errors.

```
# robust SEs with our user-defined function (from source file)
robust_se(fit_3)

##
## t test of coefficients:
##
##                             Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)               5.9308e-01  2.1572e-02 27.4937 < 2.2e-16 ***
## Total.Taxes.PC            1.6497e+00  3.4732e-02 47.4977 < 2.2e-16 ***
## Population                4.1734e-07  3.7661e-08 11.0814 < 2.2e-16 ***
## Census.RegionNortheast    3.7468e-01  4.0670e-02  9.2125 < 2.2e-16 ***
## Census.RegionSouth        3.0940e-01  2.6809e-02 11.5408 < 2.2e-16 ***
## Census.RegionWest        -1.3031e-01  2.0485e-02 -6.3611 2.118e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Call the user-defined function `cluster_se` to get clustered standard errors.

```
# cluster-robust standard errors (from source file)
cluster_se(COG.fips, fit_3, COG.fips$fipsid)

##
## t test of coefficients:
##
```

```
##                            Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)               5.9308e-01  5.0197e-02 11.8152 < 2.2e-16 ***
## Total.Taxes.PC            1.6497e+00  7.8029e-02 21.1417 < 2.2e-16 ***
## Population                4.1734e-07  9.9522e-08  4.1934 2.779e-05 ***
## Census.RegionNortheast    3.7468e-01  9.9119e-02  3.7801  0.000158 ***
## Census.RegionSouth        3.0940e-01  6.5415e-02  4.7298 2.287e-06 ***
## Census.RegionWest        -1.3031e-01  5.0369e-02 -2.5870  0.009699 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2 Automating tasks— an example

This is just one example of how you might use R to automate tasks. Other tasks you could automate include querying databases (e.g. MS Access files), scraping web pages, and collecting data from social media sites. Of course you can also take a similar approach to running simulations, randomization inference, running multiple model specifications, and many other tasks.

```r
rm(list = ls(all = TRUE))

library(dplyr)
```

```r
setwd("/your/file/path")
```

In this example, you will use the `paste()` function and a loop to automate assembly of a dataset. This example uses Census of Governments municipal finance data. The historical database comes in a series of text files with three files for each year. One way to build a time-series cross-sectional dataset is to read in the files for each year and bind them by column. The next step is to append each year.

The code below automates this process.

```r
#### build annual .csv files from COG text files

directory <- "/your/file/path"  ## this is the file path for the COG text files.
year <- 2000:2007  ## years for the data files to assemble
```

```r
COG.muni <- data.frame()  ## an empty data frame to hold output from loop

for (j in year) {

    i <- substr(as.character(j), 3, 4)

    COG.a <- read.csv(paste0(directory, "/", "IndFin", formatC(i, width = 2,
        flag = "0"), "a", ".txt"))

    COG.b <- read.csv(paste0(directory, "/", "IndFin", formatC(i, width = 2,
        flag = "0"), "b", ".txt"))
    COG.c <- read.csv(paste0(directory, "/", "IndFin", formatC(i, width = 2,
        flag = "0"), "c", ".txt"))

    COGmerge <- left_join(COG.a, COG.b)

    COGmerge <- left_join(COGmerge, COG.c)

    COG.muni.temp <- subset(COGmerge, Type.Code == 2)

    COG.muni <- rbind(COG.muni, subset(COGmerge, Type.Code == 2))

}
```

3 R Markdown

R Markdown allows you to produce documents in a variety of formats (HTML, PDF, MS Word) using Markdown syntax and RStudio. With R Markdown, you can compose a document, analyze data, and produce plots and other results, in a single R script (.Rmd file). Open the file R_Faculty_Workshop_RMarkdown.Rmd to learn more about R Markdown. This file produces a sample document and includes basic formatting tips, along with a discussion of the benefits and advantages of R Markdown.

4 Apply Functions

The apply family of functions are part of R's `base` package. Apply functions provide a powerful option for performing functions on (subsets of) data. In some respects similar to loops, apply functions perform other functions on an input list. Compared to loops, however, apply functions typically require fewer lines of code and less processing time. For example, you might use an apply function to obtain means for a list of variables in a dataset.

Apply functions include `apply()`, `lapply()`, `sapply()`, `vapply()`, `mapply()`, `rapply()`, and `tapply()`. Which apply function you use depends on the structure of the data you have and the format of the output you need. Here, we will focus on `lapply()`, `sapply()`, and `tapply()`, which can be particularly helpful for data analysis.

- `tapply()` performs operations on vectors and returns a vector. It allows you to group vectors by one or more variable and perform a function on these subsets. The first argument is the vector (i.e., the variable) to which the function should be applied, next is the group variable, and finally, the function. See the example below.

  ```
  load("muni_finance_data_cleaned.RData")

  ## group or categorical variables typically should be factors
  COG.fips$Census.Region <- factor(COG.fips$Census.Region)

  ## check the frequency distribution (optional)
  table(COG.fips$Census.Region)

  ##
  ##   Midwest Northeast     South      West
  ##      2180       976      2292      2265
  ```

```r
## use tapply() to get group means (you can use other functions as well)
tapply(COG.fips$Total.Expenditure, COG.fips$Census.Region, mean, na.rm = TRUE)

##   Midwest Northeast     South      West
##  173238.6  873376.1  283071.4  261012.3
```

- `sapply()` takes a list, a data frame, or a subset of a data frame, performs a function on each element, and returns a vector. It can quickly aggregate or summarize data. The first argument is the data object, followed by the function to perform. See a basic example below as well as one option to create a data frame of summary statistics.

```r
## basic example
sapply(COG.fips[, 7:15], mean, na.rm = TRUE)

##            Total.Revenue             Total.Taxes            Property.Tax
##               321513.190              111366.684               51595.060
##       Total.Gen.Sales.Tax    Total.General.Charges       Total.Expenditure
##                19501.377               44264.597              320247.315
##    Total.Debt.Outstanding Total.Long.Term.Debt.Out      ST.Debt.End.of.Year
##               385658.160              381477.071                4181.089
```

```r
## create a data frame of summary statistics

## start by obtaining the statistics using sapply()
COG_means <- sapply(COG.fips[, 7:15], mean, na.rm = TRUE)
COG_medians <- sapply(COG.fips[, 7:15], median, na.rm = TRUE)
COG_stdevs <- sapply(COG.fips[, 7:15], sd, na.rm = TRUE)

## create a vector of variable names
COG_variable <- names(COG.fips[, 7:15])

## bind the vectors by columns
COG_summary <- cbind.data.frame(COG_variable, COG_means, COG_medians, COG_stdevs)

## remove the row names
row.names(COG_summary) <- NULL

COG_summary

##               COG_variable  COG_means COG_medians COG_stdevs
## 1            Total.Revenue 321513.190       90371 2435261.11
## 2              Total.Taxes 111366.684       33400  952374.74
## 3             Property.Tax  51595.060       16424  359111.87
## 4      Total.Gen.Sales.Tax  19501.377        1905  135776.02
## 5    Total.General.Charges  44264.597       13092  214744.23
## 6        Total.Expenditure 320247.315       89294 2432715.99
## 7   Total.Debt.Outstanding 385658.160       79736 2579973.69
## 8 Total.Long.Term.Debt.Out 381477.071       77516 2555077.53
## 9      ST.Debt.End.of.Year   4181.089           0   37442.44
```

- `lapply()` takes a list, a data frame, or a subset of a data frame, performs a function on each element of the input object, and returns a list (i.e., the data object is a list). This function can be very powerful and quite flexible. For example, you could run a regression model on a series of dependent variables.

```
## specify a list of variables
varlist <- c("Total.Expenditure.PC", "Total.Taxes.PC", "Total.Revenue.PC")

## run the same regression on multiple DVs
COG_models <- lapply(varlist, function(x) {

    lm(substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
        list(i = x)))

})
```

The above function returns a list, so you can use `lapply()` to perform the `summary()` function on it.

```
## to perform the summary() function
COG_results <- lapply(COG_models, summary)
```

The functions above can be combined into a single `lapply()` function

```
COG_models_results <- lapply(varlist, function(x) {

    summary(lm(substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
        list(i = x))))

})

COG_models_results

## [[1]]
##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7371 -0.6466 -0.2411  0.3637 13.1445
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.382e+00  2.442e-02  56.593   <2e-16 ***
## COG.fips$Population             8.742e-07  3.836e-08  22.791   <2e-16 ***
## COG.fips$Census.RegionNortheast 1.298e+00  4.356e-02  29.792   <2e-16 ***
## COG.fips$Census.RegionSouth     5.096e-01  3.380e-02  15.075   <2e-16 ***
## COG.fips$Census.RegionWest      5.639e-02  3.392e-02   1.663   0.0964 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.129 on 7708 degrees of freedom
## Multiple R-squared:  0.1789,Adjusted R-squared:  0.1785 
## F-statistic: 419.9 on 4 and 7708 DF,  p-value: < 2.2e-16
## 
## 
## [[2]]
## 
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region, 
##     list(i = x)))
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.0568 -0.2114 -0.0693  0.1217  8.1677 
## 
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                     4.784e-01  9.110e-03  52.510   <2e-16 ***
## COG.fips$Population             2.769e-07  1.431e-08  19.356   <2e-16 ***
## COG.fips$Census.RegionNortheast 5.595e-01  1.625e-02  34.437   <2e-16 ***
## COG.fips$Census.RegionSouth     1.213e-01  1.261e-02   9.623   <2e-16 ***
## COG.fips$Census.RegionWest      1.132e-01  1.265e-02   8.946   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.4212 on 7708 degrees of freedom
## Multiple R-squared:  0.178,Adjusted R-squared:  0.1776 
## F-statistic: 417.3 on 4 and 7708 DF,  p-value: < 2.2e-16
## 
## 
## [[3]]
## 
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region, 
##     list(i = x)))
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -2.7416 -0.6475 -0.2299  0.3620 15.0132 
## 
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                     1.374e+00  2.494e-02  55.117  < 2e-16 ***
## COG.fips$Population             8.644e-07  3.916e-08  22.074  < 2e-16 ***
## COG.fips$Census.RegionNortheast 1.308e+00  4.447e-02  29.413  < 2e-16 ***
## COG.fips$Census.RegionSouth     5.531e-01  3.451e-02  16.026  < 2e-16 ***
## COG.fips$Census.RegionWest      1.001e-01  3.463e-02   2.891  0.00385 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.153 on 7708 degrees of freedom
## Multiple R-squared:  0.1723,Adjusted R-squared:  0.1719
## F-statistic: 401.1 on 4 and 7708 DF,  p-value: < 2.2e-16
```

Note that the `lapply()` function returns a list, with each element indexed. Indexing for lists is different from indexing for other data objects you have seen. Indexing information is contained within double square brackets—e.g., [[1]] is element 1 of a list. You can also assign names to a list, much like you can a data frame. As with other data objects, you can also extract elements from a list.

```
## extract the first element
COG_models_results[[1]]

##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7371 -0.6466 -0.2411  0.3637 13.1445
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.382e+00  2.442e-02  56.593   <2e-16 ***
## COG.fips$Population             8.742e-07  3.836e-08  22.791   <2e-16 ***
## COG.fips$Census.RegionNortheast 1.298e+00  4.356e-02  29.792   <2e-16 ***
## COG.fips$Census.RegionSouth     5.096e-01  3.380e-02  15.075   <2e-16 ***
## COG.fips$Census.RegionWest      5.639e-02  3.392e-02   1.663   0.0964 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.129 on 7708 degrees of freedom
## Multiple R-squared:  0.1789,Adjusted R-squared:  0.1785
## F-statistic: 419.9 on 4 and 7708 DF,  p-value: < 2.2e-16
```

Add names to the elements in a list.

```
names(COG_models_results) <- varlist

COG_models_results

## $Total.Expenditure.PC
##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7371 -0.6466 -0.2411  0.3637 13.1445
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.382e+00  2.442e-02  56.593   <2e-16 ***
## COG.fips$Population             8.742e-07  3.836e-08  22.791   <2e-16 ***
## COG.fips$Census.RegionNortheast 1.298e+00  4.356e-02  29.792   <2e-16 ***
## COG.fips$Census.RegionSouth     5.096e-01  3.380e-02  15.075   <2e-16 ***
## COG.fips$Census.RegionWest      5.639e-02  3.392e-02   1.663   0.0964 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.129 on 7708 degrees of freedom
## Multiple R-squared:  0.1789,Adjusted R-squared:  0.1785
## F-statistic: 419.9 on 4 and 7708 DF,  p-value: < 2.2e-16
##
##
## $Total.Taxes.PC
##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0568 -0.2114 -0.0693  0.1217  8.1677
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     4.784e-01  9.110e-03  52.510   <2e-16 ***
## COG.fips$Population             2.769e-07  1.431e-08  19.356   <2e-16 ***
## COG.fips$Census.RegionNortheast 5.595e-01  1.625e-02  34.437   <2e-16 ***
## COG.fips$Census.RegionSouth     1.213e-01  1.261e-02   9.623   <2e-16 ***
## COG.fips$Census.RegionWest      1.132e-01  1.265e-02   8.946   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4212 on 7708 degrees of freedom
## Multiple R-squared:  0.178,Adjusted R-squared:  0.1776
## F-statistic: 417.3 on 4 and 7708 DF,  p-value: < 2.2e-16
##
##
## $Total.Revenue.PC
##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7416 -0.6475 -0.2299  0.3620 15.0132
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.374e+00  2.494e-02  55.117  < 2e-16 ***
## COG.fips$Population            8.644e-07  3.916e-08  22.074  < 2e-16 ***
## COG.fips$Census.RegionNortheast 1.308e+00  4.447e-02  29.413  < 2e-16 ***
## COG.fips$Census.RegionSouth     5.531e-01  3.451e-02  16.026  < 2e-16 ***
## COG.fips$Census.RegionWest      1.001e-01  3.463e-02   2.891  0.00385 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.153 on 7708 degrees of freedom
## Multiple R-squared:  0.1723,Adjusted R-squared:  0.1719
## F-statistic: 401.1 on 4 and 7708 DF,  p-value: < 2.2e-16
```

If you name the elements, you can extract them using the names you assigned.

```
COG_models_results[["Total.Taxes.PC"]]

##
## Call:
## lm(formula = substitute(COG.fips$i ~ COG.fips$Population + COG.fips$Census.Region,
##     list(i = x)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0568 -0.2114 -0.0693  0.1217  8.1677
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     4.784e-01  9.110e-03  52.510   <2e-16 ***
## COG.fips$Population             2.769e-07  1.431e-08  19.356   <2e-16 ***
## COG.fips$Census.RegionNortheast 5.595e-01  1.625e-02  34.437   <2e-16 ***
## COG.fips$Census.RegionSouth     1.213e-01  1.261e-02   9.623   <2e-16 ***
## COG.fips$Census.RegionWest      1.132e-01  1.265e-02   8.946   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4212 on 7708 degrees of freedom
## Multiple R-squared:  0.178,Adjusted R-squared:  0.1776
## F-statistic: 417.3 on 4 and 7708 DF,  p-value: < 2.2e-16
```

You can also extract more detailed information, depending on the contents of the list.

```
COG_models_results[["Total.Taxes.PC"]]$coefficients

##                                     Estimate   Std. Error   t value
## (Intercept)                     4.783876e-01 9.110476e-03 52.509619
## COG.fips$Population             2.769264e-07 1.430706e-08 19.355926
## COG.fips$Census.RegionNortheast 5.594948e-01 1.624708e-02 34.436637
## COG.fips$Census.RegionSouth     1.213299e-01 1.260777e-02  9.623423
## COG.fips$Census.RegionWest      1.131717e-01 1.265074e-02  8.945853
##                                      Pr(>|t|)
## (Intercept)                      0.000000e+00
## COG.fips$Population              1.530978e-81
## COG.fips$Census.RegionNortheast 7.427465e-242
## COG.fips$Census.RegionSouth      8.441271e-22
## COG.fips$Census.RegionWest       4.559873e-19

COG_models_results[["Total.Taxes.PC"]]$coefficients[, 1:2]

##                                      Estimate   Std. Error
## (Intercept)                      4.783876e-01 9.110476e-03
## COG.fips$Population              2.769264e-07 1.430706e-08
## COG.fips$Census.RegionNortheast 5.594948e-01 1.624708e-02
## COG.fips$Census.RegionSouth      1.213299e-01 1.260777e-02
## COG.fips$Census.RegionWest       1.131717e-01 1.265074e-02
```

5 Additional tools for data analysis

This section provides some basic information about additional tools for data analysis. You will note that this section is less detailed than other parts of the guide. Here, you will mainly find information about R packages and functions, a few examples of the methods, and suggested resources for further guidance.

5.1 ANOVA

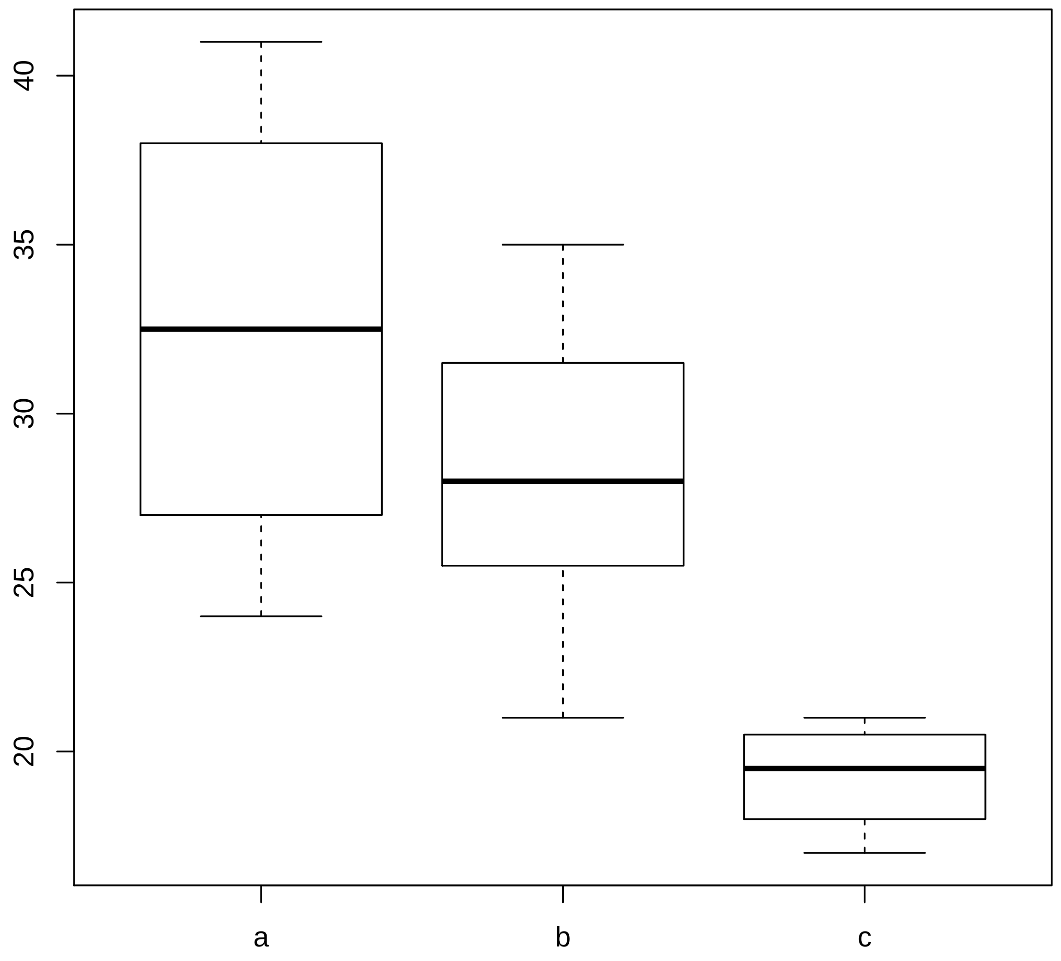
The `aov()` function in the `stats` is typically used for ANOVA models, though there are other functions that perform ANOVA analyses. For example, you can use the `oneway.test()` function (also part of the `stats` package) for a one-way ANOVA. Below you will find examples and code, which have been adapted from the Personality Project website (`https://personality-project.org/r/r.guide.html#anova` [accessed 11/16/16]). Note that the syntax for the `aov()` function is similar to that of `lm()` and other regression functions. This example code also shows one way to access data files from the internet.

A few additional sources for guidance on ANOVA in R

- `http://www.statmethods.net/stats/anova.html` —general info and basic code
- `https://ww2.coastal.edu/kingw/statistics/R-tutorials/repeated.html` —code and examples for single factor repeated measures ANOVA
- `http://www.sheffield.ac.uk/polopoly_fs/1.536444!/file/MASH_2way_ANOVA_in_R.pdf` —code and examples for two-way (between-groups) ANOVA in R
- **One-way ANOVA**

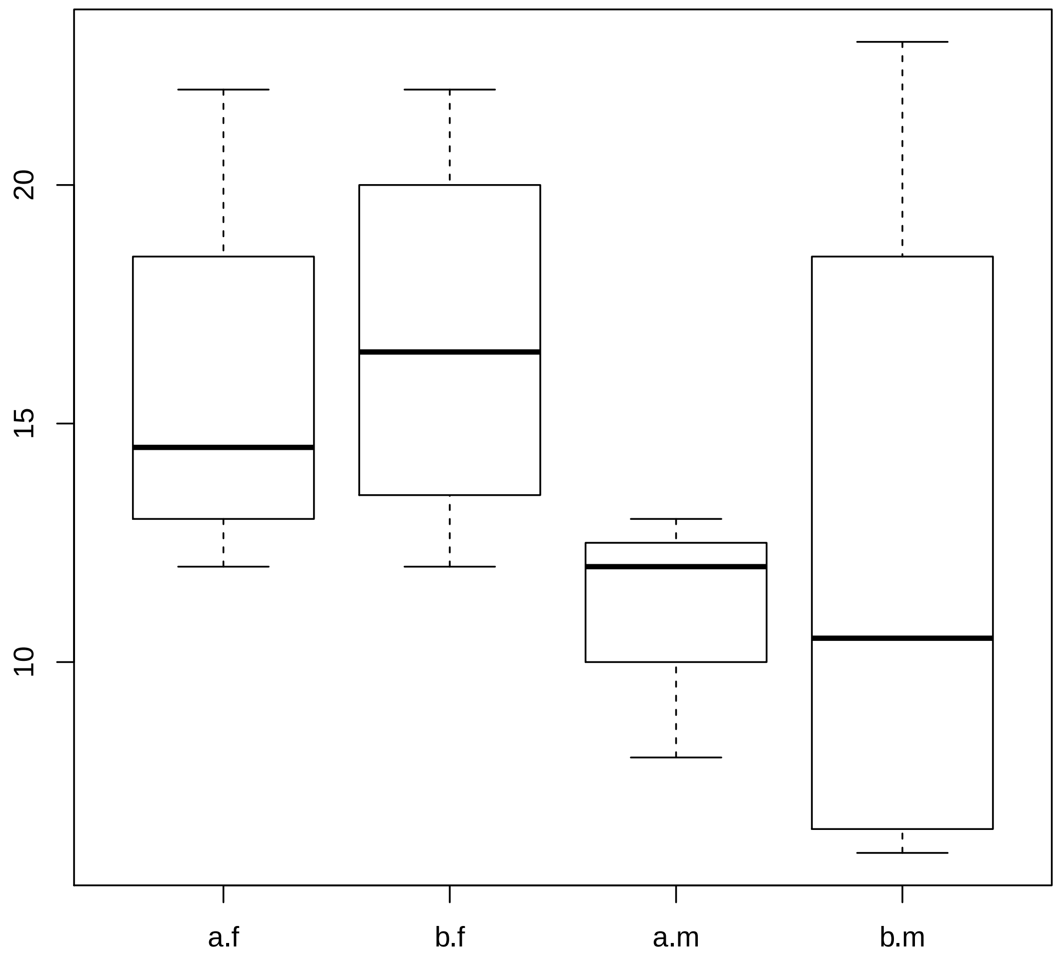
```
## Adapted from https://personality-project.org/r/r.guide.html\#anova

## load data
datafilename <- "http://personality-project.org/r/datasets/R.appendix1.data"
data.ex1 <- read.table(datafilename, header = T)

## check the class of the variables -- the IV should be a factor (group or
## categorical) variable
class(data.ex1$Dosage)

## [1] "factor"
```

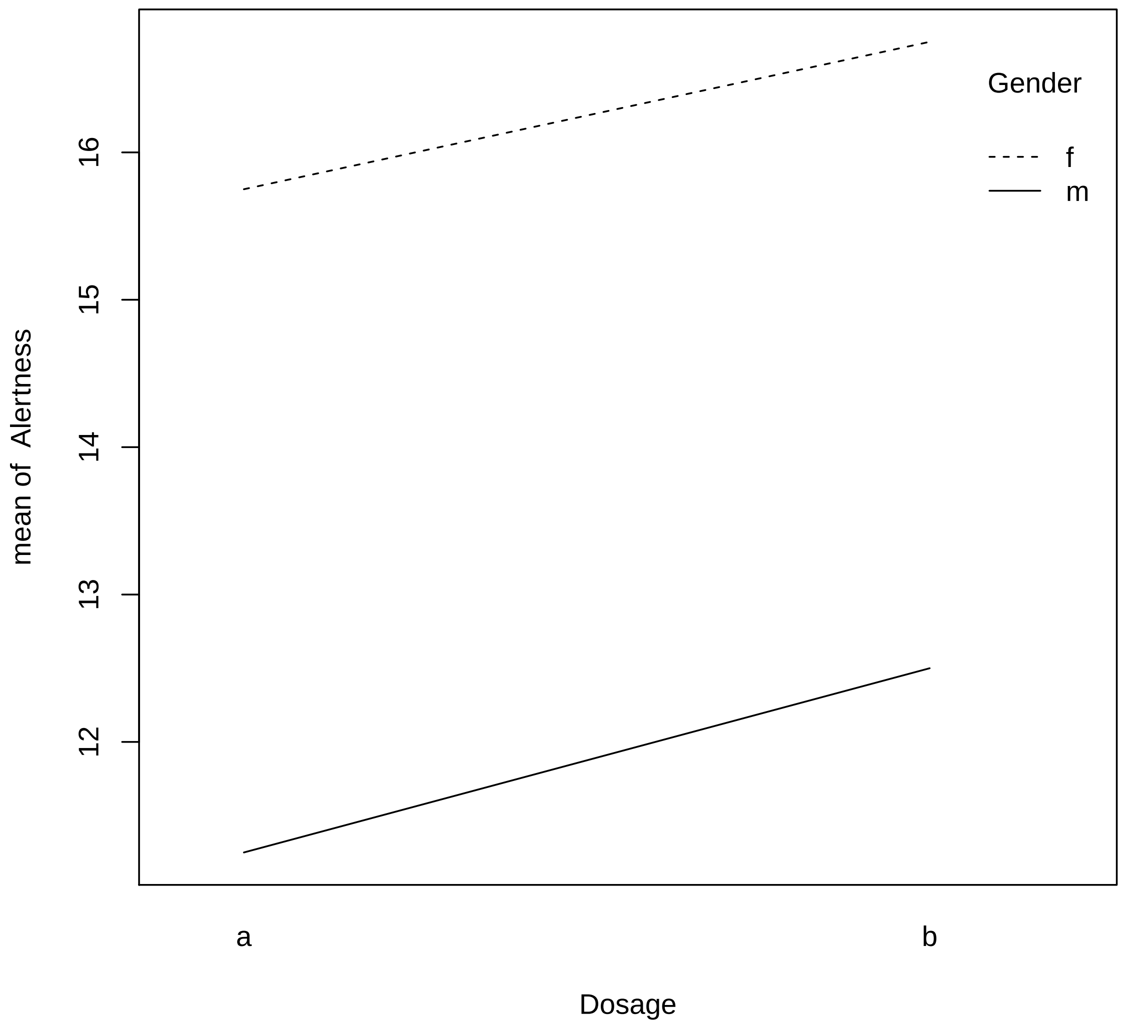
```
class(data.ex1$Alertness)

## [1] "integer"

## ANOVA model
aov.ex1 <- aov(Alertness ~ Dosage, data = data.ex1)
summary(aov.ex1)

##             Df Sum Sq Mean Sq F value  Pr(>F)
## Dosage       2  426.2  213.12   8.789 0.00298 **
## Residuals   15  363.8   24.25
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## obtain the means and the number of subjects per cell
model.tables(aov.ex1, "means")

## Tables of means
## Grand mean
##
## 27.66667
##
##  Dosage
##         a     b     c
##      32.5 28.25 19.25
## rep   6.0  8.00  4.00

## a basic boxplot -- you can add additional arguments to format the boxplot
## you could also do this in ggplot if you prefer

boxplot(Alertness ~ Dosage, data = data.ex1)
```



- **Two-way (between subjects) ANOVA**

```
## Adapted from https://personality-project.org/r/r.guide.html\#anova

## load data
datafilename = "http://personality-project.org/r/datasets/R.appendix2.data"
data.ex2 = read.table(datafilename, header = T)

## view data
data.ex2

##    Observation Gender Dosage Alertness
## 1            1      m      a         8
## 2            2      m      a        12
## 3            3      m      a        13
## 4           4      m      a        12
## 5           5      m      b         6
## 6           6      m      b         7
## 7           7      m      b        23
## 8           8      m      b        14
## 9           9      f      a        15
## 10         10      f      a        12
## 11         11      f      a        22
## 12         12      f      a        14
## 13         13      f      b        15
## 14         14      f      b        12
## 15         15      f      b        18
## 16         16      f      b        22

## ANOVA model
aov.ex2 <- aov(Alertness ~ Gender * Dosage, data = data.ex2)
summary(aov.ex2)

##                Df Sum Sq Mean Sq F value Pr(>F)
## Gender          1  76.56   76.56   2.952  0.111
## Dosage          1   5.06    5.06   0.195  0.666
## Gender:Dosage   1   0.06    0.06   0.002  0.962
## Residuals      12 311.25   25.94

## obtain the means and the number of subjects per cell
model.tables(aov.ex2, "means")

## Tables of means
## Grand mean
##
## 14.0625
##
##  Gender
## Gender
##      f      m
## 16.250 11.875
##
##  Dosage
## Dosage
##      a      b
## 13.500 14.625
##
##  Gender:Dosage
##       Dosage
## Gender a     b
##      f 15.75 16.75
##      m 11.25 12.50

## graphical summary using a boxplot
boxplot(Alertness ~ Dosage * Gender, data = data.ex2)

## another way to graph the means
with(data.ex2, interaction.plot(Dosage, Gender, Alertness))
```

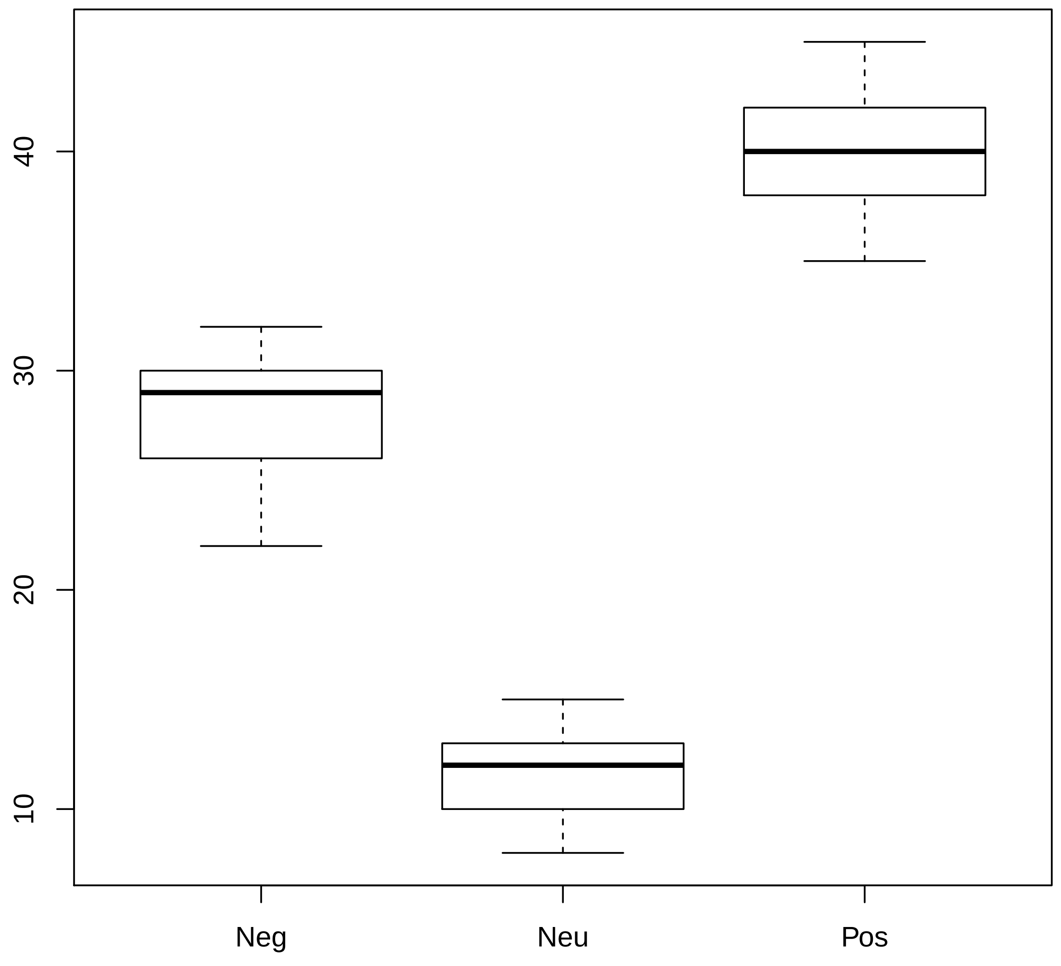

20
15
10
a.f
b.f
a.m
b.m



- **One-way repeated measures ANOVA**—note that data must be in long format (one response variable in one column). You will also add an error term, to indicate that treatments are nested within subjects. Use the error term with the subject variable and treatment variable (`Error([subject variable] / [treatment variable])`). See the example below.

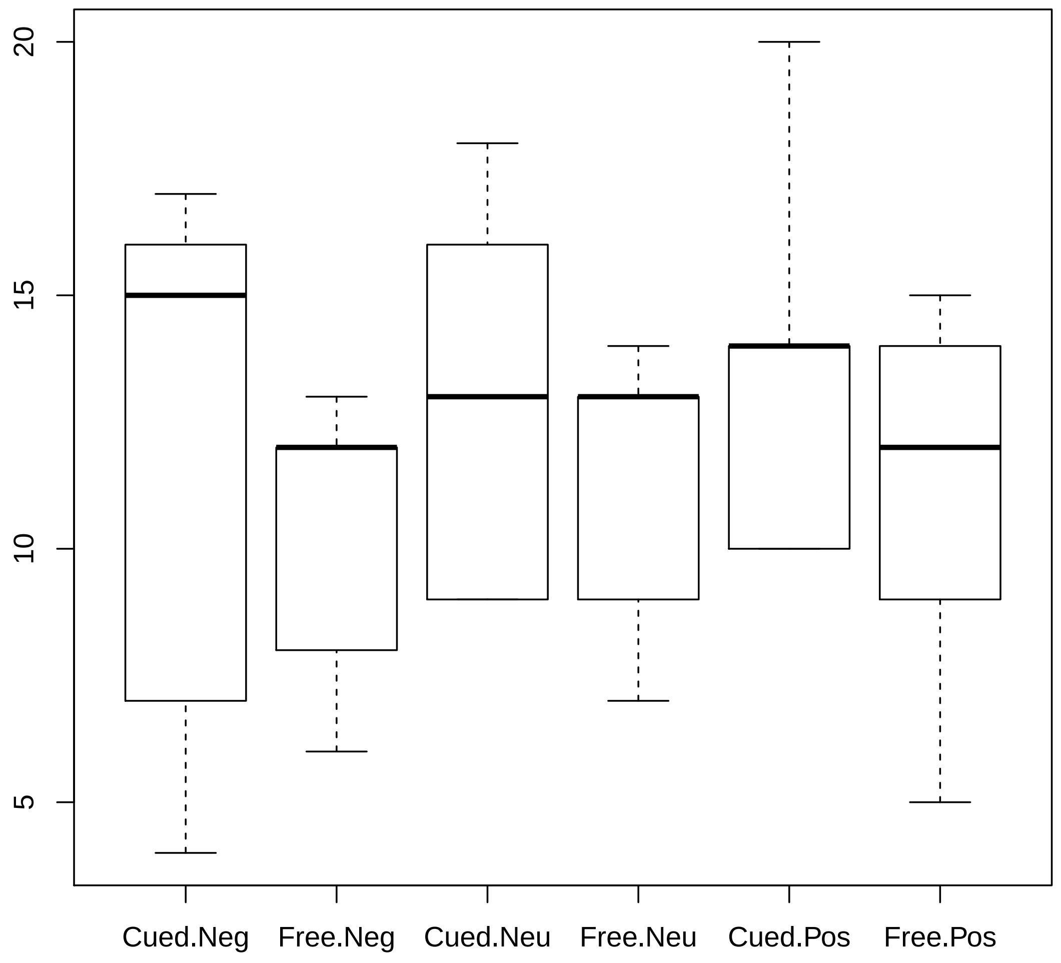
```
## Adapted from https://personality-project.org/r/r.guide.html\#anova

## load data
datafilename <- "http://personality-project.org/r/datasets/R.appendix3.data"
data.ex3 <- read.table(datafilename, header = T)



## view data -- note, data are in long format
data.ex3
```

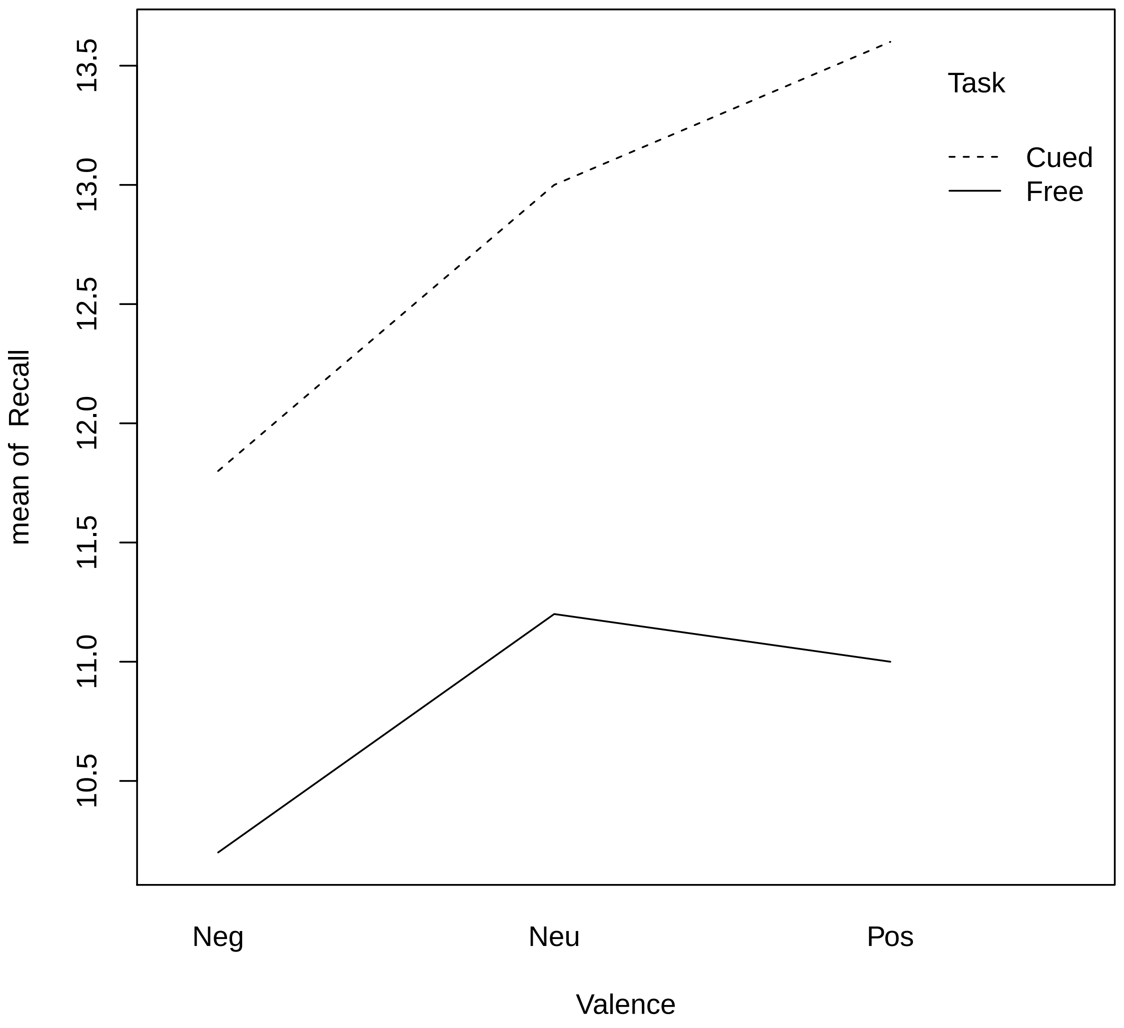
```
##    Observation Subject Valence Recall
## 1            1     Jim     Neg     32
## 2            2     Jim     Neu     15
## 3            3     Jim     Pos     45
## 4            4  Victor     Neg     30
## 5            5  Victor     Neu     13
## 6            6  Victor     Pos     40
## 7            7    Faye     Neg     26
## 8            8    Faye     Neu     12
## 9            9    Faye     Pos     42
## 10          10     Ron     Neg     22
## 11          11     Ron     Neu     10
## 12          12     Ron     Pos     38
## 13          13   Jason     Neg     29
## 14          14   Jason     Neu      8
## 15          15   Jason     Pos     35

## ANOVA model -- note the Error() term, indicates that the treatment
## (data.ex3$Valence) is nested within subjects (data.ex3Subject)
aov.ex3 <- aov(Recall ~ Valence + Error(Subject/Valence), data = data.ex3)
summary(aov.ex3)

##
## Error: Subject
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  4  105.1   26.27
##
## Error: Subject:Valence
##           Df Sum Sq Mean Sq F value   Pr(>F)
## Valence    2 2029.7  1014.9   189.1 1.84e-07 ***
## Residuals  8   42.9     5.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## obtain the means and the number of subjects per cell
model.tables(aov.ex3, "means")

## Tables of means
## Grand mean
##
## 26.46667
##
##  Valence
## Valence
##  Neg  Neu  Pos
## 27.8 11.6 40.0

## boxplot (also could use ggplot)
boxplot(Recall ~ Valence, data = data.ex3)
```



- **Two-way repeated measures ANOVA**

```
## Adapted from https://personality-project.org/r/r.guide.html\#anova

## load data
datafilename = "http://personality-project.org/r/datasets/R.appendix4.data"
data.ex4 = read.table(datafilename, header = T)

## ANOVA
aov.ex4 = aov(Recall ~ (Task * Valence) + Error(Subject/(Task * Valence)), data = data.ex4)
summary(aov.ex4)

##
## Error: Subject
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  4  349.1   87.28
##
## Error: Subject:Task
##           Df Sum Sq Mean Sq F value Pr(>F)
## Task       1  30.00  30.000   7.347 0.0535 .
## Residuals  4  16.33   4.083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Subject:Valence
##           Df Sum Sq Mean Sq F value Pr(>F)
## Valence    2   9.80   4.900   1.459  0.288
## Residuals  8  26.87   3.358
##
## Error: Subject:Task:Valence
##              Df Sum Sq Mean Sq F value Pr(>F)
## Task:Valence  2   1.40   0.700   0.291  0.755
## Residuals     8  19.27   2.408

## obtain the means and the number of subjects/cell
model.tables(aov.ex4, "means")

## Tables of means
## Grand mean
##
## 11.8
##
##  Task
## Task
## Cued Free
## 12.8 10.8
##
##  Valence
## Valence
##  Neg  Neu  Pos
## 11.0 12.1 12.3
##
##  Task:Valence
##       Valence
## Task   Neg  Neu  Pos
##   Cued 11.8 13.0 13.6
##   Free 10.2 11.2 11.0

## boxplot
boxplot(Recall ~ Task * Valence, data = data.ex4)

## interaction.plot -- another way to graph the interaction
with(data.ex4, interaction.plot(Valence, Task, Recall))
```


20
15
10
5
Cued.Neg
Free.Neg
Cued.Neu
Free.Neu
Cued.Pos
Free.Pos



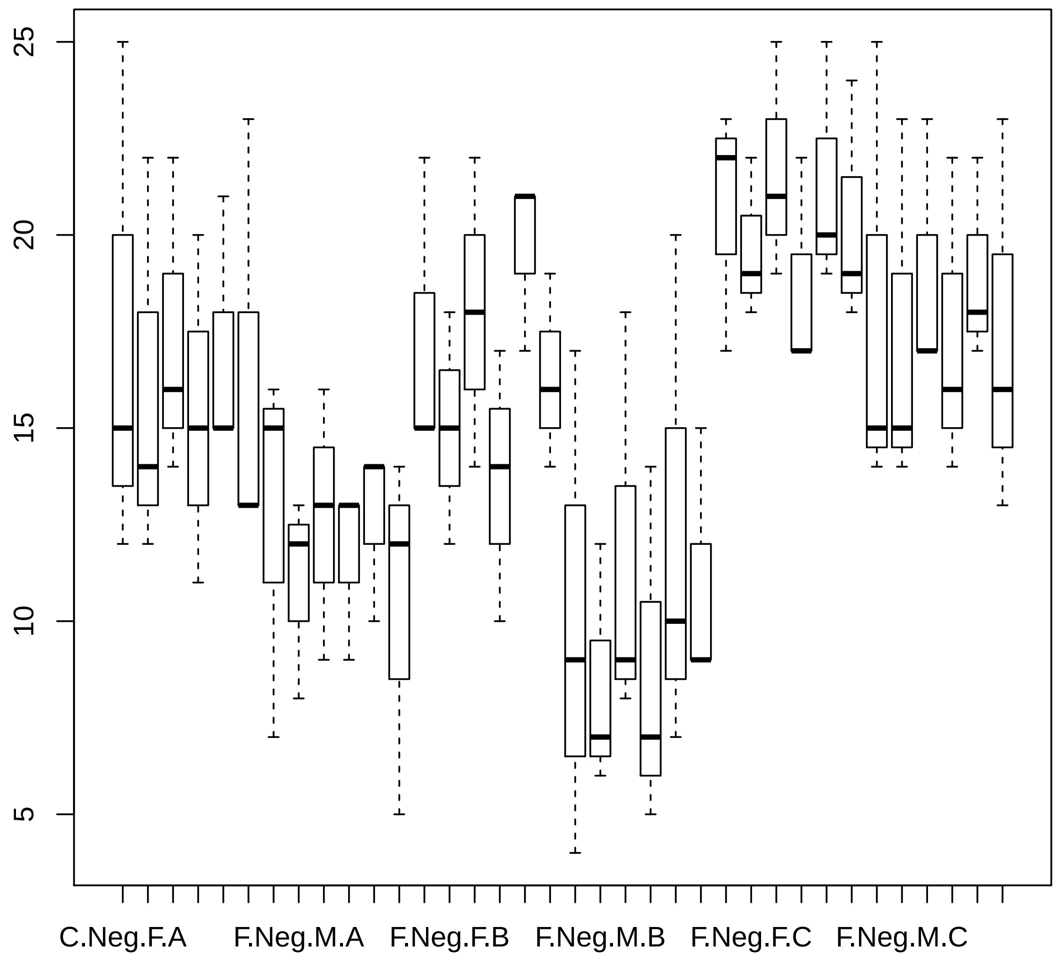
- **4-way ANOVA**: 2 repeated measures and two between-subjects

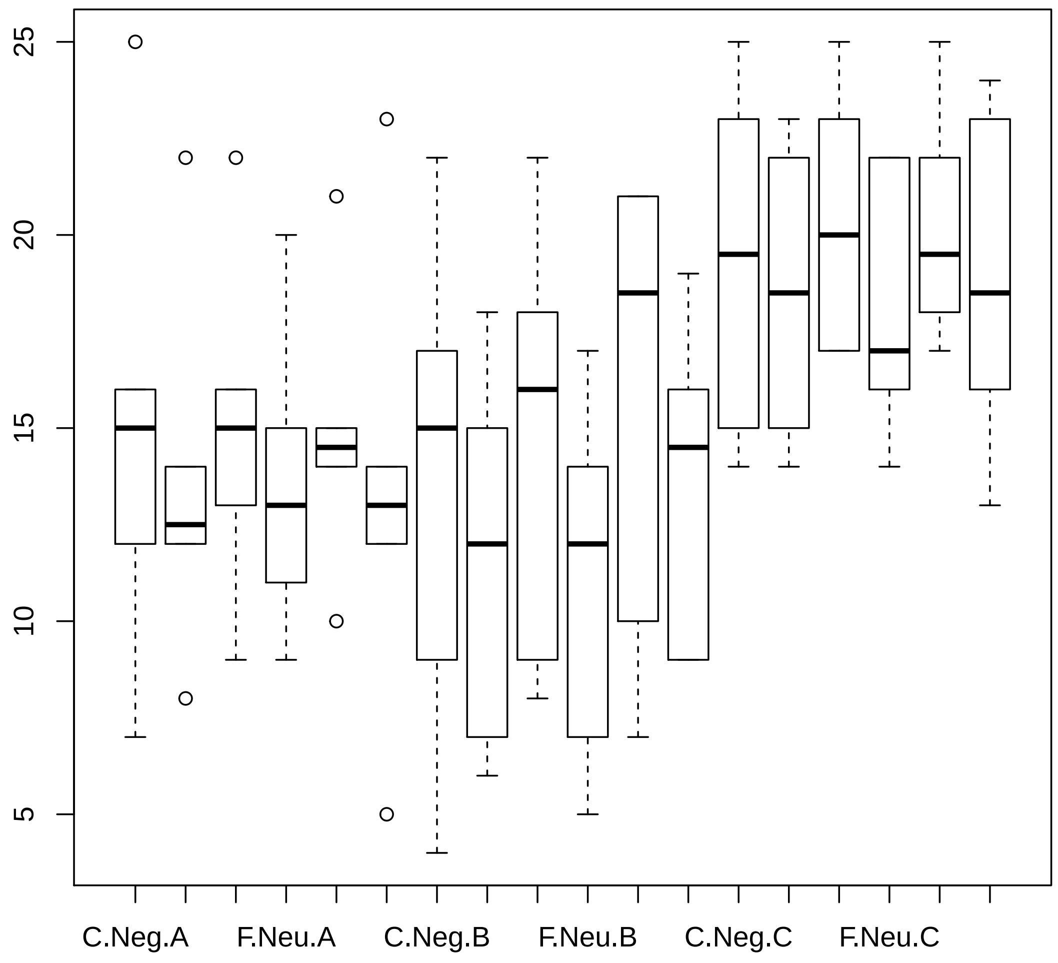
```
## Adapted from https://personality-project.org/r/r.guide.html\#anova

## load data
datafilename = "http://personality-project.org/r/datasets/R.appendix5.data"
data.ex5 = read.table(datafilename, header = T)

## ANOVA
aov.ex5 <- aov(Recall ~ (Task * Valence * Gender * Dosage) + Error(Subject/(Task *
    Valence)) + (Gender * Dosage), data = data.ex5)

summary(aov.ex5)

##
```

```
## Error: Subject
##                Df Sum Sq Mean Sq F value Pr(>F)
## Gender          1  542.3   542.3   5.685 0.0345 *
## Dosage          2  694.9   347.5   3.643 0.0580 .
## Gender:Dosage   2   70.8    35.4   0.371 0.6976
## Residuals      12 1144.6    95.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Subject:Task
##                     Df Sum Sq Mean Sq F value   Pr(>F)
## Task                 1  96.33   96.33  39.862 3.87e-05 ***
## Task:Gender          1   1.33    1.33   0.552    0.472
## Task:Dosage          2   8.17    4.08   1.690    0.226
## Task:Gender:Dosage   2   3.17    1.58   0.655    0.537
## Residuals           12  29.00    2.42
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Subject:Valence
##                        Df Sum Sq Mean Sq F value Pr(>F)
## Valence                 2  14.69   7.343   2.998 0.0688 .
## Valence:Gender          2   3.91   1.954   0.798 0.4619
## Valence:Dosage          4  20.26   5.065   2.068 0.1166
## Valence:Gender:Dosage   4   1.04   0.259   0.106 0.9793
## Residuals              24  58.78   2.449
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Subject:Task:Valence
##                             Df Sum Sq Mean Sq F value Pr(>F)
## Task:Valence                 2   5.39  2.6944   1.320  0.286
## Task:Valence:Gender          2   2.17  1.0833   0.531  0.595
## Task:Valence:Dosage          4   2.78  0.6944   0.340  0.848
## Task:Valence:Gender:Dosage   4   2.67  0.6667   0.327  0.857
## Residuals                   24  49.00  2.0417

## obtain the means and the number of subjects per cell
model.tables(aov.ex5, "means")

## Tables of means
## Grand mean
##
## 15.62963
##
##  Task
## Task
##      C      F
## 16.574 14.685
##
##  Valence
## Valence
##    Neg    Neu    Pos 
## 15.278 15.472 16.139 
## 
##  Gender 
## Gender
##      F      M 
## 17.870 13.389 
## 
##  Dosage 
## Dosage
##      A      B      C 
## 14.194 13.500 19.194 
## 
##  Task:Valence 
##     Valence
## Task Neg    Neu    Pos   
##    C 16.000 16.722 17.000
##    F 14.556 14.222 15.278
## 
##  Task:Gender 
##     Gender
## Task F      M     
##    C 18.926 14.222
##    F 16.815 12.556
## 
##  Valence:Gender 
##        Gender
## Valence F      M     
##     Neg 17.667 12.889
##     Neu 17.444 13.500
##     Pos 18.500 13.778
## 
##  Task:Dosage 
##     Dosage
## Task A      B      C     
##    C 14.944 14.833 19.944
##    F 13.444 12.167 18.444
## 
##  Valence:Dosage 
##        Dosage
## Valence A      B      C     
##     Neg 14.250 12.667 18.917
##     Neu 14.250 13.000 19.167
##     Pos 14.083 14.833 19.500
## 
##  Gender:Dosage 
##       Dosage
## Gender A      B      C     
##      F 16.556 16.667 20.389
##      M 11.833 10.333 18.000
## 
```

```
##  Task:Valence:Gender
## , , Gender = F
##
##     Valence
## Task Neg    Neu    Pos
##    C 18.444 19.000 19.333
##    F 16.889 15.889 17.667
##
## , , Gender = M
##
##     Valence
## Task Neg    Neu    Pos
##    C 13.556 14.444 14.667
##    F 12.222 12.556 12.889
##
##
##  Task:Valence:Dosage
## , , Dosage = A
##
##     Valence
## Task Neg    Neu    Pos
##    C 15.000 15.000 14.833
##    F 13.500 13.500 13.333
##
## , , Dosage = B
##
##     Valence
## Task Neg    Neu    Pos
##    C 13.667 14.833 16.000
##    F 11.667 11.167 13.667
##
## , , Dosage = C
##
##     Valence
## Task Neg    Neu    Pos
##    C 19.333 20.333 20.167
##    F 18.500 18.000 18.833
##
##
##  Task:Gender:Dosage
## , , Dosage = A
##
##     Gender
## Task F      M
##    C 17.222 12.667
##    F 15.889 11.000
##
## , , Dosage = B
##
##     Gender
## Task F      M
##    C 18.333 11.333
##    F 15.000  9.333
##
## , , Dosage = C
##
##     Gender
## Task F      M
##    C 21.222 18.667
##    F 19.556 17.333
##
##
##  Valence:Gender:Dosage
## , , Dosage = A
##
##        Gender
## Valence F      M
##     Neg 16.667 11.833
##     Neu 16.333 12.167
##     Pos 16.667 11.500
##
## , , Dosage = B
##
##        Gender
## Valence F      M
##     Neg 16.167  9.167
##     Neu 15.833 10.167
##     Pos 18.000 11.667
##
## , , Dosage = C
##
##        Gender
## Valence F      M
##     Neg 20.167 17.667
##     Neu 20.167 18.167
##     Pos 20.833 18.167
##
##
##  Task:Valence:Gender:Dosage
## , , Gender = F, Dosage = A
##
##     Valence
## Task Neg    Neu    Pos
##    C 17.333 17.333 17.000
##    F 16.000 15.333 16.333
##
## , , Gender = M, Dosage = A
##
##     Valence
## Task Neg    Neu    Pos
##    C 12.667 12.667 12.667
##    F 11.000 11.667 10.333
```

```
##
## , , Gender = F, Dosage = B
##
##     Valence
## Task Neg    Neu    Pos
##    C 17.333 18.000 19.667
##    F 15.000 13.667 16.333
##
## , , Gender = M, Dosage = B
##
##     Valence
## Task Neg    Neu    Pos
##    C 10.000 11.667 12.333
##    F  8.333  8.667 11.000
##
## , , Gender = F, Dosage = C
##
##     Valence
## Task Neg    Neu    Pos
##    C 20.667 21.667 21.333
##    F 19.667 18.667 20.333
##
## , , Gender = M, Dosage = C
##
##     Valence
## Task Neg    Neu    Pos
##    C 18.000 19.000 19.000
##    F 17.333 17.333 17.333

# graphical summary of means of the 36 cells
boxplot(Recall ~ Task * Valence * Gender * Dosage, data = data.ex5)

# graphical summary of means of 18 cells
boxplot(Recall ~ Task * Valence * Dosage, data = data.ex5)
```


25
20
15
10
5
C.Neg.F.A
F.Neg.M.A
F.Neg.F.B
F.Neg.M.B
F.Neg.F.C
F.Neg.M.C



5.2 Structural Equation Models

In this section, you will find some code and an example of structural equation modeling with R. Here, we use the `lavaan` package. You can also use the `SEM` package for structural equation models. The example below uses one of R's built-in datasets, so you may need to load the `datasets` package. In this example, you will use the `semPaths()` function to plot a path diagram. The `semPaths()` function requires the `semPlot` and `qgraph` packages.

Some resources for SEM in R

- `https://www.r-bloggers.com/first-steps-with-structural-equation-modeling/` – presentation on SEM in R by Grace Charles (Davis R Users' Group)
- `http://lavaan.ugent.be/tutorial/sem.html` – link to The lavaan Project
- `http://personality-project.org/r/tutorials/summerschool.14/rosseel_sem_intro.pdf` – link to presentation by package creator

- https://dornsife.usc.edu/assets/sites/210/docs/GC3/lavaan_tutorial.pdf – lavaan tutorial with examples by package author
- http://web.stanford.edu/class/psych253/section/section_4/section4.html – another helpful discussion of SEM in R with the lavaan package
- http://socserv.mcmaster.ca/jfox/Misc/sem/SEM-paper.pdf – resources for SEM using the SEM package in R

A few notes about specifying SEMs using the `lavaan` package

- measurement model equations are "latent" and represented by =∽
- regressions are indicated by ∽
- residual correlations (in this case because they represent different years of the same measurement) are represented by ∽∽

```
## Adapted from a presentation by Grace Charles, presented at Davis R Users'
## Group on May 15, 2015 adapted from Jim Grace's SEM workshop and Lavaan
## tutorials

rm(list = ls())

## load packages
library(lavaan)

## This is lavaan 0.5-22
## lavaan is BETA software!  Please report any bugs.

library(semPlot)
library(qgraph)

data(PoliticalDemocracy)


names(PoliticalDemocracy) <- c("free_press_1960", "free_oppo_1960", "fair_elect_1960",
    "leg_effective_1960", "free_press_1965", "free_oppo_1965", "fair_elect_1965",
    "leg_effect_1965", "GNP_pc_1960", "energy_pc_1960", "labor_force_pct_industry_1960")


## the measurement model equations are 'latent' and represented by =~
## regressions are indicated by ~ residual correlations (in this case because
## they represent different years of the same measurement) are represented by
## ~~

model <- "
# measurement model
industrialization_1960 =~ GNP_pc_1960 + energy_pc_1960 + labor_force_pct_industry_1960
democracy_1960 =~ free_press_1960 + free_oppo_1960 + fair_elect_1960 + leg_effective_1960
democracy_1965 =~ free_press_1965 + free_oppo_1965 + fair_elect_1965 + leg_effect_1965

# regressions
democracy_1960 ~ industrialization_1960
```

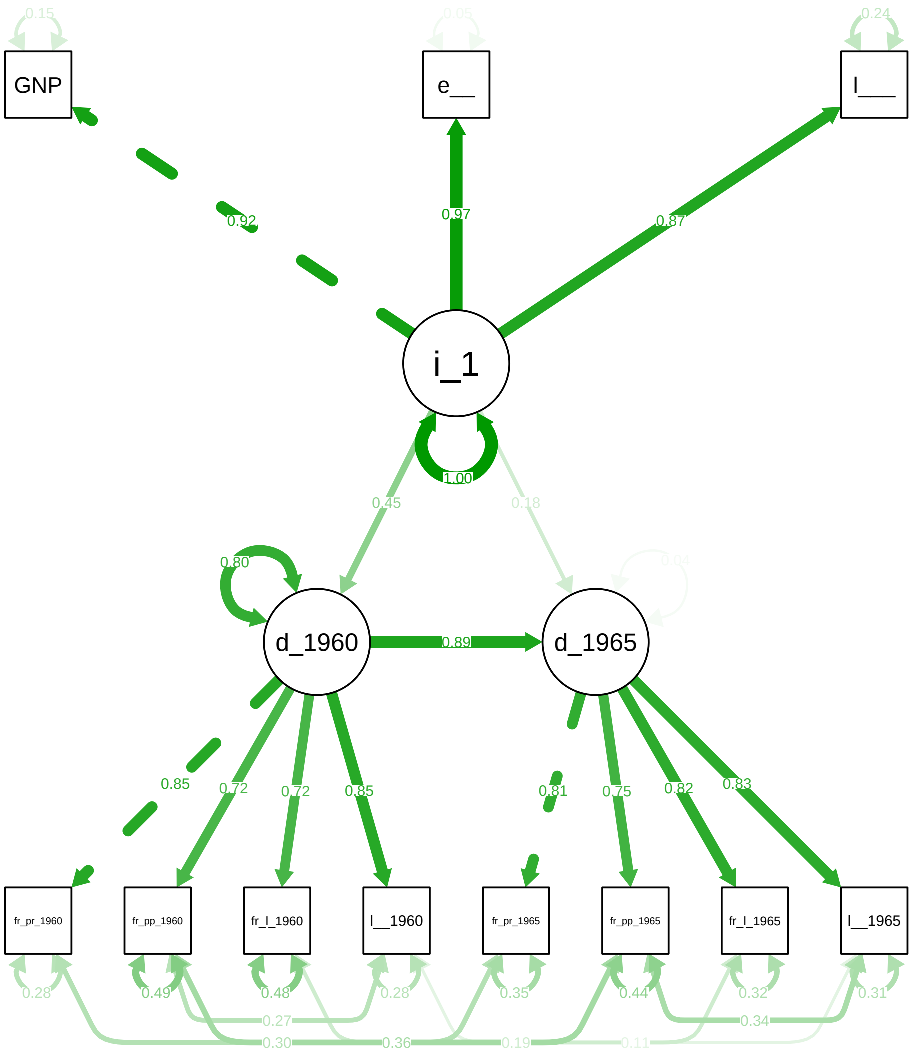
```
democracy_1965 ~ industrialization_1960 + democracy_1960

# residual correlations
free_press_1960 ~~ free_press_1965
free_oppo_1960 ~~ leg_effective_1960
free_oppo_1960 ~~ free_oppo_1965
fair_elect_1960 ~~ fair_elect_1965
leg_effective_1960 ~~ leg_effect_1965
free_oppo_1965 ~~ leg_effect_1965
"


# fit your SEM
fit <- sem(model, data = PoliticalDemocracy)
```

```
## Found more than one class "Model" in cache; using the first, from namespace 'lavaan'
```

```
# summarize results
summary(fit, standardized = TRUE, rsq = T)
```

```
##     0.973
##     0.872
##
##     0.850
##     0.717
##     0.722
##     0.846
##
##     0.808
##     0.746
##     0.824
##     0.828
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   democracy_1960 ~
##     indstrlzt_1960    1.483    0.399    3.715    0.000    0.447    0.447
##   democracy_1965 ~
##     indstrlzt_1960    0.572    0.221    2.586    0.010    0.182    0.182
##     democracy_1960    0.837    0.098    8.514    0.000    0.885    0.885
##
## Covariances:
##                         Estimate  Std.Err  z-value  P(>|z|)   Std.lv
##  .free_press_1960 ~~
##    .free_prss_1965         0.624    0.358    1.741    0.082    0.624
##  .free_oppo_1960 ~~
##    .leg_ffctv_1960         1.313    0.702    1.871    0.061    1.313
##    .free_oppo_1965         2.153    0.734    2.934    0.003    2.153
##  .fair_elect_1960 ~~
##    .fair_elct_1965         0.795    0.608    1.308    0.191    0.795
##  .leg_effective_1960 ~~
##    .leg_effct_1965         0.348    0.442    0.787    0.431    0.348
##  .free_oppo_1965 ~~
##    .leg_effct_1965         1.356    0.568    2.386    0.017    1.356
##   Std.all
##
##     0.296
##
##     0.273
##     0.356
##
##     0.191
##
##     0.109
##
##     0.338
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .GNP_pc_1960       0.082    0.019    4.184    0.000    0.082    0.154
##    .energy_pc_1960    0.120    0.070    1.718    0.086    0.120    0.053
##    .lbr_frc___1960    0.467    0.090    5.177    0.000    0.467    0.239
##    .free_prss_1960    1.891    0.444    4.256    0.000    1.891    0.277
##    .free_oppo_1960    7.373    1.374    5.366    0.000    7.373    0.486
##    .fair_elct_1960    5.067    0.952    5.324    0.000    5.067    0.478
##    .leg_ffctv_1960    3.148    0.739    4.261    0.000    3.148    0.285
##    .free_prss_1965    2.351    0.480    4.895    0.000    2.351    0.347
##    .free_oppo_1965    4.954    0.914    5.419    0.000    4.954    0.443
##    .fair_elct_1965    3.431    0.713    4.814    0.000    3.431    0.322
##    .leg_effct_1965    3.254    0.695    4.685    0.000    3.254    0.315
##     indstrlzt_1960    0.448    0.087    5.173    0.000    1.000    1.000
##    .democracy_1960    3.956    0.921    4.295    0.000    0.800    0.800
##    .democracy_1965    0.172    0.215    0.803    0.422    0.039    0.039
##
## R-Square:
##                    Estimate
##     GNP_pc_1960       0.846
##     energy_pc_1960    0.947
##     lbr_frc___1960    0.761
##     free_prss_1960    0.723
##     free_oppo_1960    0.514
##     fair_elct_1960    0.522
##     leg_ffctv_1960    0.715
##     free_prss_1965    0.653
##     free_oppo_1965    0.557
##     fair_elct_1965    0.678
##     leg_effct_1965    0.685
##     democracy_1960    0.200
##     democracy_1965    0.961

## you may see the following message: Found more than one class 'Model' in
## cache; using the first, from namespace âĂŹlavaanâĂŹ this is the result of a
## conflict between packages and does not indicate that there is a problem
## with the model you can find a discussion of this here if you'd like more
## info: https://groups.google.com/forum/#!topic/lavaan/fyAaGZws12g
```

```
## plot results using semPaths function in qgraph
semPaths(fit, "std", edge.label.cex = 0.5, curvePivot = TRUE, layout = "tree")
```



```
## check to see if you missed anything. High mi values suggest that there is
## a path that you missed.
modindices(fit)
```

```
##                         lhs op                   rhs    mi
## 35        industrialization_1960 =~      free_press_1960 2.286
## 36        industrialization_1960 =~       free_oppo_1960 1.519
## 37        industrialization_1960 =~      fair_elect_1960 0.035
## 38        industrialization_1960 =~   leg_effective_1960 4.796
## 39        industrialization_1960 =~      free_press_1965 4.456
## 40        industrialization_1960 =~       free_oppo_1965 0.116
## 41        industrialization_1960 =~      fair_elect_1965 2.046
## 42        industrialization_1960 =~      leg_effect_1965 0.057
## 43               democracy_1960 =~         GNP_pc_1960 1.611
## 44                democracy_1960 =~                energy_pc_1960 0.292
## 45                democracy_1960 =~ labor_force_pct_industry_1960 0.644
## 46                democracy_1960 =~               free_press_1965 2.687
## 47                democracy_1960 =~                free_oppo_1965 0.635
## 48                democracy_1960 =~               fair_elect_1965 0.581
## 49                democracy_1960 =~               leg_effect_1965 0.000
## 50                democracy_1965 =~                    GNP_pc_1960 1.501
## 51                democracy_1965 =~                energy_pc_1960 0.279
## 52                democracy_1965 =~ labor_force_pct_industry_1960 0.587
## 53                democracy_1965 =~               free_press_1960 2.178
## 54                democracy_1965 =~                free_oppo_1960 0.354
## 55                democracy_1965 =~               fair_elect_1960 0.227
## 56                democracy_1965 =~            leg_effective_1960 4.260
## 57                   GNP_pc_1960 ~~                energy_pc_1960 0.238
## 58                   GNP_pc_1960 ~~ labor_force_pct_industry_1960 0.148
## 59                   GNP_pc_1960 ~~               free_press_1960 1.284
## 60                   GNP_pc_1960 ~~                free_oppo_1960 3.040
## 61                   GNP_pc_1960 ~~               fair_elect_1960 0.105
## 62                   GNP_pc_1960 ~~            leg_effective_1960 0.945
## 63                   GNP_pc_1960 ~~               free_press_1965 1.378
## 64                   GNP_pc_1960 ~~                free_oppo_1965 0.018
## 65                   GNP_pc_1960 ~~               fair_elect_1965 1.504
## 66                   GNP_pc_1960 ~~               leg_effect_1965 0.001
## 67                energy_pc_1960 ~~ labor_force_pct_industry_1960 0.698
## 68                energy_pc_1960 ~~               free_press_1960 1.044
## 69                energy_pc_1960 ~~                free_oppo_1960 0.896
## 70                energy_pc_1960 ~~               fair_elect_1960 0.356
## 71                energy_pc_1960 ~~            leg_effective_1960 0.213
## 72                energy_pc_1960 ~~               free_press_1965 0.020
## 73                energy_pc_1960 ~~                free_oppo_1965 0.476
## 74                energy_pc_1960 ~~               fair_elect_1965 0.005
## 75                energy_pc_1960 ~~               leg_effect_1965 0.295
## 76  labor_force_pct_industry_1960 ~~               free_press_1960 0.554
## 77  labor_force_pct_industry_1960 ~~                free_oppo_1960 0.077
## 78  labor_force_pct_industry_1960 ~~               fair_elect_1960 0.829
## 79  labor_force_pct_industry_1960 ~~            leg_effective_1960 0.691
## 80  labor_force_pct_industry_1960 ~~               free_press_1965 0.142
## 81  labor_force_pct_industry_1960 ~~                free_oppo_1965 1.336
## 82  labor_force_pct_industry_1960 ~~               fair_elect_1965 0.403
## 83  labor_force_pct_industry_1960 ~~               leg_effect_1965 0.998
## 84               free_press_1960 ~~                free_oppo_1960 0.254
## 85               free_press_1960 ~~               fair_elect_1960 3.771
## 86               free_press_1960 ~~            leg_effective_1960 0.636
## 87               free_press_1960 ~~                free_oppo_1965 2.080
## 88               free_press_1960 ~~               fair_elect_1965 0.464
## 89               free_press_1960 ~~               leg_effect_1965 0.206
## 90                free_oppo_1960 ~~               fair_elect_1960 0.814
## 91                free_oppo_1960 ~~               free_press_1965 0.111
## 92                free_oppo_1960 ~~               fair_elect_1965 0.632
## 93                free_oppo_1960 ~~               leg_effect_1965 0.792
## 94               fair_elect_1960 ~~            leg_effective_1960 0.169
## 95              fair_elect_1960 ~~               free_press_1965 0.003
## 96              fair_elect_1960 ~~                free_oppo_1965 1.301
## 97              fair_elect_1960 ~~               leg_effect_1965 0.695
## 98           leg_effective_1960 ~~               free_press_1965 0.013
## 99           leg_effective_1960 ~~                free_oppo_1965 0.639
## 100          leg_effective_1960 ~~               fair_elect_1965 0.757
## 101             free_press_1965 ~~                free_oppo_1965 0.810
## 102             free_press_1965 ~~               fair_elect_1965 0.250
## 103             free_press_1965 ~~               leg_effect_1965 0.512
## 104              free_oppo_1965 ~~               fair_elect_1965 0.152
## 105             fair_elect_1965 ~~               leg_effect_1965 2.193
##        epc sepc.lv sepc.all sepc.nox
## 35  -0.527  -0.353   -0.135   -0.135
## 36  -0.613  -0.411   -0.105   -0.105
## 37   0.091   0.061    0.019    0.019
## 38   0.862   0.577    0.174    0.174
## 39   0.835   0.559    0.215    0.215
## 40  -0.152  -0.102   -0.030   -0.030
## 41  -0.708  -0.474   -0.145   -0.145
## 42  -0.098  -0.066   -0.020   -0.020
## 43   0.026   0.058    0.080    0.080
## 44  -0.021  -0.047   -0.032   -0.032
## 45  -0.037  -0.082   -0.059   -0.059
## 46  -0.954  -2.122   -0.815   -0.815
## 47   0.507   1.126    0.337    0.337
## 48   0.543   1.206    0.369    0.369
## 49  -0.012  -0.026   -0.008   -0.008
## 50   0.029   0.062    0.085    0.085
## 51  -0.024  -0.051   -0.034   -0.034
## 52  -0.041  -0.086   -0.062   -0.062
## 53  -0.945  -1.987   -0.760   -0.760
## 54  -0.505  -1.062   -0.273   -0.273
## 55  -0.395  -0.832   -0.255   -0.255
## 56   1.420   2.986    0.898    0.898
## 57  -0.052  -0.052   -0.047   -0.047
## 58  -0.025  -0.025   -0.024   -0.024
## 59   0.061   0.061    0.032    0.032
## 60  -0.155  -0.155   -0.055   -0.055
## 61   0.027   0.027    0.012    0.012
## 62   0.065   0.065    0.027    0.027
## 63   0.067   0.067    0.035    0.035
## 64  -0.010  -0.010   -0.004   -0.004
## 65  -0.089  -0.089   -0.038   -0.038
## 66  -0.002  -0.002   -0.001   -0.001
## 67   0.132   0.132    0.063    0.063
## 68  -0.100  -0.100   -0.026   -0.026
## 69   0.152   0.152    0.026    0.026
## 70   0.092   0.092    0.019    0.019
## 71  -0.056  -0.056   -0.011   -0.011
## 72   0.015   0.015    0.004    0.004
## 73  -0.089  -0.089   -0.018   -0.018
## 74  -0.009  -0.009   -0.002   -0.002
## 75   0.064   0.064    0.013    0.013
## 76  -0.091  -0.091   -0.025   -0.025
## 77  -0.056  -0.056   -0.010   -0.010
## 78  -0.175  -0.175   -0.039   -0.039
## 79   0.127   0.127    0.027    0.027
## 80  -0.049  -0.049   -0.013   -0.013
## 81   0.188   0.188    0.040    0.040
## 82   0.105   0.105    0.023    0.023
## 83  -0.146  -0.146   -0.033   -0.033
## 84  -0.229  -0.229   -0.023   -0.023
## 85   0.849   0.849    0.100    0.100
## 86  -0.318  -0.318   -0.037   -0.037
## 87   0.503   0.503    0.058    0.058
## 88  -0.273  -0.273   -0.032   -0.032
## 89  -0.152  -0.152   -0.018   -0.018
## 90  -0.599  -0.599   -0.047   -0.047
## 91   0.148   0.148    0.015    0.015
## 92   0.457   0.457    0.036    0.036
## 93   0.613   0.613    0.049    0.049
## 94   0.218   0.218    0.020    0.020
## 95   0.023   0.023    0.003    0.003
## 96  -0.600  -0.600   -0.055   -0.055
## 97  -0.409  -0.409   -0.039   -0.039
## 98  -0.042  -0.042   -0.005   -0.005
## 99   0.468   0.468    0.042    0.042
## 100 -0.407  -0.407   -0.037   -0.037
## 101 -0.326  -0.326   -0.037   -0.037
## 102  0.203   0.203    0.024    0.024
## 103 -0.244  -0.244   -0.029   -0.029
## 104 -0.185  -0.185   -0.017   -0.017
## 105  0.672   0.672    0.064    0.064

## looks good
```

```
## can also look at variance tables
vartable(fit)

##                               name idx nobs    type exo user  mean    var
## 1                      GNP_pc_1960   9   75 numeric   0    0 5.054  0.537
## 2                   energy_pc_1960  10   75 numeric   0    0 4.792  2.282
## 3  labor_force_pct_industry_1960  11   75 numeric   0    0 3.558  1.976
## 4                  free_press_1960   1   75 numeric   0    0 5.465  6.879
## 5                   free_oppo_1960   2   75 numeric   0    0 4.256 15.580
## 6                  fair_elect_1960   3   75 numeric   0    0 6.563 10.764
## 7               leg_effective_1960   4   75 numeric   0    0 4.453 11.219
## 8                  free_press_1965   5   75 numeric   0    0 5.136  6.826
## 9                   free_oppo_1965   6   75 numeric   0    0 2.978 11.375
## 10                 fair_elect_1965   7   75 numeric   0    0 6.196 10.799
## 11                 leg_effect_1965   8   75 numeric   0    0 4.043 10.534
##    nlev lnam
## 1     0
## 2     0
## 3     0
## 4     0
## 5     0
## 6     0
## 7     0
## 8     0
## 9     0
## 10    0
## 11    0

## sometimes you get warnings about the scale of your variables Warning
## message: In getDataFull(data = data, group = group, group.label =
## group.label, : lavaan WARNING: some observed variances are (at least) a
## factor 100 times larger than others; please rescale

# in this case, all you have to do to make this error go away is rescale
# variables
```

```
# model comparison

# you can compare alternative pathway models using AIC, BIC, etc:

# create second alternative model

model2 <- "
# measurement model
industrialization_1960 =~ GNP_pc_1960 + energy_pc_1960 + labor_force_pct_industry_1960
democracy_1960 =~ free_press_1960 + free_oppo_1960 + fair_elect_1960 + leg_effective_1960
democracy_1965 =~ free_press_1965 + free_oppo_1965 + fair_elect_1965 + leg_effect_1965

# regressions
# leave ind60 out of regression
democracy_1960 ~ industrialization_1960
democracy_1965 ~ democracy_1960

# residual correlations
free_press_1960 ~~ free_press_1965
free_oppo_1960 ~~ leg_effective_1960
free_oppo_1960 ~~ free_oppo_1965
fair_elect_1960 ~~ fair_elect_1965
leg_effective_1960 ~~ leg_effect_1965
free_oppo_1965 ~~ leg_effect_1965
"


fit2 <- sem(model2, data = PoliticalDemocracy)
summary(fit2)

## lavaan (0.5-22) converged normally after  67 iterations
##
```

```
##   Number of observations                            75
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               44.120
##   Degrees of freedom                                36
##   P-value (Chi-square)                           0.166
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                            Estimate  Std.Err  z-value  P(>|z|)
##   industrialization_1960 =~
##     GNP_pc_1960               1.000
##     energy_pc_1960            2.177    0.139   15.699    0.000
##     lbr_frc___1960            1.817    0.152   11.978    0.000
##   democracy_1960 =~
##     free_prss_1960            1.000
##     free_oppo_1960            1.273    0.190    6.705    0.000
##     fair_elct_1960            1.072    0.158    6.772    0.000
##     leg_ffctv_1960            1.303    0.151    8.644    0.000
##   democracy_1965 =~
##     free_prss_1965            1.000
##     free_oppo_1965            1.209    0.175    6.910    0.000
##     fair_elct_1965            1.317    0.166    7.915    0.000
##     leg_effct_1965            1.297    0.165    7.852    0.000
##
## Regressions:
##                  Estimate  Std.Err  z-value  P(>|z|)
##   democracy_1960 ~
##     indstrlzt_1960    1.663    0.379    4.386    0.000
##   democracy_1965 ~
##     democracy_1960    0.936    0.106    8.825    0.000
##
## Covariances:
##                        Estimate  Std.Err  z-value  P(>|z|)
##  .free_press_1960 ~~
##    .free_prss_1965         0.581    0.363    1.600    0.110
##  .free_oppo_1960 ~~
##    .leg_ffctv_1960         1.380    0.693    1.993    0.046
##    .free_oppo_1965         2.131    0.735    2.901    0.004
##  .fair_elect_1960 ~~
##    .fair_elct_1965         0.798    0.611    1.307    0.191
##  .leg_effective_1960 ~~
##    .leg_effct_1965         0.324    0.438    0.741    0.459
##  .free_oppo_1965 ~~
##    .leg_effct_1965         1.297    0.570    2.274    0.023
##
## Variances:
```

```
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .GNP_pc_1960       0.081    0.020    4.126    0.000
##    .energy_pc_1960    0.122    0.071    1.725    0.085
##    .lbr_frc___1960    0.467    0.090    5.164    0.000
##    .free_prss_1960    2.074    0.450    4.603    0.000
##    .free_oppo_1960    7.574    1.386    5.463    0.000
##    .fair_elct_1960    5.228    0.967    5.405    0.000
##    .leg_ffctv_1960    3.067    0.711    4.316    0.000
##    .free_prss_1965    2.460    0.498    4.939    0.000
##    .free_oppo_1965    4.874    0.913    5.338    0.000
##    .fair_elct_1965    3.263    0.703    4.639    0.000
##    .leg_effct_1965    3.177    0.701    4.531    0.000
##     indstrlzt_1960    0.449    0.087    5.178    0.000
##    .democracy_1960    3.464    0.828    4.182    0.000
##    .democracy_1965    0.156    0.226    0.688    0.491

## AIC
AIC(fit, fit2)

##      df      AIC
## fit  31 3157.582
## fit2 30 3161.577
```