

Wednesday, Jan 19

Example: The *data frame* `trees` comes with R. We can see it if we just type `trees` at the prompt in the R console.

```
trees
```

	Girth	Height	Volume
1	8.3	70	10.3
2	8.6	65	10.3
3	8.8	63	10.2
4	10.5	72	16.4
5	10.7	81	18.8
6	10.8	83	19.7
7	11.0	66	15.6
8	11.0	75	18.2
9	11.1	80	22.6
10	11.2	75	19.9
11	11.3	79	24.2
12	11.4	76	21.0
13	11.4	76	21.4
14	11.7	69	21.3
15	12.0	75	19.1
16	12.9	74	22.2
17	12.9	85	33.8
18	13.3	86	27.4
19	13.7	71	25.7
20	13.8	64	24.9
21	14.0	78	34.5
22	14.2	80	31.7
23	14.5	74	36.3
24	16.0	72	38.3
25	16.3	77	42.6
26	17.3	81	55.4
27	17.5	82	55.7
28	17.9	80	58.3
29	18.0	80	51.5
30	18.0	80	51.0
31	20.6	87	77.0

Let's specify the model

$$E(V_i) = \beta_0 + \beta_1 g_i + \beta_2 h_i,$$

where V_i , g_i , and h_i are the volume, girth, and height from the i -th observation.

```
m <- lm(formula = Volume ~ Girth + Height, data = trees)
```

There's a lot to say here.

1. `lm` is a *function* to which we have provided two *arguments*: `formula` and `data`. Other arguments can be found using `args(lm)`, some of which we will use later.

```
args(lm)
```

```
function (formula, data, subset, weights, na.action, method = "qr",  
  model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,  
  contrasts = NULL, offset, ...)  
NULL
```

We do not need to name the arguments *if we provide them in the same order as they are expected*. For example, this would also work:

```
m <- lm(Volume ~ Girth + Height, trees)
```

Just about all functions that estimate a regression model expect **formula** to be the first argument. A common convention (and one that I use) is to name all arguments except the first:

```
m <- lm(Volume ~ Girth + Height, data = trees)
```

2. The **formula** argument is *symbolic*, not literal. It is a system for communicating with R the regression model you want to specify. The model formula `Volume ~ Girth + Height` implies the statistical model $E(V_i) = \beta_0 + \beta_1 g_i + \beta_2 h_i$.
3. We have *assigned* the output of this function to an *object* called `m` (R is an *object-oriented* programming language). Note that you can also use `=` instead of `<-`. We can apply other functions to this object. For example, `print(m)`.

```
print(m)
```

Call:

```
lm(formula = Volume ~ Girth + Height, data = trees)
```

Coefficients:

(Intercept)	Girth	Height
-57.9877	4.7082	0.3393

But if you just have `m` it will interpret that as `print(m)`.

```
m
```

Call:

```
lm(formula = Volume ~ Girth + Height, data = trees)
```

Coefficients:

(Intercept)	Girth	Height
-57.9877	4.7082	0.3393

A bit more information can be extracted by using the **summary** function.

```
summary(m)
```

Call:

```
lm(formula = Volume ~ Girth + Height, data = trees)
```

Residuals:

Min	1Q	Median	3Q	Max
-6.4065	-2.6493	-0.2876	2.2003	8.4847

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-57.9877	8.6382	-6.713	2.75e-07 ***
Girth	4.7082	0.2643	17.816	< 2e-16 ***

```
Height          0.3393      0.1302    2.607    0.0145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948,    Adjusted R-squared:  0.9442
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

In lecture I will often trim the output from `summary` using something like the following.

```
summary(m)$coefficients
```

```
              Estimate Std. Error  t value    Pr(>|t|)
(Intercept) -57.9876589   8.6382259 -6.712913 2.749507e-07
Girth         4.7081605   0.2642646  17.816084 8.223304e-17
Height        0.3392512   0.1301512   2.606594 1.449097e-02
```

This shows estimates of the parameters β_0 , β_1 , and β_2 , and some other information concerning inferences for those parameters (more on that later).

Example: Here are the data and a specified linear model for the study on dopamine activity in schizophrenics. The data are available in the **BSDA** package.

```
library(BSDA) # install with install.packages("BSDA")
head(Dopamine)
```

```
# A tibble: 6 x 2
  dbh group
<int> <fct>
1   104 nonpsychotic
2   105 nonpsychotic
3   112 nonpsychotic
4   116 nonpsychotic
5   130 nonpsychotic
6   145 nonpsychotic
```

```
tail(Dopamine)
```

```
# A tibble: 6 x 2
  dbh group
<int> <fct>
1   226 psychotic
2   245 psychotic
3   270 psychotic
4   275 psychotic
5   306 psychotic
```

6 320 psychotic

Also you can try `?Dopamine` or `help(Dopamine)` to see the help file (you can also do this for functions like `lm`). A `tibble` is another way that data are stored in R that is mostly interchangeable with data frames (they are effectively “enhanced” data frames).

Let’s specify a linear model using the `lm` function.

```
m <- lm(dbh ~ group, data = Dopamine)
summary(m)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	242.60000	15.41419	15.738750	8.320341e-14
groupnonpsychotic	-78.33333	19.89963	-3.936422	6.586761e-04

Note: The `grouppsychotic` tells us that an indicator variable was created for the level/category `psychotic` for the categorical variable `group`. How would we write this model *mathematically*?

Example: Here are the data from the anorexia study.

```
library(MASS) # install with install.packages("MASS")
head(anorexia)
```

	Treat	Prewt	Postwt	change
1	Cont	80.7	80.2	-0.5
2	Cont	89.4	80.1	-9.3
3	Cont	91.8	86.4	-5.4
4	Cont	74.0	86.3	12.3
5	Cont	78.1	76.1	-2.0
6	Cont	88.3	78.1	-10.2

```
summary(anorexia)
```

	Treat	Prewt	Postwt	change
Cont:26	Min.	:70.00	Min. : 71.30	Min. : -12.200
CBT :29	1st Qu.:	79.60	1st Qu. : 79.33	1st Qu. : -2.225
FT :17	Median :	82.30	Median : 84.05	Median : 1.650
	Mean :	82.41	Mean : 85.17	Mean : 2.764
	3rd Qu.:	86.00	3rd Qu. : 91.55	3rd Qu. : 9.100
	Max. :	94.90	Max. : 103.60	Max. : 21.500

Here we are going to create another variable `change` which is the change in weight. (Note: I redefined `change` from the original lecture as the post-weight minus the pre-weight so that a positive value indicates weight gain and a negative value indicates weight loss.)

```
anorexia$change <- anorexia$Postwt - anorexia$Prewt
head(anorexia)
```

	Treat	Prewt	Postwt	change
1	Cont	80.7	80.2	-0.5
2	Cont	89.4	80.1	-9.3
3	Cont	91.8	86.4	-5.4

4	Cont	74.0	86.3	12.3
5	Cont	78.1	76.1	-2.0
6	Cont	88.3	78.1	-10.2

Let's specify a linear model.

```
m <- lm(change ~ Treat, data = anorexia)
summary(m)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.450000	1.476449	-0.3047854	0.761447048
TreatCBT	3.456897	2.033297	1.7001438	0.093607652
TreatFT	7.714706	2.348163	3.2854224	0.001602338

What are the indicator variables? How would we write this model mathematically?

We can change the parameterization using `relevel`.

```
anorexia$Treat <- relevel(anorexia$Treat, ref = "Cont")
m <- lm(change ~ Treat, data = anorexia)
summary(m)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.450000	1.476449	-0.3047854	0.761447048
TreatCBT	3.456897	2.033297	1.7001438	0.093607652
TreatFT	7.714706	2.348163	3.2854224	0.001602338

What are the indicator variables? How would we write this model mathematically?

Note: We can change the parameterization for the model for the previous example, but we need to make `group` a *factor* because the `relevel` function will only work on variables that are factors (although the function `lm` automatically converts a character variable to a factor). We can see that it is not a factor (yet) from `summary` and also from the `is.factor` function.

```
summary(Dopamine)
```

```
      dbh      group
Min.   :104.0  psychotic   :10
1st Qu.:150.0  nonpsychotic:15
Median :200.0
Mean   :195.6
3rd Qu.:230.0
Max.   :320.0
```

```
is.factor(Dopamine$group)
```

```
[1] TRUE
```

Here's what happens if you try to use `relevel` with the `group` variable.

```
Dopamine$group <- relevel(Dopamine$group, ref = "psychotic")
```

Let's make `group` a factor and then change the parameterization.

```
Dopamine$group <- factor(Dopamine$group)
Dopamine$group <- relevel(Dopamine$group, ref = "psychotic")
```

Note that we could also have done this with one line of code.

```
Dopamine$group <- relevel(factor(Dopamine$group), ref = "psychotic")
```

Here's the reparameterized model.

```
m <- lm(dbh ~ group, data = Dopamine)
summary(m)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	242.60000	15.41419	15.738750	8.320341e-14
groupnonpsychotic	-78.33333	19.89963	-3.936422	6.586761e-04

What is this model?

Here's another parameterization. Including the term `-1` in the model formula argument suppresses the inclusion of the β_0 parameter (again, it is symbolic, not literal).¹

```
m <- lm(dbh ~ -1 + group, data = Dopamine)
summary(m)$coefficients
```

	Estimate	Std. Error	t value	Pr(> t)
--	----------	------------	---------	----------

¹When we say something like `dbh ~ group` this is actually translated as `dbh ~ 1 + group` where the “explanatory variable” of 1 effectively becomes the term β_0 . Including this term is the default even if we do not mention it explicitly. But if we do not want it then we “add” `-1` to the model formula to remove it.

```

grouppsychotic    242.6000    15.41419 15.73875 8.320341e-14
groupnonpsychotic 164.2667    12.58563 13.05192 4.058662e-12

```

Let's try that with the anorexia data.

```

m <- lm(change ~ -1 + Treat, data = anorexia)
summary(m)$coefficients

```

```

              Estimate Std. Error    t value    Pr(>|t|)
TreatCont -0.450000    1.476449  -0.3047854 0.7614470484
TreatCBT   3.006897    1.397996   2.1508614 0.0349919664
TreatFT    7.264706    1.825915   3.9786656 0.0001687833

```

What are these models?

Example: Sometimes we may also want so basic descriptive statistics. There are *many* ways to do this in R. One flexible approach is to use the **dplyr** package.

```

library(dplyr) # install with install.packages("dplyr") or install.packages("tidyverse")
Dopamine %>% group_by(group) %>% summarize(dbh = mean(dbh))

```

```

# A tibble: 2 x 2
  group      dbh
  <fct>    <dbl>
1 psychotic 243.
2 nonpsychotic 164.

```

```

anorexia %>% mutate(change = Prewt - Postwt) %>% group_by(Treat) %>%
  summarize(meanchange = mean(change), sdchange = sd(change), samplesize = n())

```

```

# A tibble: 3 x 4
  Treat meanchange sdchange samplesize
  <fct>    <dbl>    <dbl>    <int>
1 Cont     0.450     7.99      26
2 CBT    -3.01     7.31      29
3 FT     -7.26     7.16      17

```

The **dplyr** and **tidyr** packages are very useful for manipulating and summarizing data. They have a bit of a learning curve, but they are well worth learning. I will provide many examples of how they can be used.