Lab 4: Reproducing Sutter, 2009

Simon Halliday

Fall 2017

Loading required packages

Be sure to load the packages you require. We use Hadley Wickham's "Tidyverse" of packages.

1. We need `dplyr` to use `summarise` and `group_by`
2. We need `tidyr` for `gather`.
3. We also need `ggplot2` for `ggplot`.
4. All of these three are included in the call `library(tidyverse)`
5. We use panel models, for which we require `plm`.

```
library(tidyverse)
library(plm)
library(stargazer)
```

Data Import

We now need to import the data. You can download the csv of the file from Google drive here: https://drive.google.com/file/d/0B9jjwkjdUJU7cElvcVI5U3hhdEU/view?usp=sharing (https://drive.google.com/file/d/0B9jjwkjdUJU7cElvcVI5U3hhdEU/view?usp=sharing).

The .csv is a version of the supplementary data provided by Sutter at the AER here: https://www.aeaweb.org/articles?id=10.1257/aer.99.5.2247 (https://www.aeaweb.org/articles?id=10.1257/aer.99.5.2247)) that I cleaned up to make it easier to import for this exercise.

Download the original paper here to make sense of the exercises below: https://drive.google.com/file/d/0B9jjwkjdUJU7eE5MVWRBZktlTm8/view?usp=sharing (https://drive.google.com/file/d/0B9jjwkjdUJU7eE5MVWRBZktlTm8/view?usp=sharing).

When you call on `read.csv()` or `read_csv()` be sure to include the file extension – .csv – at the end. Notice we use uppercase letters to name the data table we've prepared.

```
SutExp <- read.csv("../more/sutterexperiment.csv")
```

Or alternatively, using `read_csv`:

```
SutExp <- read_csv("../more/sutterexperiment.csv")
```

Using rename to have consistent variable

names

We want to have consistent naming protocols. So we shall quickly *clean* the data to make sure that all the variables are consistently labeled with lower case first letters to ensure we don't confuse them with data tables.

There are a couple of ways to do it, the shortes and easiest is as follows using the base R `tolower()` function which takes the input and change it to lowercase (hence its name `tolower()`):

```
names(SutExp) <- tolower(names(SutExp))
```

Alterantively, but more verbosely, we could use the `rename()` command for each variable:

```
SutExp <-
  SutExp %>%
  rename(session = Session, subject = Subject, r1 = R1, r2 = R2, r3 = R3, r4 = R4, r5
= R5, r6 = R6, r7 = R7, r8 = R8, r9 = R9, treatment = Treatment)
```

summarise and group_by

Now that we have an object with the data in it called `SutExp`, we want a data table that contains the *means* for each treatment for each round.

1. To find the means of the relevant variables we need to use `group_by` and `summarise`.
2. As you can see below, we want to `group_by()` with our Treatment variable (from Sutter's paper)
3. We then need to use `summarise()` to create the *means* of the variable in which we're interested: r1 through r9 (corresponding to rounds 1 through 9) respectively.
4. Notice that I create variables with lower case names to be consistent with our practice of naming variables in lowercase when we can.

```
groupAves <- #new data table called GroupAves
  SutExp %>%  #Using the SutExp data table
  group_by(treatment) %>% #Group the data by treatment
  summarise(r1 = mean(r1), r2 = mean(r2), r3 = mean(r3), r4 = mean(r4), r5 = mean(r5),
r6 = mean(r6), r7 = mean(r7), r8 = mean(r8), r9 = mean(r9))
```

1. This is the point at which you will need to make this *wide* data *narrow* before you can turn it into a plot.
2. To make the data narrow you will need to use the `gather` command.

Once you have narrowed your data you will then be able to call on `ggplot` to plot the data in a sensible way.

Narrowing the data:

Now we need narrow data to plot

```
narrowAve <- #name for the new data table
  groupAves %>% #start with the old data table
  gather(round, average, r1:r9) #gather the data to narrow it
```

This would be equivalent to:

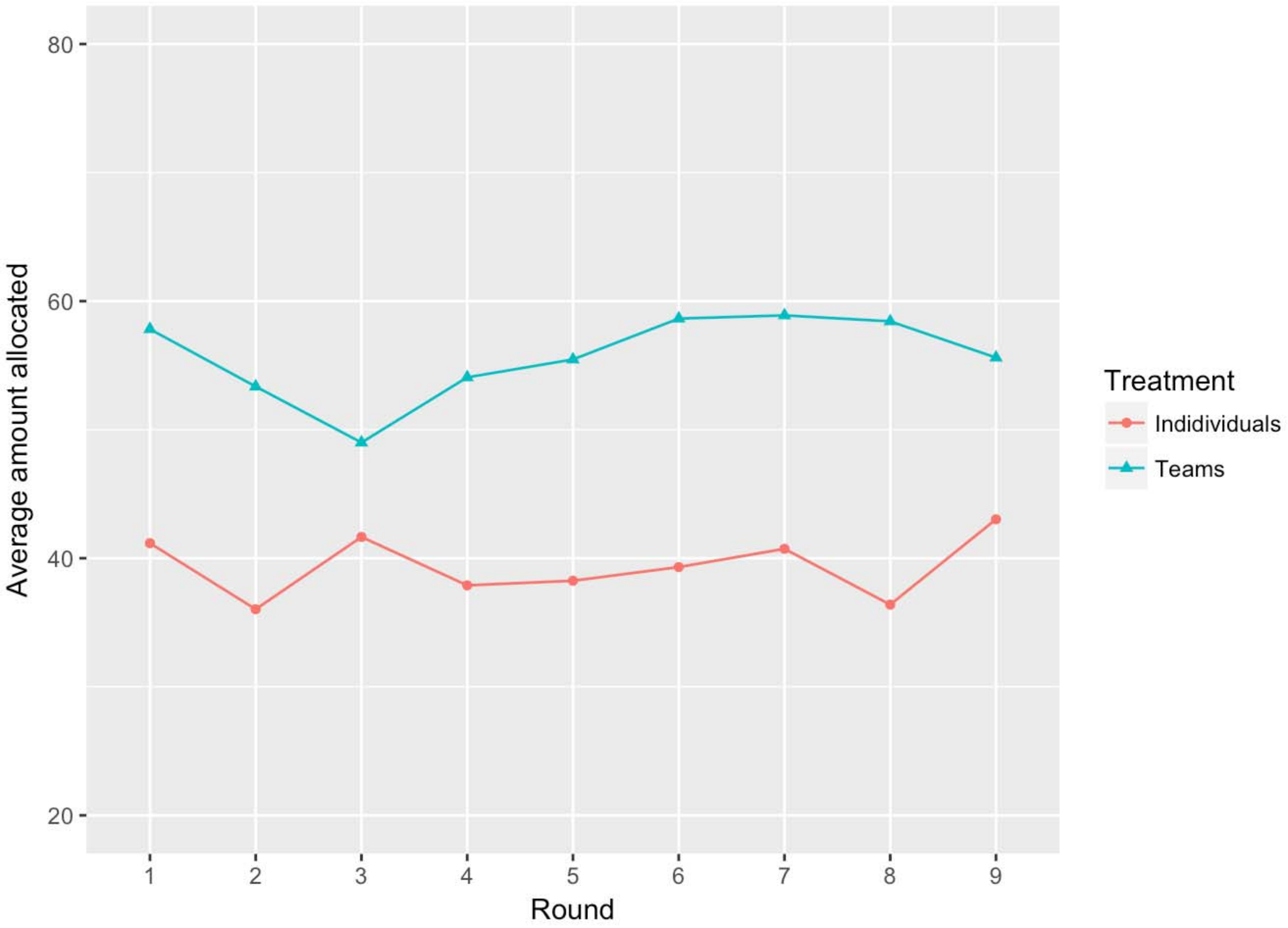
```
narrowAve <- gather(groupAves, round, average, r1:r9)
```

And now we can filter this data and put it into a plot.

Plot 1

We need to filter for individuals and teams, then construct the ggplot.

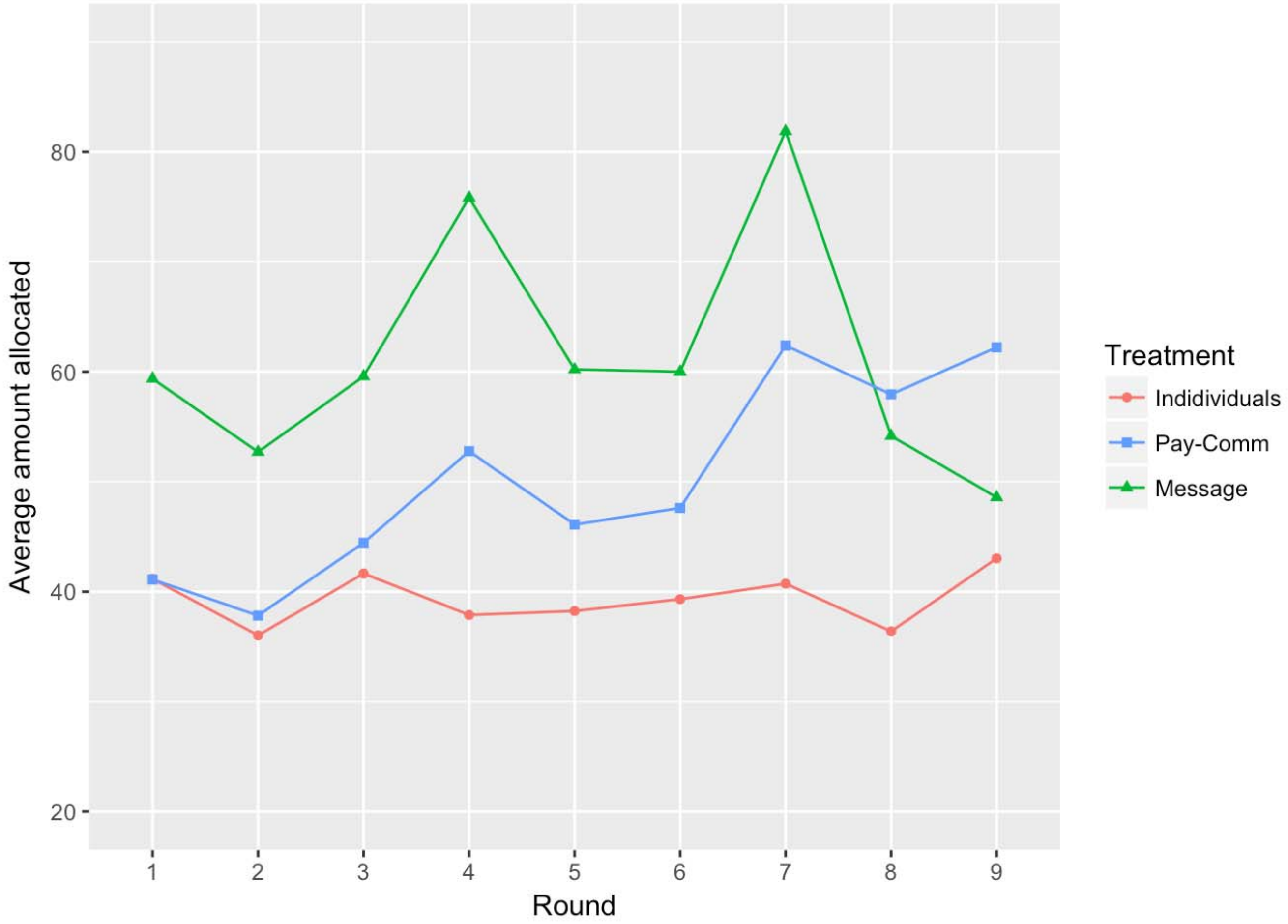
```
Plot1 <-
  narrowAve %>%
  filter(treatment == "individual" | treatment == "teamtreat" ) %>%
  group_by(treatment) %>%
  arrange(treatment)
Plot1 %>%
  ggplot(aes(x = round, y = average, color = treatment)) +
  geom_point(aes(shape = treatment)) +
  geom_line(aes(group = treatment)) +
  ylim(20, 80) +
  xlab("Round") +
  ylab("Average amount allocated") +
  scale_x_discrete(labels = seq(1, 9, by = 1)) +
  scale_colour_discrete(name = "Treatment",
                        breaks = c("individual", "teamtreat"),
                        labels = c("Indidividuals", "Teams")) +
  scale_shape_discrete(name = "Treatment",
                        breaks = c("individual", "teamtreat"),
                        labels = c("Indidividuals", "Teams"))
```



The second plot

We need to filter for individuals, paycomm and message, then construct the ggplot.

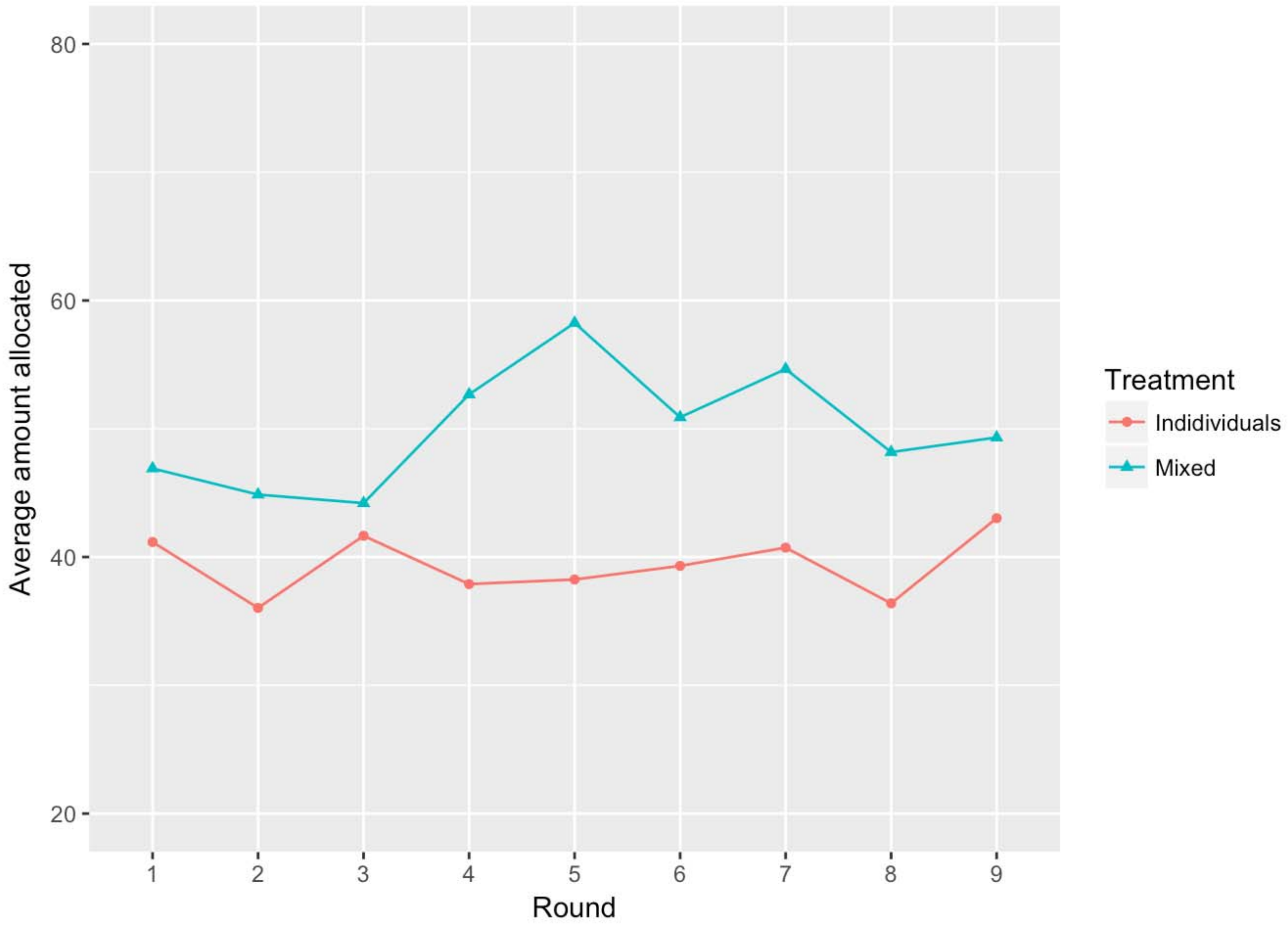
```
Plot2 <-
  narrowAve %>%
  filter(treatment == "individual" | treatment == "paycomm" |  treatment == "message")
%>%
  group_by(treatment) %>%
  arrange(treatment)
Plot2 %>%
  ggplot(aes(x = round, y = average, color = treatment)) +
  geom_point(aes(shape = treatment)) +
  geom_line(aes(group = treatment)) +
  ylim(20, 90) +
  xlab("Round") +
  ylab("Average amount allocated") +
  scale_x_discrete(labels = seq(1, 9, by = 1)) +
  scale_colour_discrete(name = "Treatment",
                        breaks = c("individual", "paycomm", "message"),
                        labels = c("Indidividuals", "Pay-Comm", "Message")) +
  scale_shape_discrete(name = "Treatment",
                        breaks = c("individual", "paycomm", "message"),
                        labels = c("Indidividuals", "Pay-Comm", "Message"))
```



The third plot

We need to filter for individuals and mixed, then construct the ggplot.

```
Plot3 <-
  narrowAve %>%
  filter(treatment == "individual" | treatment == "mixed" ) %>%
  group_by(treatment) %>%
  arrange(treatment)
Plot3 %>%
  ggplot(aes(x = round, y = average, color = treatment)) +
  geom_point(aes(shape = treatment)) +
  geom_line(aes(group = treatment)) +
  ylim(20, 80) +
  xlab("Round") +
  ylab("Average amount allocated") +
  scale_x_discrete(labels = seq(1, 9, by = 1)) +
  scale_colour_discrete(name = "Treatment",
                        breaks = c("individual", "mixed"),
                        labels = c("Indidividuals", "Mixed")) +
  scale_shape_discrete(name = "Treatment",
                        breaks = c("individual", "mixed"),
                        labels = c("Indidividuals", "Mixed"))
```



Reproducing Average Results

We’re now going to go about re-creating the averages that Sutter tests against each other in the paper. To do this, we’re first going to need unique identifiers for each subject. We’ll then look at different methods to find the average and diagnose why one doesn’t work.

We first need to create unique identifiers for each subject so that we don’t get confused across individuals (we will use this `uniqid` later in the regressions).

```
SutExpUniq <-
  SutExp %>%
  mutate(uniqid = paste(session, treatment, subject, sep = "_"))
head(SutExpUniq)
```

```
## # A tibble: 6 x 14
##   session subject    r1    r2    r3    r4    r5    r6    r7    r8    r9
##     <int>   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       1       1     0     0     0     0    10    10     0     0     0
## 2       1       2     0     0    30    40    40     0     0     0    20
## 3       1       3    30    30     0     0     0    60    60    10     0
## 4       1       4    20     0   100     0     0    30    75   100   100
## 5       1       5   100   100   100   100   100   100   100   100   100
## 6       1       6   100   100   100   100   100   100   100     0     0
## # ... with 3 more variables: treatment <chr>, team <int>, uniqid <chr>
```

We need individual averages for round, so we need to `group_by()` with uniqid:

```
SutUniqAves <-
  SutExpUniq %>%
  group_by(uniqid, treatment) %>%
  summarise(meanval = mean(r1:r9))
```

---

1. What information do we get from this? Look at `SutUniqAves` and think about what it contains and write a brief description of the table.
2. Is `SutUniqAves` informative? What are we comparing with what?
3. What other comparisons might you want to make?
4. Can we tell from what Sutter describes in the paper what he has averaged over? Why is this important in terms of *reproducibility*? * * *

Wilcoxon/Mann-Whitney Tests

In the paper, there's a statistical test that may be unfamiliar to many of you: the Mann-Whitney U-Test. Look up what the Mann-Whitney test is on Google. It is also known as the Wilcoxon Rank Sum test.

---

1. What does the Mann-Whitney test check?
2. Why do you think we use it rather than the Student t-test? (Think about the *assumptions* we have to make to use the Student t-test).

Wilcoxon tests check the differences of samples across a factor variable, such as an experimental treatment. The problem is it only wants *two* levels of that treatment, so we have to filter out the treatments we don't want to use. We only want "teamtreat" and "individuals" for now.

We could run a Wilcoxon test for the entire sample of the data for subjects for plot 1 and treat the rounds *for each individual* as if they are independent.

First, we need to narrow the data from our original SutExp data table:

```
SutNarrow <-
  SutExpUniq %>%
  gather(round, value, r1:r9) %>%
  arrange(session, subject, treatment, team) #arranging isn't necessary, it's just more attractive
Plot1data <- #define the object 'plot 1 data'
  SutNarrow %>% #use SutNarrow to get that data
  filter(treatment == "individual" | treatment == "teamtreat" ) #filter on the categories we want
wilcox.test(value ~ treatment, data = Plot1data) # run the M-W test
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  value by treatment
## W = 52876, p-value = 3.838e-10
## alternative hypothesis: true location shift is not equal to 0
```

An alternative might be to find an average for each subject. We've already done this with creating the data table `SutUniqAves`. We now filter out the rows we don't need and run a Wilcoxon (Mann-Whitney) test.

```
Plot1AveData <-
  SutUniqAves %>%
  filter(treatment == "individual" | treatment == "teamtreat" )
wilcox.test(meanval ~ treatment, data = Plot1AveData)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  meanval by treatment
## W = 660, p-value = 0.04532
## alternative hypothesis: true location shift is not equal to 0
```

What might the commands below tell us? Is it helpful?

```
Plot1Aves <-
  Plot1data %>%
  group_by(uniqid, round) %>%
  summarise(aveval = mean(value))
```

Regressions

We now want to run a regression using value as the dependent variable (LHS) and each of the treatments as a dummy variable (or categorical variable). We can think of dummy variables as variables that take 1 value of '1' if you are in that category and '0' if you are not in that category. So we have categorical (dummy) variables for the participants in each experimental treatment. We then use R's command for an OLS regression, `lm()` to find out what the value is predicted by participating in each treatment. The coefficient of each treatment will tell us what the average amount to add to the intercept of value should be for participating in that treatment relative to an excluded category.

Running a regression in R requires specifying an equation. The equation takes the form $y \sim x + z + \ldots$. Because we only have one categorical variable we are interested in, we just have to specify that variable. Notice that R is intelligent in how it reads this. It will specify a list of dummy variables corresponding to each category (each treatment). This is because categories are encoded as ‘factors’ in R, and R is programmed to treat factor (category) variables in this way when dealing with factor variables.

```
m1 <- lm(value ~ treatment, data = SutNarrow)
summary(m1)
```

```
##
## Call:
## lm(formula = value ~ treatment, data = SutNarrow)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.370 -29.385  -0.542  38.630  60.615
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          39.385      1.451  27.152  < 2e-16 ***
## treatmentmessage     21.985      1.994  11.028  < 2e-16 ***
## treatmentmixed       10.609      1.925   5.510 3.92e-08 ***
## treatmentpaycomm     10.886      2.144   5.077 4.09e-07 ***
## treatmentteamtreat   16.313      2.629   6.204 6.34e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34.81 on 2713 degrees of freedom
## Multiple R-squared:  0.04473,    Adjusted R-squared:  0.04333
## F-statistic: 31.76 on 4 and 2713 DF,  p-value: < 2.2e-16
```

The first command assigns to “m1” a linear model/OLS model (using the `lm()` function) where value is a function of treatment. Treatment is a “factor”, so R automatically converts the different components of the factor into dummy variables in a linear regression.

You then call on `summary()` to get a summary of the object `m1` created by the linear model. `summary()` provides us the basic details we need to know about coefficient size, intercept, statistical significance and goodness of fit.

An alternative way of getting this regression output is to use the `stargazer` package, which makes much prettier output tables. (Be sure to use `install.packages("stargazer")` if you don’t have it installed.)

```
stargazer(m1, type = "html")
```

	Dependent variable:
	value
treatmentmessage	21.985^{***}

	(1.994)
treatmentmixed	10.609***
	(1.925)
treatmentpaycomm	10.886***
	(2.144)
treatmentteamtreat	16.313***
	(2.629)
Constant	39.385***
	(1.451)
Observations	2,718
R^2	0.045
Adjusted R^2	0.043
Residual Std. Error	34.813 (df = 2713)
F Statistic	31.762*** (df = 4; 2713)
Note:	*p<0.1;* ***p<0.05;*** p<0.01

As you can see, the output from `stargazer()` looks much nicer in our html file than the ASCII text output of the `summary()` function. Notice that we had to specify the option 'results = "asis" in the code chunk option. We also had to specify `"type = "html"` for the `stargazer` function because we are using html output.

Note, if we call `stargazer` on a data table, then it immediately will produce summary statistics of the relevant table. Check this by calling stargazer on a relevant data table in which you are interested. Note, `stargazer` can be a little finicky as it was written by people *outside* of the tidyverse, so it likes you to specify that your object is a dataframe rather than a tibble (the structure of the data table that we have used so far with tidyverse).

What did I do here about specifying the type of output stargazer produced? How is it different to the previous time I ran the command?

```
SutExpDF <- as.data.frame(SutExp) #stargazer likes dataframes not tibbles
stargazer(SutExpDF, type = "text")
```

```
##
## =====================================
## Statistic  N   Mean  St. Dev. Min Max
## -------------------------------------
## session   302 2.089   0.948    1   4
## subject   302 12.563  7.371    1  28
## r1        302 48.639  30.816   0  100
## r2        302 44.394  34.167   0  100
## r3        302 47.818  35.917   0  100
## r4        302 55.212  33.229   0  100
## r5        302 52.050  32.818   0  100
## r6        302 50.742  36.227   0  100
## r7        302 59.970  36.994   0  100
## r8        302 49.805  38.007   0  100
## r9        302 50.702  39.505   0  100
## team      238 4.504   2.440    1   9
## -------------------------------------
```

Exercise 1 Is the summary table above *informative*? What are we able to learn from it?

Exercise 2 If you wanted a different set of information, what would you want to do change in SutExpDF to get other useful information? (this is a broad question; I'm asking you what you think useful information is in this context) * * *

Panel data

Another way of doing this regression is as a "panel regression." For a panel regresion, you treat each variable as if it comes from the same individual over time (hence a panel). We have to specify a few options if we are running a panel regression. We need to tell R the equation and data, as previously, but we also need to tell it what variable uniquely identifies each individual over time and we need to tell it what kind of model we want (as there are several kinds of models you can use for panel regressions).

Here's one way of doing it using the `plm` package.

```
m2 <- plm(value ~ treatment, data = SutNarrow, index = c("uniqid"), model = "pooling")
summary(m2)
```

```
## Pooling Model
##
## Call:
## plm(formula = value ~ treatment, data = SutNarrow, model = "pooling",
##     index = c("uniqid"))
##
## Balanced Panel: n=302, T=9, N=2718
##
## Residuals :
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -61.37037 -29.38542  -0.54191  38.62963  60.61458
##
## Coefficients :
##                    Estimate Std. Error t-value  Pr(>|t|)
## (Intercept)         39.3854     1.4505 27.1523 < 2.2e-16 ***
## treatmentmessage    21.9850     1.9936 11.0279 < 2.2e-16 ***
## treatmentmixed      10.6093     1.9254  5.5102 3.921e-08 ***
## treatmentpaycomm    10.8862     2.1442  5.0770 4.095e-07 ***
## treatmentteamtreat  16.3130     2.6293  6.2043 6.335e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    3441900
## Residual Sum of Squares: 3288000
## R-Squared:      0.044734
## Adj. R-Squared: 0.043325
## F-statistic: 31.7615 on 4 and 2713 DF, p-value: < 2.22e-16
```

And another similar way of doing it with the same package:

```
m3 <- plm(value ~ treatment, data = SutNarrow, index = c("uniqid"), model = "random",
effect = "time")
summary(m3)
```

```
## Oneway (time) effect Random Effect Model
##    (Swamy-Arora's transformation)
##
## Call:
## plm(formula = value ~ treatment, data = SutNarrow, effect = "time",
##     model = "random", index = c("uniqid"))
##
## Balanced Panel: n=302, T=9, N=2718
##
## Effects:
##                     var  std.dev share
## idiosyncratic 1197.631   34.607 0.985
## time            18.044    4.248 0.015
## theta:  0.5755
##
## Residuals :
##     Min.  1st Qu.   Median  3rd Qu.     Max.
## -61.9533 -28.6148  -2.6745  33.8010  64.4377
##
## Coefficients :
##                     Estimate Std. Error t-value  Pr(>|t|)
## (Intercept)          39.3854     2.0207 19.4914 < 2.2e-16 ***
## treatmentmessage    21.9850     1.9815 11.0951 < 2.2e-16 ***
## treatmentmixed      10.6093     1.9137  5.5437 3.246e-08 ***
## treatmentpaycomm    10.8862     2.1313  5.1079 3.484e-07 ***
## treatmentteamtreat  16.3130     2.6134  6.2420 4.996e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    3402300
## Residual Sum of Squares: 3248300
## R-Squared:      0.045255
## Adj. R-Squared: 0.043848
## F-statistic: 32.1494 on 4 and 2713 DF, p-value: < 2.22e-16
```

Exercise 3 See if you can produce the output of `m2` and `m3` using `stargazer` .

Exercise 4 What do you think the difference is between the ways in which the two models – `m2` and `m3` are specified? How are they different to `m1` ?

Exercise 5 Why do you think it's important statistically to tell a program that there is a person taking all the actions repeatedly rather than not telling the program that when doing the statistical calculations? * * *