cdf5579_econdev_HW5

2023-04-06

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Carlos Figueroa cdf5579

1 Randomization and Luck

In class, I vaguely mentioned that you can get "unlucky" when you randomize, and I want to go over that a bit more. So let's think of a really simple experiment: we are interested in understanding the effect of a placebo on height. Thankfully, Rafael Irizarry has provided some teaching datasets including one with heights (in inches). Enter the following code to load the dataset.

install.packages("dslabs") library(dslabs) data("heights") df = as_tibble(heights)

Here is conceptually what we are doing. We are going to randomize the students into treatment and control. We are then going to give the treatment students the placebo, and nothing to the control students.

After a year, we measure their heights.

1. First, let's do one randomization. We want to randomize students into zero or one with probability .5, which we can do with the rbinom() command. Type

df$random <- rbinom(n = nrow(df), size = 1,prob = .5)

to create a new column that's random (0,1) with probability .5

```
#install.packages("dslabs")
library(dslabs)
```

```
## Warning: package 'dslabs' was built under R version 4.2.3
```

```
library(tibble)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data("heights")

df = as_tibble(heights)

df$random <- rbinom(n = nrow(df), size = 1,prob = .5)
```

Now it's a year later. Let's see if the placebo worked. Run a regression predicting height using random using the lm() command.1

What coefficient do you get for random? You don't have to type out the whole thing, just a few digits.

```r
lmfit <- lm(height ~ random, df)

summary(lmfit)
```

```
##
## Call:
## lm(formula = height ~ random, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.3848  -2.3848   0.1191   2.7403  14.2923
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  68.2597     0.1791 381.136   <2e-16 ***
## random        0.1251     0.2518   0.497    0.619
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.08 on 1048 degrees of freedom
## Multiple R-squared:  0.0002354,  Adjusted R-squared:  -0.0007186
## F-statistic: 0.2468 on 1 and 1048 DF,  p-value: 0.6195
```

We get -0.2131 for the random coefficient

2.Now do it again - overwrite the random column with the exact same df$random <- rbinom(n = nrow(df), size = 1,prob = .5) command and run the exact same regression (so the thought experiment is we go back in time, re-randomize, and then see what happens a year later). What coefficient do you get for random?

It's different! Which seems bad - we would like our work to be reproducible. For instance, if you tell me that you randomized a group into treatment and control, I would like to be able to verify that you did the randomization correctly. It turns out that R doesn't make a truly random number, but something that can be reproduced. The relevant thing here is called the "seed:" if you know the seed, then you can exactly get the same random values in a systematic way. Set the seed. You can pick whatever seed you want (google if you don't know how to set a seed in R). We're now going to want to randomize this many many times. The way to automate this is to start by defining a function.

Type: mySimulation <- function() { df$random < -rbinom(n = nrow(df), size = 1, prob = .5) lmfit < -summary(lm(height random, df))$coefficients[2,1] return(lmfit) }

What this does is calls a function mySimulation() which first creates the random column random, and then runs the regression, and then prints the coefficient on random.

If you run mySimulation() over and over it will give you a different answer each time, but that's good: once you set a seed, the path of estimated placebo coefficients will be the same, even if the specific value changes each iteration.

```r
df$random <- rbinom(n = nrow(df), size = 1,prob = .5)

lmfit <- lm(height ~ random, df)

summary(lmfit)
```

```
##
## Call:
## lm(formula = height ~ random, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.4978  -2.1616   0.1222   2.8384  14.5155
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  68.1616     0.1745 390.648   <2e-16 ***
## random        0.3362     0.2518   1.335    0.182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.077 on 1048 degrees of freedom
## Multiple R-squared:  0.001697,   Adjusted R-squared:  0.0007449
## F-statistic: 1.782 on 1 and 1048 DF,  p-value: 0.1822
```

```r
summary(lm(height ~ random, df))$coefficients[2,1]
```

```
## [1] 0.336191
```

At random we got -0.4692879. Way different from before.

So now what we're going to do is run mysimulation 10000 times, in order to get a sense of the distribution of coefficients of the placebo on heights. This is called a Monte Carlo simulation, and its invention was crucial to the development of the atomic bomb, which is a little higher stakes than this problem set.

A way to do this in R is with the sapply function. This is a helpful website walking through sapply for functions: https://r-coder.com/sapply-function-r/ (If you want to use a different function that's totally fine for me - there are lots of ways to do this in R)

```r
#lets do montecarlo simulation setting seed to zero
set.seed(0)

mySimulation <- function(foobar){
df$random <- rbinom(n = nrow(df), size = 1,prob = .5)
lmfit <- summary(lm(height ~ random, df))$coefficients[2,1]
return(lmfit)
}

runs <- 10000

#run the function 10000 times

MCplacebo <- replicate(runs,mySimulation())
```

3. Create a vector with 10000 elements called MCplacebo in which each cell is one run of mySimulation().

Write down the code you used in order to create MCplacebo Depending on how you do this question, it may be a really wide vector, which isn't super convenient. t() will transpose it to a column, and then we can use as tibble() to make a dataframe.

MCplacebo <- t(MCplacebo) %>% as_tibble()

This code is to make it convenient, you have to do MCplacebo from scratch

```
MCplacebo <- as_tibble((MCplacebo))

head(MCplacebo)
```

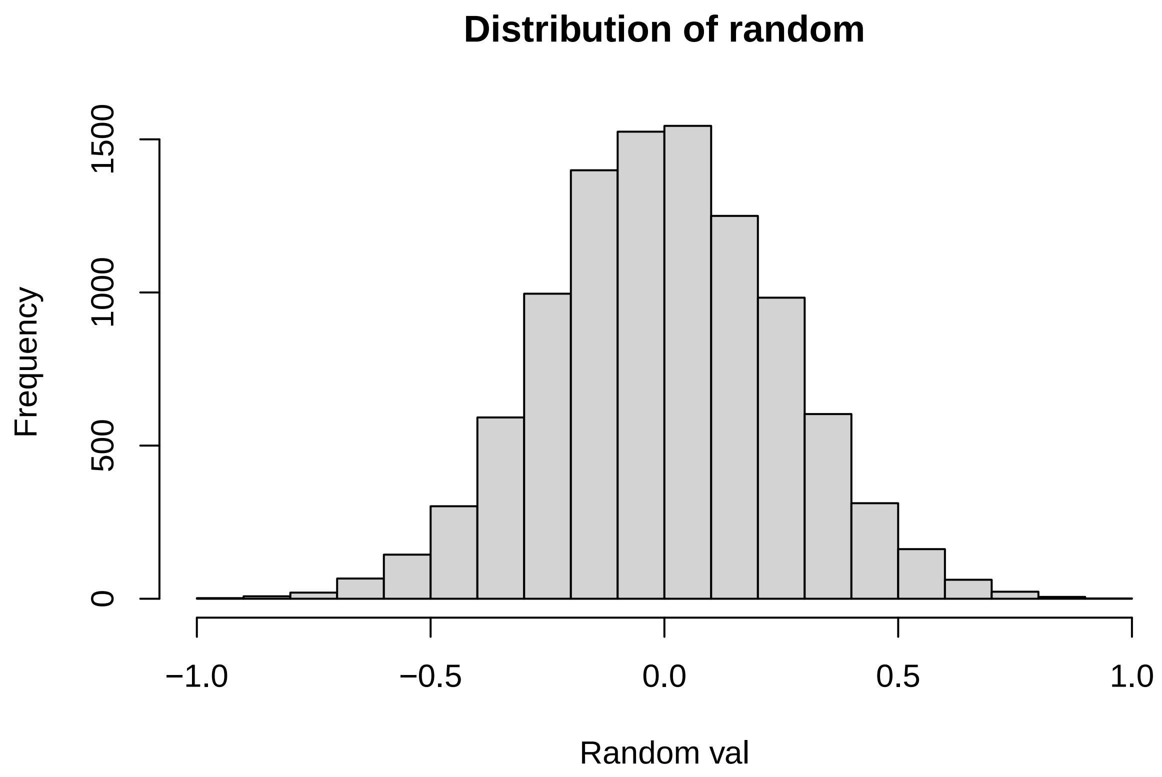
```
## # A tibble: 6 x 1
##     value
##     <dbl>
## 1  0.0884
## 2 -0.381
## 3 -0.469
## 4  0.279
## 5 -0.206
## 6 -0.0916
```

```
#configure it to be a dataset and have the right dimensions
```

4. What's the distribution of estimated coefficients? You could show this with a histogram (or a kernel density plot if you want to be fancy)

```
colnames(MCplacebo) <- c('random')


hist(MCplacebo$random, main = 'Distribution of random', xlab = 'Random val')
```



Kind of a normal distribution centerd around zero.

5. Hopefully your distribution is centered around zero. But what do the tails (the upper and lower extremes) tell us about getting "unlucky?"

It tells us that even though it is unlikely, it is still possible to get unlucky when you randomize, especially without a seed. So sometimes it might look like the treatment had an effect because we randomized, but in reality we just were unlucky and the randomization wasn't enough to measure the real treatment effect.

6. Remember the thought experiment: we gave some students the placebo, then measured their heights a year later. Obviously the treatment didn't affect their actual heights, although in your previous question hopefully you wrote about how sometimes it might have looked like it did. What additional data might you want to collect in order to minimize the chance of getting unlucky (This is also not something we've talked about in class, so describe your intuition).

In order to minimize the chance of getting unlucky I will also collect the ages of the participants, together with the heights of their parents. If participants are above 23 years old, it is safer to say that the treatment did not work for them since they developed the height they had to develop already at that point of their lives. Randomizing withouth having age can lead us to having a kid that is just developing, and shifting our results, and that is one example of how we might get unlucky with data. Also, by having the height of their parents, we have a pretty good approximation of where the bounds of his height might be due to genetics.

2 Permanent Vs. Transitory Shocks This section is based on Christina Paxson's paper Using Weather Variability to Estimate the Response of Savings to Transitory Income in Thailand (1992). Please familiarize yourself with the paper moving on.

There is a csv in the assignment folder on Brightspace, which you will want to load into R called paxson corrected, and an associated codebook (which tells you what each variable name means in english).

7. Paxson (1992) attempts to estimate the marginal propensity to save out of transitory income. She states, "finding that these marginal propensities are high would indicate that farmers do use savings to smooth consumption." Explain the intuition for why this is true Paxson only has data on total income (the variable inc). She wants to decompose it into a permanent income component (call it incperm) and a transitory income component (call it inctrans). For the following questions, you will need to replicate the results in Table 3 of Paxson (1992) and obtain estimates of incperm and inctrans

```
df <- read.csv("paxson_corrected.csv")

head(df)
```

```
##        hid       save3 year region       inc       save1      save2 m12 m18 m65
## 1 1.21e+08 -2998.38100   76     19 3263.3600 -576.72890 -396.69610   0   0   1
## 2 2.21e+08   251.14160   76     32 1890.5190   31.33758  288.90550   1   2   1
## 3 2.11e+08   677.06710   76     26  863.9617 -623.06190   58.78345   1   0   0
## 4 3.21e+08   757.66870   76     28 7453.7150 2339.59600 2691.20200   0   1   0
## 5 2.11e+08   -44.97751   76     26 1110.3420  231.66560  236.27920   0   0   0
## 6 1.21e+08  5522.62100   76     19 5374.6990 1000.43600 1696.79200   0   0   0
##   m18elem m18sec m18pos f12 f18 f65 f18elem f18sec f18pos of1 of2 of3 of4 of5
## 1       1      0      0   0   0   1       0      0      0   0   0   0   0   1
## 2       0      0      0   1   0   0       1      0      0   0   0   0   1   0
## 3       1      0      0   0   0   0       0      0      0   0   0   0   1   0
## 4       2      0      0   2   1   1       0      0      0   0   0   0   0   0
## 5       1      0      0   1   1   0       0      0      0   0   0   0   1   0
## 6       1      0      0   0   0   0       1      0      0   0   0   0   0   0
##   p0to5 p6to11 p12to17 p18to64 p65 of0 month       cpi      dev1      dev2
## 1     1      0       0       1   2   0    12 0.6502500  1.676471  74.34706
## 2     1      2       2       1   1   0     5 0.6363333 -4.964706 -97.27941
## 3     1      1       0       1   0   0     2 0.6285000 76.517650 119.19710
## 4     0      2       2       2   1   0     7 0.6400833 -9.182353  67.60588
## 5     0      1       1       1   0   0    12 0.6502500 -1.100000 119.19710
## 6     0      0       0       2   0   0     2 0.6285000 83.514710  74.34706
##         dev3       dev4      sd1      sd2      sd3       sd4       dvsq1
## 1  53.27576   8.194118 36.84246 115.7548 130.0240 108.53860    2.810554
## 2 -55.50882  97.952940 29.06564 110.1495 143.6269  89.68748   24.648300
## 3 -38.65000  42.208820 37.74985 122.7939 158.2939  75.52906 5854.950000
## 4 -43.02353  71.055890 42.40917 127.8315 122.4276 146.26660   84.315610
## 5 -38.65000 147.105900 39.96710 122.7939 158.2939  71.55458    1.210000
## 6  53.27576 184.944100 33.94309 115.7548 130.0240 103.80610 6974.706000
##       dvsq2    dvsq3       dvsq4
## 1  5527.485 2838.306    67.14356
## 2  9463.284 3081.229  9594.77800
## 3 14207.940 1493.823  1781.58500
## 4  4570.555 1851.024  5048.93800
## 5 14207.940 1493.823 21640.14000
## 6  5527.485 2838.306 34204.33000
```

```
df <- read.csv("paxson_corrected.csv")

colnames(df)
```

```
##  [1] "hid"     "save3"   "year"    "region"  "inc"     "save1"   "save2"
##  [8] "m12"     "m18"     "m65"     "m18elem" "m18sec"  "m18pos"  "f12"
## [15] "f18"      "f65"      "f18elem"  "f18sec"   "f18pos"   "of1"      "of2"
## [22] "of3"      "of4"      "of5"      "p0to5"    "p6to11"   "p12to17"  "p18to64"
## [29] "p65"      "of0"      "month"    "cpi"      "dev1"     "dev2"     "dev3"
## [36] "dev4"     "sd1"      "sd2"      "sd3"      "sd4"      "dvsq1"    "dvsq2"
## [43] "dvsq3"    "dvsq4"
```

8. Explain the logic of Table 3 - what exactly does Paxson do to get a "permanent" measure of income? Using the "dummies" package (that you will have to install and then load into your library), add to your dataframe dummy variables for region and year, you will need these for the regression.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.2.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:purrr':
##
##     some
##
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(broom)

#Dummies per unique years

year.name <- paste('year', unique(df$year), sep='.')

df <- df %>%
  mutate(new = 1) %>%
  spread(year, new, fill = 0, sep = '.')

#Dummies per region
region.name <- paste('region', unique(df$region), sep='.')

df <- df %>%
  mutate(new = 1) %>%
  spread(region, new, fill = 0, sep = '.')

#Now we check if there are here

colnames(df)
```

```
##  [1] "hid"       "save3"     "inc"       "save1"     "save2"     "m12"
##  [7] "m18"       "m65"       "m18elem"   "m18sec"    "m18pos"    "f12"
## [13] "f18"       "f65"       "f18elem"   "f18sec"    "f18pos"    "of1"
## [19] "of2"       "of3"       "of4"       "of5"       "p0to5"     "p6to11"
## [25] "p12to17"   "p18to64"   "p65"       "of0"       "month"     "cpi"
## [31] "dev1"      "dev2"      "dev3"      "dev4"      "sd1"       "sd2"
## [37] "sd3"       "sd4"       "dvsq1"     "dvsq2"     "dvsq3"     "dvsq4"
## [43] "year.76"   "year.81"   "year.86"   "region.2"  "region.5"  "region.6"
## [49] "region.8"  "region.9"  "region.10" "region.11" "region.13" "region.14"
## [55] "region.15" "region.17" "region.19" "region.20" "region.24" "region.26"
## [61] "region.27" "region.28" "region.29" "region.31" "region.32" "region.34"
```

Using the "lm" command, run table 3 column 1. The way I would do it would be to create a dataframe called irdat that contains only the relevant variables (so inc, the rainfall variables, the education variables, the fixed effects you just made, and so on). Then you can run the regression ir <- lm(inc ~ .,data=irdat)

```
#we list the variables we don't want in irdat

ex_vars <- c("hid", "save3", "save1", "save2", "month", "cpi", "sd1", "sd2", "sd3", "sd4", "p6to11",
             "p12to17", "p18to64", "p65" ,"year.76", "region.19")

#we then exclude them and create the dataset used for column 1 in the paper

irdat <- df[,!names(df) %in% ex_vars]

ir <- summary(lm(inc ~ .,data=irdat))

#lets print the summary
ir
```

```
##
## Call:
## lm(formula = inc ~ ., data = irdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3912.3  -625.3  -138.6   392.4 20135.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.692e+03  1.151e+02  23.382  < 2e-16 ***
## m12          5.973e+01  2.606e+01   2.292 0.021929 *
## m18          2.206e+02  2.718e+01   8.114 6.18e-16 ***
## m65          1.355e+02  6.817e+01   1.988 0.046881 *
## m18elem      3.494e+02  2.659e+01  13.138  < 2e-16 ***
## m18sec       7.657e+02  9.344e+01   8.195 3.19e-16 ***
## m18pos       1.043e+03  1.356e+02   7.690 1.77e-14 ***
## f12          7.955e+01  2.515e+01   3.162 0.001575 **
## f18          1.930e+02  2.724e+01   7.084 1.61e-12 ***
## f65          1.597e+02  6.129e+01   2.605 0.009211 **
## f18elem      6.231e+01  3.850e+01   1.618 0.105641
## f18sec       3.456e+02  1.336e+02   2.588 0.009682 **
## f18pos       6.769e+02  2.040e+02   3.318 0.000912 ***
## of1         -1.700e+03  3.110e+02  -5.465 4.87e-08 ***
## of2         -1.769e+03  1.084e+02 -16.317  < 2e-16 ***
## of3         -1.583e+03  7.549e+01 -20.973  < 2e-16 ***
## of4         -1.368e+03  6.481e+01 -21.114  < 2e-16 ***
## of5         -1.008e+03  6.308e+01 -15.986  < 2e-16 ***
## p0to5        3.769e+01  2.178e+01   1.731 0.083518 .
## of0         -1.339e+03  7.071e+01 -18.933  < 2e-16 ***
## dev1         1.909e+00  7.565e-01   2.524 0.011642 *
## dev2         1.250e+00  2.252e-01   5.552 2.97e-08 ***
## dev3         2.282e-01  2.274e-01   1.004 0.315598
## dev4         1.610e+00  6.269e-01   2.568 0.010272 *
## dvsq1       -4.499e-02  1.127e-02  -3.993 6.63e-05 ***
## dvsq2        8.775e-04  1.322e-03   0.664 0.506826
## dvsq3        4.446e-04  7.157e-04   0.621 0.534433
## dvsq4       -9.472e-03  3.323e-03  -2.851 0.004382 **
## year.81      3.017e+02  4.721e+01   6.391 1.81e-10 ***
## year.86     -4.023e+02  8.294e+01  -4.850 1.28e-06 ***
## region.2    -2.367e+02  1.411e+02  -1.678 0.093437 .
## region.5    -2.309e+02  1.557e+02  -1.484 0.137973
## region.6     2.989e+01  1.419e+02   0.211 0.833150
## region.8    -6.910e+02  2.022e+02  -3.417 0.000637 ***
## region.9    -7.152e+02  1.532e+02  -4.668 3.12e-06 ***
## region.10   -2.096e+02  1.867e+02  -1.123 0.261630
## region.11   -1.125e+03  1.314e+02  -8.561  < 2e-16 ***
## region.13   -3.362e+02  1.045e+02  -3.216 0.001309 **
## region.14   -7.978e+02  1.287e+02  -6.198 6.20e-10 ***
## region.15   -3.039e+01  1.727e+02  -0.176 0.860303
## region.17   -4.043e+02  1.405e+02  -2.878 0.004024 **
## region.20    8.351e+02  2.060e+02   4.054 5.11e-05 ***
## region.24    1.034e+02  1.120e+02   0.924 0.355626
## region.26   -8.330e+02  9.398e+01  -8.864  < 2e-16 ***
## region.27   -3.640e+02  1.305e+02  -2.789 0.005310 **
## region.28    8.232e+02  1.295e+02   6.357 2.24e-10 ***
## region.29    5.713e+02  1.281e+02   4.460 8.38e-06 ***
## region.31   -2.529e+02  1.134e+02  -2.230 0.025809 *
## region.32   -6.524e+02  9.500e+01  -6.867 7.38e-12 ***
## region.34    1.008e+03  2.104e+02   4.791 1.71e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1243 on 4805 degrees of freedom
## Multiple R-squared:  0.3426, Adjusted R-squared:  0.3359
## F-statistic: 51.11 on 49 and 4805 DF,  p-value: < 2.2e-16
```

Using the coefficients estimated in the regression for Table 3, Column 1, Paxson constructs a predicted value for permanent income as follows. She multiples the permanent characteristics by their respective coefficients, and adds them up to form incperm (see equation 2 on page 17 of the paper.) Generate the variable incperm.

Here's how I would do this:

You can store the summary() of a linear model, then use coef() to generate a table of coefficients. To get a specific coefficient, use coef(reg sum)[' 'x",“Estimate”]. Make a dataframe just of the permanent variables, call it permvars

num_pv <- length(permvars) INCPERM <- 0 for (i in 1:num_pv) { thisvar <- permvars[i] INCPERM <- INCPERM + coef(betas)[thisvar,“Estimate”]*irdat[,c(thisvar)] }

I put INCPERM in caps because you will want to put it in a dataframe, and I'm not giving you hints about that since you should know how to do it. Eventually you are going to want to run a regression of savings on income, so be thoughtful about exactly how you do this, since you will also want to include the controls of Table 4 column 1

```
#these variables aren't used to construct INCPERM since they aren't permanent per se
#they were useful only for the modeling part

ex_vars <- c("dev1","dev2","dev3","dev4","dvsq1","dvsq2","dvsq3","dvsq4")

#lets extract transitory variables listed above
irdat <- irdat[,!names(irdat) %in% ex_vars]

#now we set the variables right in order to replicate
permvars <- colnames(irdat)

#take inc variable out
permvars <- permvars[-1]

num_pv <- length(permvars)
INCPERM <- 0

for (i in 1:num_pv) {
thisvar <- permvars[i]
INCPERM <- INCPERM + coef(ir)[thisvar,"Estimate"]*irdat[,c(thisvar)]
}

#INCPERM

df <- df %>%
  mutate(incperm =INCPERM)
```

9. What is the standard deviation of incperm? Also, explain what the function is doing, row by row.

```
print(paste("Standard deviation of estimated permanent income:", sd(df$incperm))) # standard deviation
```

```
## [1] "Standard deviation of estimated permanent income: 906.971430727156"
```

Explain what the function is doing, row by row: Basically, its multiplying the linear regression coefficient corresponding to the variable vector, to all of the rows in the specific vector. In this case, it is multiplying the vectors of permanent variables of the dataset by their corresponding coefficients from the model to approximate the percentage of income coming from these permanent variables, since the linear model is predicting the relation between all variables with income. So that is why it is an appropriate way of measuring permanent income per individual in the dataset.

10. Do something similar to create a variable of transitory income, inctrans. What is the standard deviation of inctrans?

```
#only transitory variables
in_vars <- c("dev1","dev2","dev3","dev4","dvsq1","dvsq2","dvsq3","dvsq4", "year.81", "year.86")

#in this case we are not excluding but only including the list above
rdat <- df[,names(df) %in% in_vars]

#same procedure now
inmvars <- colnames(rdat)

num_pv <- length(inmvars)

inctrans <- 0

for (i in 1:num_pv) {
thisvar <- inmvars[i]
inctrans <- inctrans + coef(ir)[thisvar,"Estimate"]*rdat[,c(thisvar)]
}


df <- df %>%
  mutate(inctrans =inctrans)

print(paste("Standard deviation of estimated transient income:",sd(df$inctrans)))
```

```
## [1] "Standard deviation of estimated transient income: 325.410827363853"
```

11. Paxson also has a category called unexplained income, defined as inc – incperm – inctrans. Form this variable and call it incunexp. What is the standard deviation of incunexp?

```
df <- df %>%
  mutate(incunexp = inc - incperm - inctrans)

print(paste("Standard deviation of estimated transient income:",sd(df$incunexp)))
```

```
## [1] "Standard deviation of estimated transient income: 1258.25912521469"
```

You will now run a regression to estimate the effect of income on savings. Use the variable save2 as your measure of savings. So the x variables are the three measures of income that you made, and the contols from footnote 2.

12. What do you estimate for the marginal propensity to save out of each additional dollar of permanent income? What do you estimate for the marginal propensity to save out of each additional dollar of transitory income? Do the magnitudes align with the theory?

```
#I manually copy pasted all the variables since they are less than before

second.regression <- 'save2 ~ incperm+inctrans+incunexp+
p0to5+p6to11+p12to17+p18to64+p65+
sd1+sd2+sd3+sd4+year.81+year.86'

results <- lm(as.formula(second.regression),df)

summary(results)
```

```
##
## Call:
## lm(formula = as.formula(second.regression), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -122338    -226     174     473    4095
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.754e+03  4.415e+02  -3.973 7.20e-05 ***
## incperm      4.400e-01  4.854e-02   9.064  < 2e-16 ***
## inctrans     8.039e-01  1.633e-01   4.922 8.86e-07 ***
## incunexp     6.925e-01  2.331e-02  29.707  < 2e-16 ***
## p0to5       -5.285e+01  3.454e+01  -1.530   0.1260
## p6to11       7.832e+00  2.950e+01   0.266   0.7906
## p12to17     -4.973e+01  3.388e+01  -1.468   0.1422
## p18to64     -3.881e+01  3.725e+01  -1.042   0.2975
## p65         -1.228e+02  6.401e+01  -1.918   0.0552 .
## sd1          1.738e+00  2.958e+00   0.587   0.5570
## sd2         -3.075e+00  1.588e+00  -1.937   0.0528 .
## sd3          4.007e+00  2.177e+00   1.841   0.0657 .
## sd4          3.473e+00  2.037e+00   1.705   0.0883 .
## year.81     -6.984e+01  8.273e+01  -0.844   0.3986
## year.86     -2.881e+02  1.374e+02  -2.097   0.0361 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2008 on 4840 degrees of freedom
## Multiple R-squared:  0.1868, Adjusted R-squared:  0.1844
## F-statistic: 79.39 on 14 and 4840 DF,  p-value: < 2.2e-16
```

What do you estimate for the marginal propensity to save out of each additional dollar of permanent income?

Marginal propensity to save 0.4400 out of each additional dollar of permanent income

What do you estimate for the marginal propensity to save out of each additional dollar of transitory income?

Marginal propensity to save 0.8039 out of each additional dollar of transitory income

Do the magnitudes align with the theory?

It appears that they are saving transitory income more than permanent income. In theory, they should be equal, since its source shouldn't matter in terms of smoothing consumption in general. So in theory, MPC_incperm - MPC_tranperm = 0. In order to do this a bit more formal, we can do a hypothesis test on this difference:

```
linearHypothesis(results, 'inctrans-incperm = 0')
```

```
## Linear hypothesis test
##
## Hypothesis:
## - incperm  + inctrans = 0
##
## Model 1: restricted model
## Model 2: save2 ~ incperm + inctrans + incunexp + p0to5 + p6to11 + p12to17 +
##     p18to64 + p65 + sd1 + sd2 + sd3 + sd4 + year.81 + year.86
##
##   Res.Df        RSS Df Sum of Sq      F  Pr(>F)
## 1   4841 1.9538e+10
## 2   4840 1.9519e+10  1  19460285 4.8254 0.02809 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So given Pr(>F) = 0.02809, means we cannot reject the null hypothesis of PC_incperm - MPC_tranperm = 0, at 0.05 level of significance, but we can reject is at 0.01 level of significance. This might come from imperfections in the data, so more research is required in order to have more structure to say that theory of savings doesn't follows in these scenarios. But in this model, it is safe to say that households save more from transitory income than from permanent income.