Annan and Blattman (2010) Matching Example

```
library(foreign) # Loading normally unsupported file formats
library(tidyverse)
library(Matching) # Finding matches
library(ebal) # Entropy balancing weights
library(cobalt) # Easy love plots.

dat <- read.csv("BlattmanAnnan.csv")

# Let's take a look at the top few rows of the data frame:
head(dat)
```

```
##   abd C_ach C_akw C_ata C_kma C_oro C_pad C_paj C_pal age fthr_ed mthr_ed
## 1   1     0     0     0     1     0     0     0     0  21       7       4
## 2   1     0     0     0     1     0     0     0     0  29       7       4
## 3   1     0     0     0     1     0     0     0     0  23       7       0
## 4   1     0     0     0     1     0     0     0     0  18       0       0
## 5   1     0     0     0     1     0     0     0     0  18       4       4
## 6   1     0     0     0     1     0     0     0     0  25       7       4
##   orphan96 hh_fthr_frm hh_size96 educ distress log.wage
## 1        0           1         9    6 4.000000 8.517393
## 2        0           1         9    3 1.000000 6.370472
## 3        0           1         2    4 3.000000 7.670362
## 4        0           1         6    7 3.333333 8.853808
## 5        0           1         5    7 5.000000 5.525453
## 6        0           0         9    7 5.000000 5.993961
```

MATCHING

Check the covariate balance in the unmatched dataset.

First, before reporting actual balance statistics let's take a look at what seems to predict treatment status. This implies a regression where the dependent variable is treatment status and the predictors are every other covariate we can fit.

What are some of the methods we can predict balance? We could try a multiple regression setting:

```
# We keep covariates to use going forward
covariates <- c("age", "fthr_ed", "mthr_ed", "hh_size96", "orphan96", "C_ach",
                "C_akw", "C_ata", "C_kma", "C_oro", "C_pad", "C_paj", "C_pal")

# We can specify a formula using `paste` function for use later
balance_formula <- as.formula(paste("abd ~ ", paste(covariates, collapse=" + ")))
balance_formula
```

```
## abd ~ age + fthr_ed + mthr_ed + hh_size96 + orphan96 + C_ach +
##     C_akw + C_ata + C_kma + C_oro + C_pad + C_paj + C_pal
```

```
summary(lm(balance_formula, data=dat))
```

```
##
## Call:
## lm(formula = balance_formula, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8657 -0.5045  0.2351  0.3629  0.8009
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.493810   0.100578   4.910 1.13e-06 ***
## age          0.010719   0.003495   3.067  0.00224 **
## fthr_ed     -0.003368   0.005168  -0.652  0.51487
## mthr_ed     -0.007832   0.006295  -1.244  0.21388
## hh_size96   -0.012068   0.004412  -2.735  0.00638 **
## orphan96    -0.018592   0.065032  -0.286  0.77504
## C_ach        0.087929   0.067985   1.293  0.19630
## C_akw        0.227361   0.070341   3.232  0.00128 **
## C_ata       -0.141541   0.068752  -2.059  0.03988 *
## C_kma        0.111132   0.068576   1.621  0.10554
## C_oro       -0.180631   0.077813  -2.321  0.02054 *
## C_pad        0.051055   0.070317   0.726  0.46803
## C_paj        0.120687   0.068834   1.753  0.07997 .
## C_pal              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4678 on 728 degrees of freedom
## Multiple R-squared:  0.0843, Adjusted R-squared:  0.06921
## F-statistic: 5.585 on 12 and 728 DF,  p-value: 3.303e-09
```

We could try a t-test and KS test:

```
t.test(dat$age ~ dat$abd, var.equal=FALSE)
```

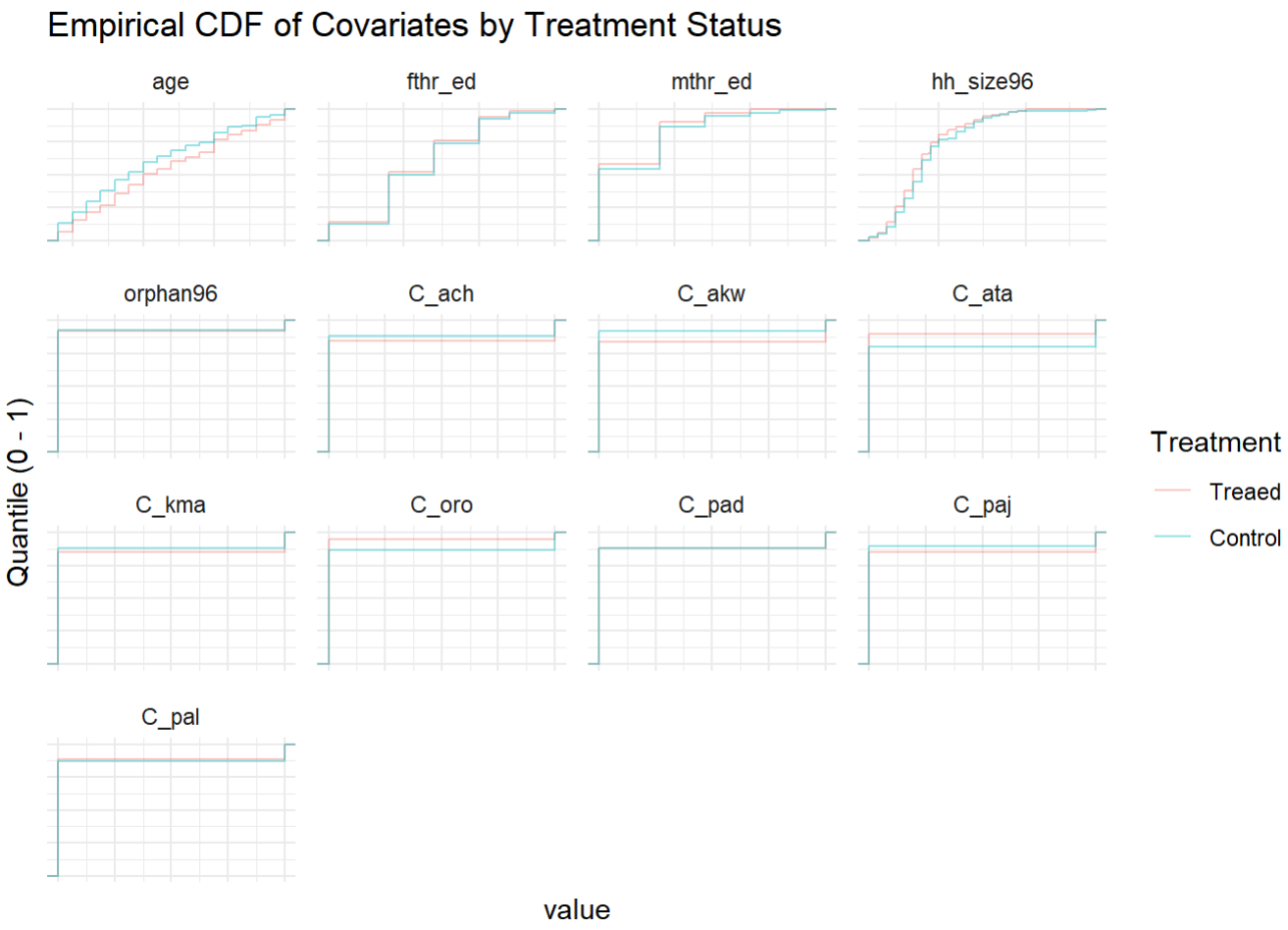
```
##
##  Welch Two Sample t-test
##
## data:  dat$age by dat$abd
## t = -3.239, df = 595.88, p-value = 0.001266
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to
0
## 95 percent confidence interval:
##  -1.9521343 -0.4783922
## sample estimates:
## mean in group 0 mean in group 1
##        20.15054        21.36580
```

```
ks.test(x=dat[dat$abd==1, ]$age,
        y=dat[dat$abd==0, ]$age)
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  dat[dat$abd == 1, ]$age and dat[dat$abd == 0, ]$age
## D = 0.11227, p-value = 0.02491
## alternative hypothesis: two-sided
```

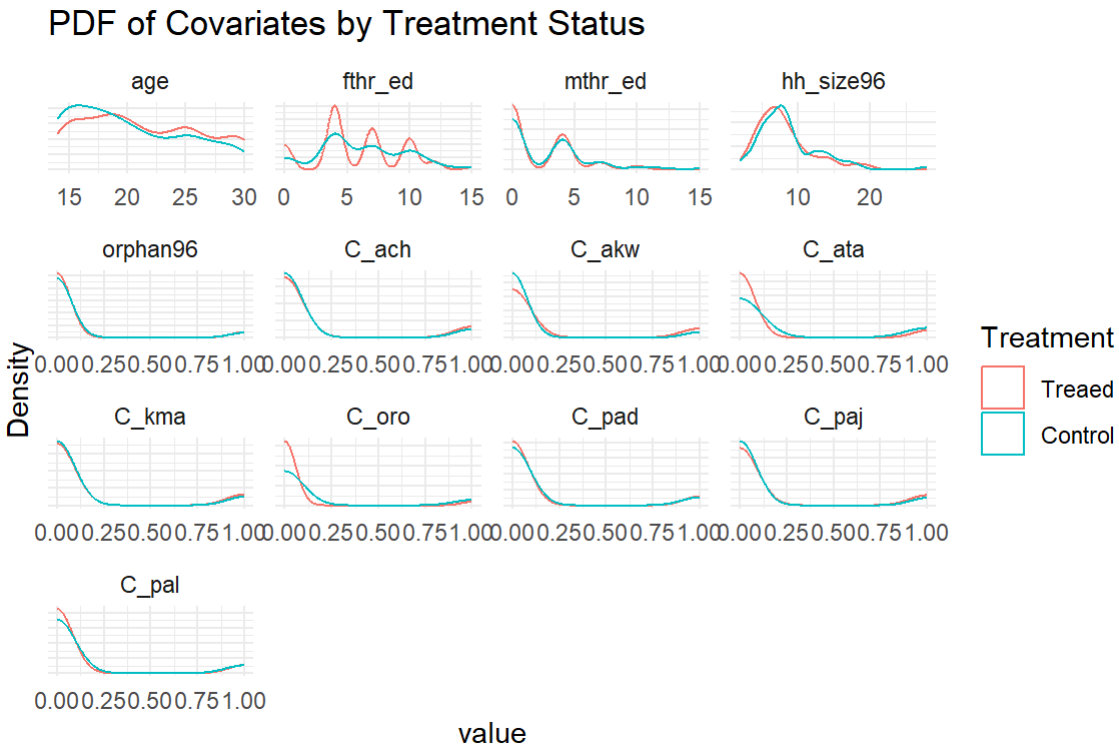
We could plot the eCDF of each covariate between treatment and control,

```
library(reshape2) #for `melt` function
dat %>% dplyr::select(abd, all_of(covariates)) %>%
  mutate(abd = as_factor(ifelse(abd==1, "Treaed", "Control"))) %>%
  melt(all_of(covariates), id.vars = "abd") %>% #make it "long format" for plotting
  ggplot(aes(x = value)) +
  stat_ecdf(aes(col = as_factor(abd)), alpha = .4) +
  facet_wrap(~ variable, scales = "free") +
  labs(title=paste0("Empirical CDF of Covariates by Treatment Status"),
       y="Quantile (0 - 1)", col = "Treatment") +
  theme_minimal() +
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



We could also plot the PDF of each covariate between treatment and control:

```
dat %>% dplyr::select(abd, all_of(covariates)) %>%
  mutate(abd = as_factor(ifelse(abd==1, "Treaed", "Control"))) %>%
  melt(all_of(covariates), id.vars = "abd") %>% #make it "long format" for plotting
  ggplot(aes(x = value)) +
  geom_density(aes(col = as_factor(abd)), alpha = 0.4) +
  facet_wrap(~ variable, scales = "free") +
  labs(title=paste0("PDF of Covariates by Treatment Status"),
       y="Density",col = "Treatment") +
  labs(col = "Treatment") +
  theme_minimal() +
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank())
```



But we can automate this process using `MatchBalance` function in `Matching` package:

```
library(Matching)

# First argument: A formula describing the balance we're checking; the left
# hand side is the dependent variable (in our case, treatment status). The
# right hand side is all the variables you want to check balance on.

# nboots because Matching uses bootstrapping to get SEs on the KS test.
# How many boots? Default is 500, but make it way higher.

mb <- MatchBalance(balance_formula, data=dat, nboots=10000)
```

```
##
## ***** (V1) age *****
## before matching:
## mean treatment........ 21.366
## mean control.......... 20.151
## std mean diff......... 24.242
##
## mean raw eQQ diff..... 1.1971
## med  raw eQQ diff..... 1
## max  raw eQQ diff..... 3
##
## mean eCDF diff........ 0.071486
## med  eCDF diff........ 0.068775
## max  eCDF diff........ 0.11227
##
## var ratio (Tr/Co)..... 1.0428
## T-test p-value........ 0.0012663
## KS Bootstrap p-value.. 0.0101
## KS Naive p-value...... 0.024912
## KS Statistic.......... 0.11227
##
##
## ***** (V2) fthr_ed *****
## before matching:
## mean treatment........ 5.7641
## mean control.......... 6.0681
## std mean diff......... -8.5898
##
## mean raw eQQ diff..... 0.31541
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 4
##
## mean eCDF diff........ 0.016738
## med  eCDF diff........ 0.01962
## max  eCDF diff........ 0.022809
##
## var ratio (Tr/Co)..... 0.9439
## T-test p-value........ 0.2664
## KS Bootstrap p-value.. 0.8597
## KS Naive p-value...... 0.99999
## KS Statistic.......... 0.022809
##
##
## ***** (V3) mthr_ed *****
## before matching:
## mean treatment........ 2.0931
## mean control.......... 2.4946
## std mean diff......... -14.493
##
## mean raw eQQ diff..... 0.39068
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 5
```

```
##
## mean eCDF diff........ 0.021622
## med  eCDF diff........ 0.02282
## max  eCDF diff........ 0.038868
##
## var ratio (Tr/Co)..... 0.72944
## T-test p-value........ 0.085556
## KS Bootstrap p-value.. 0.3617
## KS Naive p-value...... 0.95529
## KS Statistic.......... 0.038868
##
##
## ***** (V4) hh_size96 *****
## before matching:
## mean treatment........ 8.0905
## mean control.......... 8.6953
## std mean diff......... -15.495
##
## mean raw eQQ diff..... 0.64158
## med  raw eQQ diff..... 1
## max  raw eQQ diff..... 8
##
## mean eCDF diff........ 0.029702
## med  eCDF diff........ 0.029395
## max  eCDF diff........ 0.093097
##
## var ratio (Tr/Co)..... 0.803
## T-test p-value........ 0.057539
## KS Bootstrap p-value.. 0.0349
## KS Naive p-value...... 0.098052
## KS Statistic.......... 0.093097
##
##
## ***** (V5) orphan96 *****
## before matching:
## mean treatment........ 0.077922
## mean control.......... 0.075269
## std mean diff......... 0.98877
##
## mean raw eQQ diff..... 0.0035842
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.0013266
## med  eCDF diff........ 0.0013266
## max  eCDF diff........ 0.0026533
##
## var ratio (Tr/Co)..... 1.0308
## T-test p-value........ 0.89531
##
##
## ***** (V6) C_ach *****
```

```
## before matching:
## mean treatment........ 0.15368
## mean control.......... 0.1147
## std mean diff......... 10.798
##
## mean raw eQQ diff..... 0.039427
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.019492
## med  eCDF diff........ 0.019492
## max  eCDF diff........ 0.038984
##
## var ratio (Tr/Co)..... 1.2791
## T-test p-value........ 0.12597
##
##
## ***** (V7) C_akw *****
## before matching:
## mean treatment........ 0.15801
## mean control.......... 0.078853
## std mean diff......... 21.678
##
## mean raw eQQ diff..... 0.078853
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.039578
## med  eCDF diff........ 0.039578
## max  eCDF diff........ 0.079156
##
## var ratio (Tr/Co)..... 1.829
## T-test p-value........ 0.00077667
##
##
## ***** (V8) C_ata *****
## before matching:
## mean treatment........ 0.099567
## mean control.......... 0.19713
## std mean diff......... -32.549
##
## mean raw eQQ diff..... 0.096774
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.048783
## med  eCDF diff........ 0.048783
## max  eCDF diff........ 0.097566
##
## var ratio (Tr/Co)..... 0.56565
## T-test p-value........ 0.00045627
##
```

```
##
## ***** (V9) C_kma *****
## before matching:
## mean treatment........ 0.15152
## mean control.......... 0.11828
## std mean diff......... 9.2594
##
## mean raw eQQ diff..... 0.032258
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.016618
## med  eCDF diff........ 0.016618
## max  eCDF diff........ 0.033236
##
## var ratio (Tr/Co)..... 1.231
## T-test p-value........ 0.19421
##
##
## ***** (V10) C_oro *****
## before matching:
## mean treatment........ 0.051948
## mean control.......... 0.1362
## std mean diff......... -37.924
##
## mean raw eQQ diff..... 0.086022
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.042126
## med  eCDF diff........ 0.042126
## max  eCDF diff........ 0.084253
##
## var ratio (Tr/Co)..... 0.41801
## T-test p-value........ 0.0002848
##
##
## ***** (V11) C_pad *****
## before matching:
## mean treatment........ 0.12121
## mean control.......... 0.12186
## std mean diff......... -0.19946
##
## mean raw eQQ diff..... 0
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 0
##
## mean eCDF diff........ 0.00032584
## med  eCDF diff........ 0.00032584
## max  eCDF diff........ 0.00065168
##
## var ratio (Tr/Co)..... 0.99397
```

```
## T-test p-value........ 0.97906
##
##
## ***** (V12) C_paj *****
## before matching:
## mean treatment........ 0.15152
## mean control.......... 0.10394
## std mean diff......... 13.254
##
## mean raw eQQ diff..... 0.046595
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.023786
## med  eCDF diff........ 0.023786
## max  eCDF diff........ 0.047572
##
## var ratio (Tr/Co)..... 1.3783
## T-test p-value........ 0.055286
##
##
## ***** (V13) C_pal *****
## before matching:
## mean treatment........ 0.11255
## mean control.......... 0.12903
## std mean diff......... -5.2082
##
## mean raw eQQ diff..... 0.017921
## med  raw eQQ diff..... 0
## max  raw eQQ diff..... 1
##
## mean eCDF diff........ 0.0082391
## med  eCDF diff........ 0.0082391
## max  eCDF diff........ 0.016478
##
## var ratio (Tr/Co)..... 0.88753
## T-test p-value........ 0.5087
##
##
## Before Matching Minimum p.value: 0.0002848
## Variable Name(s): C_oro  Number(s): 10
```

There are two issues here. First, Matching does not obey R convention – rather than quietly returning the data of interest into an object for future printing, Matching will automatically print a ton of output. Second, the output is ugly. There are a variety of packages you can use to make it not ugly, one of which is `ebal`. Let's use `baltest.collect` from `ebal`.

```
library(ebal)
# baltest.collect takes three arguments; first, the Matching object.
# Second, which variables do you care about balance on? (Why does this not
# default to every right-hand side variable? Who knows.) How do we specify the
# variable names? As a vector of character strings -- annoying! Let's quickly
# pop them out of the formula from before:
covariates
```

```
##  [1] "age"       "fthr_ed"   "mthr_ed"   "hh_size96" "orphan96"  "C_ach"
##  [7] "C_akw"     "C_ata"     "C_kma"     "C_oro"     "C_pad"     "C_paj"
## [13] "C_pal"
```

```
# Why do we do [-1]? The first variable is the treatment status, and we don't
# need "balance" on that.

# Third, do you want to see balance before any Matching attempt, or after?
# We want to see before, so we use after = FALSE
balance_test <- baltest.collect(mb,
                        var.names = covariates,
                        after = FALSE)

# balance_test includes a bunch of stuff we don't care about. Let's discard down:
balance_test <- balance_test[, c("mean.Tr", "mean.Co", "T pval", "KS pval")]

# Let's make a nice kable table:
library(knitr)
library(kableExtra) #for aesthetics
balance_test %>%
  kable(booktabs = T, digits=3, align = rep('c', 4),
        col.names = c("Treatment Mean", "Control Mean", "T p-value", "Ks p-value")) %>%
  kable_styling(full_width = F, position = "center")
```

	Treatment Mean	Control Mean	T p-value	Ks p-value
age	21.366	20.151	0.001	0.010
fthr_ed	5.764	6.068	0.266	0.860
mthr_ed	2.093	2.495	0.086	0.362
hh_size96	8.090	8.695	0.058	0.035
orphan96	0.078	0.075	0.895	NA
C_ach	0.154	0.115	0.126	NA
C_akw	0.158	0.079	0.001	NA
C_ata	0.100	0.197	0.000	NA

	Treatment Mean	Control Mean	T p-value	Ks p-value
C_kma	0.152	0.118	0.194	NA
C_oro	0.052	0.136	0.000	NA
C_pad	0.121	0.122	0.979	NA
C_paj	0.152	0.104	0.055	NA
C_pal	0.113	0.129	0.509	NA

What do we think of this balance?

Matching

Let's do some matching. Let's try to extract the ATT – so we'll use our real treatment observations, matched control observations, and discard the rest. We'll do a simple match; one match per treated unit. And we'll use replacement (so the same control unit can be picked as a match for more than one treated unit)

We also need to specify which variables we want to match on exactly. First we'll subset to the variables we care about, and then we'll take a peek at those variable to see whether they're likely to need exact matches:

```
outcome <- dat$educ
treated <- dat$abd
X <- dat[, covariates]
str(X)
```

```
## 'data.frame':    741 obs. of  13 variables:
##  $ age      : int  21 29 23 18 18 25 23 29 28 22 ...
##  $ fthr_ed  : int  7 7 7 0 4 7 7 7 7 7 ...
##  $ mthr_ed  : int  4 4 0 0 4 4 0 0 4 4 ...
##  $ hh_size96: num  9 9 2 6 5 9 11 11 5 8 ...
##  $ orphan96 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_ach    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_akw    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_ata    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_kma    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ C_oro    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_pad    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_paj    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ C_pal    : int  0 0 0 0 0 0 0 0 0 0 ...
```

We don't have any variables that are obviously "continuous", but a few look to me like they can take many possible values: `age` and `hh_size96`. Let's just briefly confirm that:

```
length(unique(X$age))
```

```
## [1] 17
```

```
length(unique(X$hh_size96))
```

```
## [1] 23
```

We've only got 741 observations so with 23 possible values of household size and 17 of age we probably don't want exact matching for those. The rest all look like they're binary or only take a few values, so for them we'll do exact matching.

```
exact_matches <- ifelse(colnames(X) %in% c("age", "hh_size96"), FALSE, TRUE)
```

How does `ifelse` work? It takes three arguments and outputs a vector. The first argument is a statement to evaluate. We're going to map every single column name, and check for each of them are they "in" the vector `c("age", "hh_size96")`. If they are, use the value `FALSE` for that item in the vector. If they are not, use the value `TRUE` for that item in the vector.

Now let's do the matching.

```
match_model_1 <- Match(Y = outcome,
                       Tr = treated,
                       X = X,
                       M = 1,
                       exact = exact_matches,
                       estimand = "ATT")
```

What are the arguments here? Well, it's obvious you need `Y` to estimate a treatment effect. It's obvious you need `Tr` so it knows which units are treated and which are controlled. `X` is the covariates to match on. `M` tells you how many matches you want for each treatment unit. `exact` says, for each covariate, if you want an exact match or not. `estimand` tells `Matching` your estimand: in our case we want to keep the real treated units and find matching control units.

We can also easily do bias adjustment:

```
match_model_2 <- Match(Y = outcome,
                       Tr = treated,
                       X = X,
                       M = 1, # one-to-one matching
                       exact = exact_matches,
                       estimand = "ATT",
                       BiasAdjust = TRUE)
```

Bias Adjustment compensates for the fact that since we can't find exact matches, the effect size will be a little skewed (imagine there's a relationship between X and Y, but the Xs of treated units are all slightly higher than the Xs of the control units they are matched with – or vice versa – wouldn't this bias our effect estimate?)

Let's now check what `Matching` actually gives us:

```
ls(match_model_1)
```

```
##  [1] "caliper"           "ecaliper"          "est"
##  [4] "est.noadj"         "estimand"          "exact"
##  [7] "index.control"     "index.dropped"     "index.treated"
## [10] "MatchLoopC"        "mdata"             "ndrops"
## [13] "ndrops.matches"    "nobs"              "orig.nobs"
## [16] "orig.treated.nobs" "orig.wnobs"        "se"
## [19] "se.cond"           "se.standard"       "version"
## [22] "weights"           "wnobs"
```

This tells us what is contained in the `Match` object; We see a few we might like to check out:

```
match_model_1$est
```

```
##           [,1]
## [1,] -0.8005618
```

```
match_model_1$se # SEs from Abadie-Imbens
```

```
## [1] 0.2238294
```

We can also check what got matched with what:

```
match_model_1$orig.nobs #number of units in original data
```

```
## [1] 741
```

```
length(match_model_1$index.treated) #number of treated units in matched data
```

```
## [1] 366
```

```
length(match_model_1$index.control) #number of control units in matched data
```

```
## [1] 366
```

```
length(match_model_1$index.dropped) #number of dropped units during matching
```

```
## [1] 106
```

```
length(unique(match_model_1$index.control)) #number of unique control units (some duplicates!)
```

```
## [1] 162
```

We can set the number of matched controls per treated, using `M` option. How many control matches do we have to include? We have this trade-off: small M (e.g., one-to-one matching) means small matched-sample sizes from which we estimate our quantity of interest, whereas large M may give more bad matches (since we allow for worse matches to be included).

As can be seen below, we increased the number of matched control units by 302 compared to one-to-one match ($M = 1$), but only 15 more unique controls used. It suggests that by setting `M=2` , we allowed for more bad

```
match_model_3 <- Match(Y = outcome,
                       Tr = treated,
                       X = X,
                       M = 2, #how many control units we match to each treatment unit: now 2 ins
tead of 1
                       exact = exact_matches,
                       estimand = "ATT",
                       BiasAdjust = TRUE)

length(match_model_3$index.control) - length(match_model_1$index.control)
```

```
## [1] 302
```

```
length(unique(match_model_3$index.control)) - length(unique(match_model_1$index.control))
```

```
## [1] 15
```

And with the bias-adjusted estimates:

```
match_model_2$est
```

```
##            [,1]
## [1,] -0.8134043
```

```
match_model_2$se
```

```
## [1] 0.2246478
```

Compare to “naive” estimate:

```
summary(lm(educ ~ abd, data = dat))
```

```
##
## Call:
## lm(formula = educ ~ abd, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4158 -1.8203 -0.4158  2.1797  8.5842
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.4158     0.1722  43.072  < 2e-16 ***
## abd          -0.5954     0.2180  -2.731  0.00647 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.876 on 739 degrees of freedom
## Multiple R-squared:  0.00999,    Adjusted R-squared:  0.00865
## F-statistic: 7.457 on 1 and 739 DF,  p-value: 0.00647
```

Balance After Matching

Normally, before you even look at results you'd examine balance post-matching. What we're trying to do here is make sure that the Matching process actually resulted in good balance. And we can include variables that we didn't match on, if we want.

Note, you can check balance on things that weren't matched on to make sure they got balanced too.

```
match_balance_after <- MatchBalance(match.out = match_model_1,
                                    formul = balance_formula,
                                    data = dat,
                                    print.level = 0,
                                    nboots = 10000)

balance_test_after <- baltest.collect(match_balance_after,
                                      var.names = covariates,
                                      after = TRUE)[, c("mean.Tr", "mean.Co",
                                                        "T pval", "KS pval")]

balance_test_after %>%
  kable(booktabs = T, digits=3, align = rep('c', 4),
        col.names = c("Treatment Mean", "Control Mean", "T p-value", "Ks p-value")) %>%
  kable_styling(full_width = F, position = "center")
```

	Treatment Mean	Control Mean	T p-value	Ks p-value
age	21.441	20.503	0.000	0.004
fthr_ed	5.174	5.174	1.000	1.000
mthr_ed	1.421	1.421	1.000	1.000

	Treatment Mean	Control Mean	T p-value	Ks p-value
hh_size96	7.876	7.876	0.997	0.180
orphan96	0.014	0.014	1.000	NA
C_ach	0.169	0.169	1.000	NA
C_akw	0.107	0.107	1.000	NA
C_ata	0.107	0.107	1.000	NA
C_kma	0.138	0.138	1.000	NA
C_oro	0.059	0.059	1.000	NA
C_pad	0.124	0.124	1.000	NA
C_paj	0.169	0.169	1.000	NA
C_pal	0.129	0.129	1.000	NA

Balance with Covariate-Balancing Weights

Instead of choosing a 1-to-1 match, we could choose weights on the control units in order to ensure that the weighted average of each X for the controls equals the unweighted average among the treated. We could similarly get weighted averages for other numbered moments (X^2, X^3, etc.) This would give us exact balance on those moments.

We can do this with `ebal`:

```
library(ebal)

# ebal needs us to remove one of the fixed effect dummies.
X_ebal <- X[, -13]
variable_names_ebal <- covariates[-13]

ebal_model <- ebalance(treated, X=X_ebal)
```

```
## Converged within tolerance
```

A quirk of `ebal` is that it returns a weight vector for only the control units, not the treated units. Which means that if we need a weight vector for, say, a weighted regression, we need to fill in those units. We do this by creating a vector of 1s with length equal to the number of the observations, and then override the control observations with the weights from `ebal`.

```
# Make the vector of 1 weights:
ebal_weights <- rep(1, length(outcome))

# Now override the control weights:
ebal_weights[treated == 0] <- ebal_model$w
```

Now let's use these weights with `MatchBalance`

```
ebal_balance <- MatchBalance(treated ~ as.matrix(X),
                             weights = ebal_weights,
                             print.level = 0,
                             nboots = 10000)

# Another weird quirk: we use after = FALSE, because we're not using MatchBalance
# to do the matching, we're just using the weights from ebal.
ebal_table <- baltest.collect(matchbal.out = ebal_balance,
                              var.names = colnames(X),
                              after = FALSE)[, c("mean.Tr", "mean.Co",
                                                 "T pval", "KS pval")]

# Now output the table
bind_cols(balance_test, ebal_table) %>%
  kable(booktabs = T, digits=3, align = rep('c', 8),
        col.names = rep(c("Treatment Mean", "Control Mean", "T p-value", "Ks p-value"), 2)) %>%
  kable_styling(full_width = F, position = "center")
```

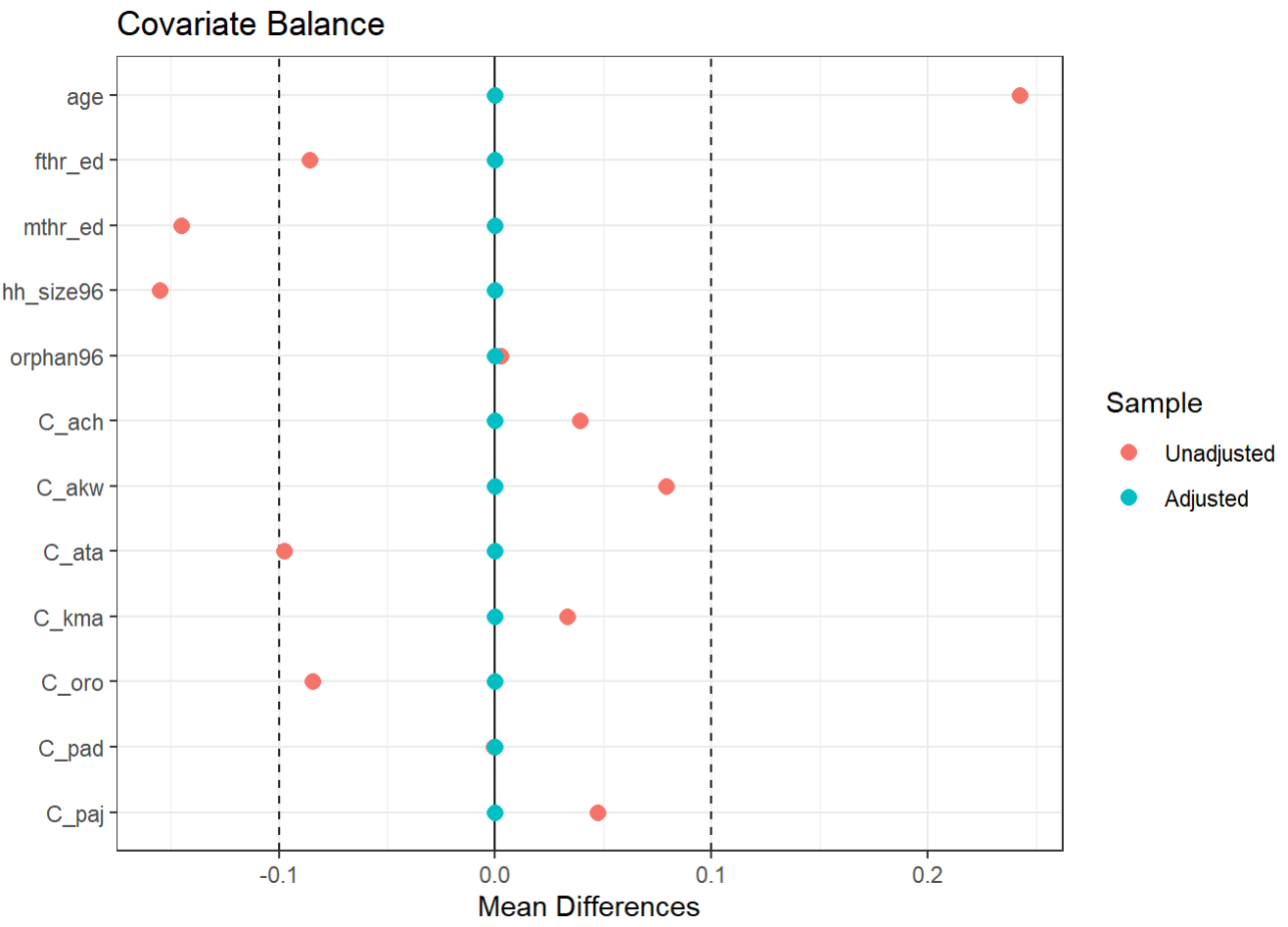
Treatment Mean	Control Mean	T p-value	Ks p-value	Treatment Mean	Control Mean	T p-value	Ks p-value
21.366	20.151	0.001	0.010	21.366	21.366	1.000	0.010
5.764	6.068	0.266	0.860	5.764	5.764	1.000	0.861
2.093	2.495	0.086	0.362	2.093	2.093	1.000	0.363
8.090	8.695	0.058	0.035	8.090	8.090	1.000	0.033
0.078	0.075	0.895	NA	0.078	0.078	1.000	NA
0.154	0.115	0.126	NA	0.154	0.154	1.000	NA
0.158	0.079	0.001	NA	0.158	0.158	1.000	NA
0.100	0.197	0.000	NA	0.100	0.100	1.000	NA
0.152	0.118	0.194	NA	0.152	0.152	1.000	NA
0.052	0.136	0.000	NA	0.052	0.052	0.997	NA
0.121	0.122	0.979	NA	0.121	0.121	1.000	NA

Treatment Mean	Control Mean	T p-value	Ks p-value	Treatment Mean	Control Mean	T p-value	Ks p-value
0.152	0.104	0.055	NA	0.152	0.152	1.000	NA
0.113	0.129	0.509	NA	0.113	0.113	1.000	NA

Another quirk (this is starting to get frustrating) – the KS test p-values are incorrect when `MatchBalance` is used with weights.

We can plot the standardized imbalance (and the subsequent balance) using the `love.plot` command in the `cobalt` package. `cobalt` is designed to work with the output from a wide variety of matching and weighting packages.

```
library(cobalt)

# bal.tab() is a function which takes as arguments:
# ebal <-- an ebalance object (in our case, ebal_model)
# treat <-- a vector of treatment statuses
# covs <-- a matrix or data frame of Xes.
love.plot(bal.tab(x = ebal_model,
                  treat = treated,
                  covs = X_ebal),
          threshold = 0.1, theme=theme_bw())
```



Nice looking plot, almost no code. And as we see, `ebal` produces exact moment matching, ensuring equal means across groups (note: it might do this by putting an extremely high weight on one or more of the observations). Let's just quickly check the summary stats of our weights:

```
summary(ebal_weights)
```

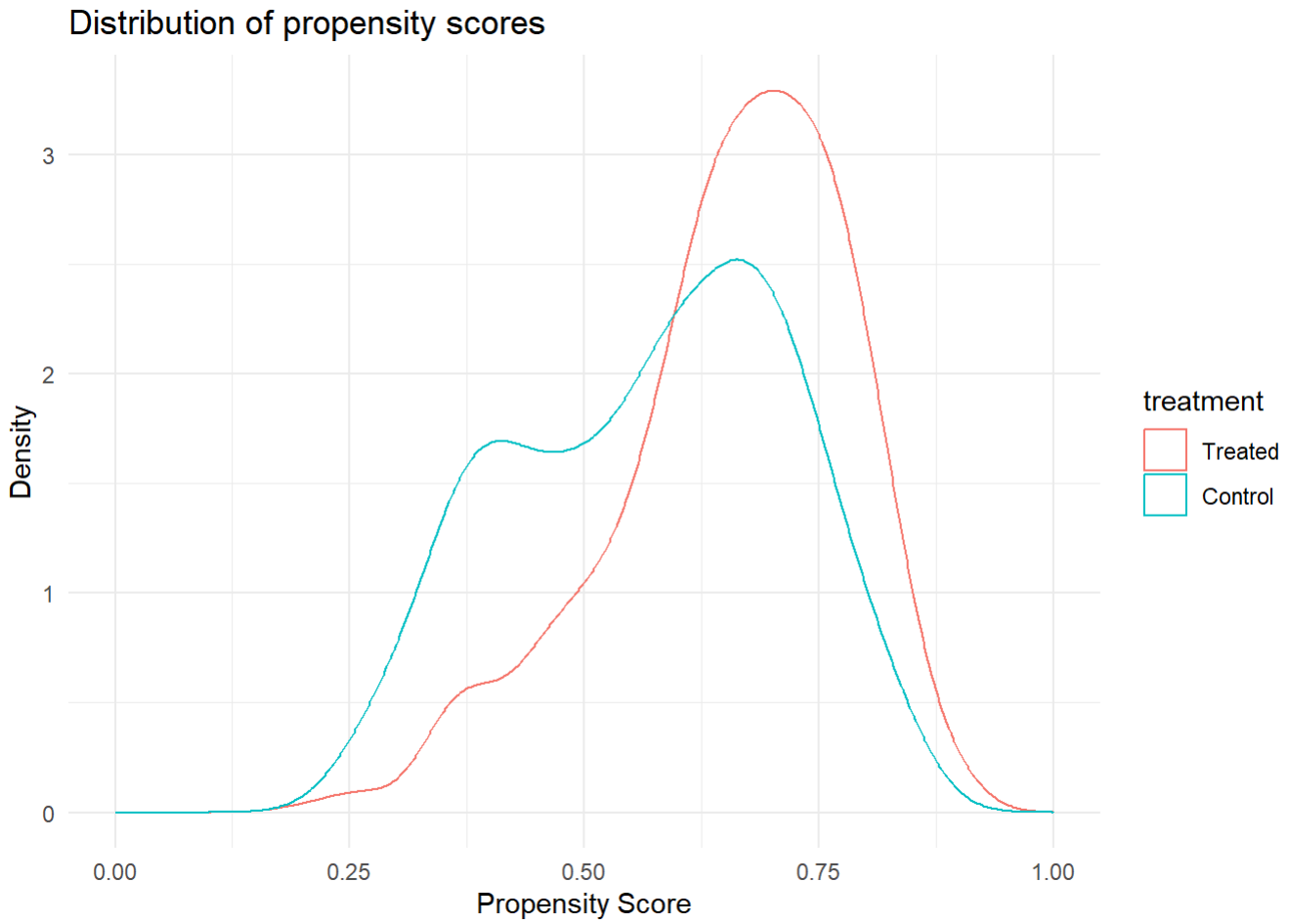
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2917  1.0000  1.0000  1.2470  1.0150  5.4471
```

The maximum weight doesn't seem too bad here, and fewer than a quarter of observations had weights much above 1.

Propensity Scores

Another approach, as we saw in the previous week, is that we can estimate propensity scores for treatment. We can either match units on propensity score (look for control units with similar propensity scores to the treatment), or we can use the propensity scores to generate IPW weights and weight on them. First, let's get the scores using logistic regression.

```
logit_fit <- glm(balance_formula, data = dat, family = binomial(link = logit))

# Extract these using predicted probabilities or with the direct fit object:
pi.out <- logit_fit$fitted.values

# Plot
ggplot(tibble(ps = pi.out, treatment = as_factor(ifelse(dat$abd==1, "Treated", "Control"))),
       aes(x=ps, col=treatment)) +
  geom_density() +
  labs(title="Distribution of propensity scores", y="Density", x="Propensity Score") +
  xlim(c(0,1)) +
  theme_minimal()
```



We can see that these distributions do not look quite the same. Your first intuition is likely to say “well, the treated look like they’re more likely to take the treatment than the control units, isn’t that normal?”. But this is a sign of imbalance on its own. What we’d instead like is a sense that units, no matter what their propensity score, are in balanced proportions in the treated and control groups.

So, we’re going to use our two methods. First, let’s use `Match` to calculate the ATT with propensity score as the covariate:

```
p_score_match <- Match(Y = outcome,
                       Tr = treated,
                       X = pi.out,
                       M = 1,
                       estimand = "ATT")
summary(p_score_match)
```

```
##
## Estimate...  -0.67063
## AI SE......  0.32085
## T-stat.....  -2.0902
## p.val......  0.036599
##
## Original number of observations..............  741
## Original number of treated obs...............  462
## Matched number of observations...............  462
## Matched number of observations  (unweighted).  552
```

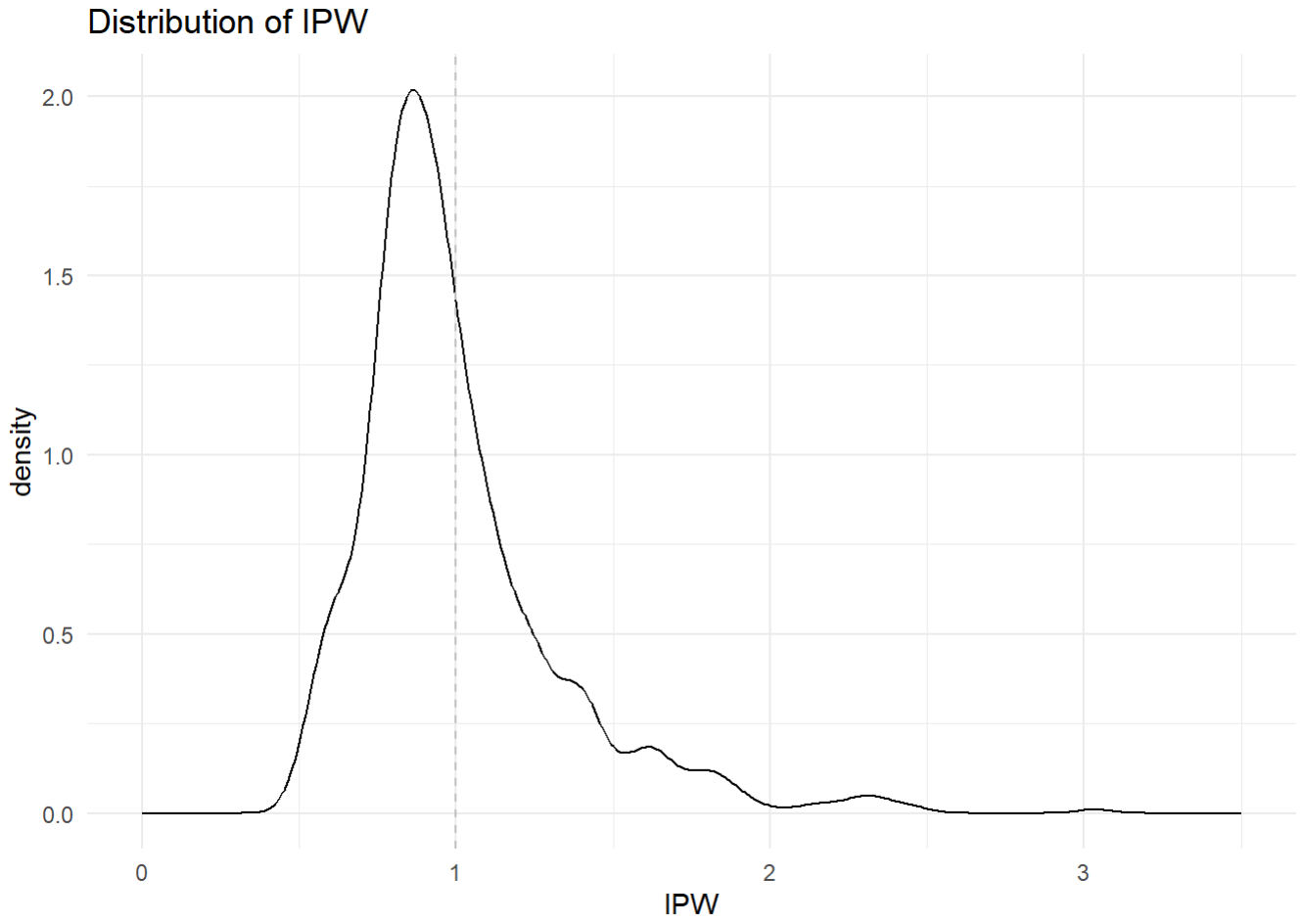
And checking the balance:

```
p_score_match_balance <- MatchBalance(match.out = p_score_match,
                                      formul = balance_formula,
                                      data = dat,
                                      print.level = 0)

balance_table_ps_match <- baltest.collect(p_score_match_balance,
                                          var.names = covariates,
                                          after = TRUE)[,
                                                        c("mean.Tr", "mean.Co",
                                                          "T pval", "KS pval")]

balance_table_ps_match %>%
  kable(booktabs = T, digits=3, align = rep('c', 4),
        col.names = c("Treatment Mean", "Control Mean", "T p-value", "Ks p-value")) %>%
  kable_styling(full_width = F, position = "center")
```

	Treatment Mean	Control Mean	T p-value	Ks p-value
age	21.366	21.042	0.269	0.186
fthr_ed	5.764	5.598	0.478	0.074
mthr_ed	2.093	2.405	0.108	0.230
hh_size96	8.090	8.025	0.795	0.026
orphan96	0.078	0.112	0.074	NA
C_ach	0.154	0.166	0.598	NA
C_akw	0.158	0.168	0.625	NA
C_ata	0.100	0.090	0.548	NA
C_kma	0.152	0.155	0.872	NA
C_oro	0.052	0.051	0.952	NA
C_pad	0.121	0.089	0.101	NA
C_paj	0.152	0.154	0.910	NA
C_pal	0.113	0.127	0.453	NA

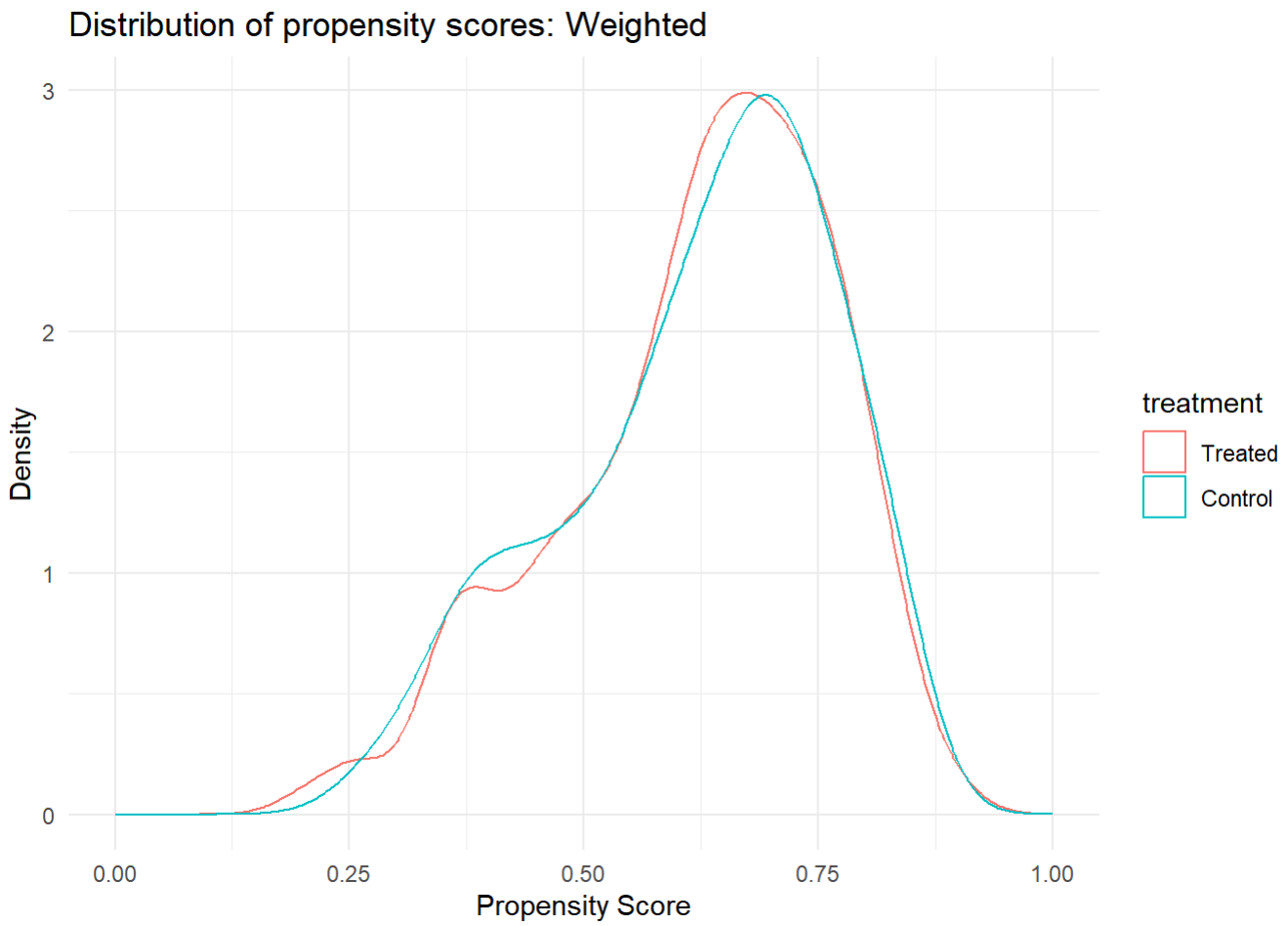
Final alternative; using stabilized inverse propensity score weights.

```
probability_treated <- mean(treated)
IPW <- ifelse(treated==1, probability_treated/pi.out, (1 - probability_treated)/(1 - pi.out))

ggplot(as_tibble(IPW), aes(x=value)) +
  geom_density() + geom_vline(xintercept=1, linetype="dashed", col="gray") +
  labs(title = "Distribution of IPW", x="IPW") + xlim(c(0, 3.5)) + theme_minimal()
```



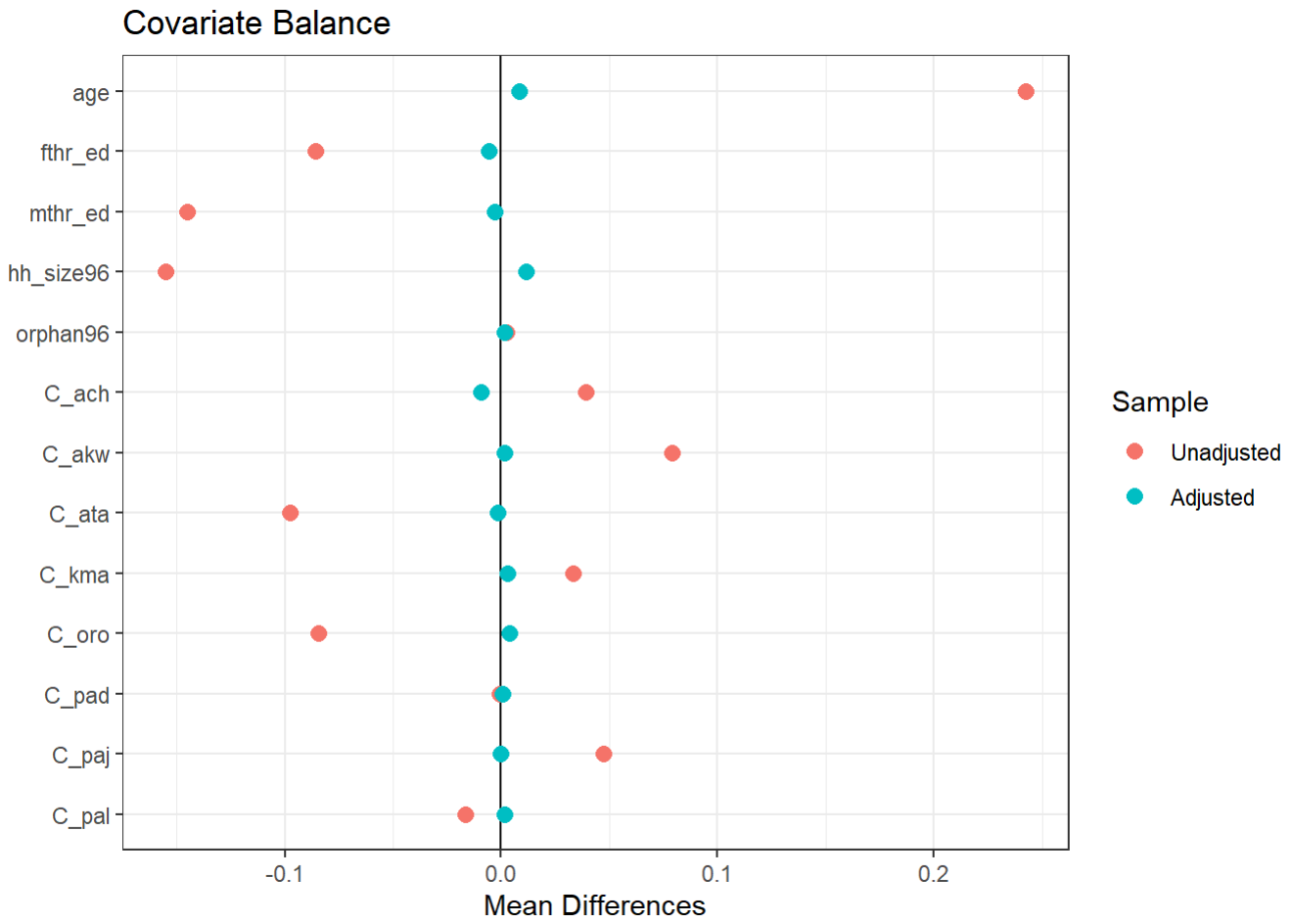
See how distribution of propensity scores has changed:

```
plot_df <- tibble(ps = pi.out, treatment = as_factor(ifelse(dat$abd==1, "Treated", "Control")),
ipw = IPW) %>%
  group_by(treatment) %>%
  mutate(weight = ipw/sum(ipw))

ggplot(plot_df, aes(x=ps, col=treatment)) +
  geom_density(aes(weight=weight)) +
  labs(title="Distribution of propensity scores: Weighted", y="Density", x="Propensity Score") +
  xlim(c(0,1)) +
  theme_minimal()
```



Using IPWs, much closer balance on propensity scores. Remember to check the balance again. And one final love plot, this time with the IPW weights. This time, since we are no longer using an `ebalance` object, we do a slightly different set of arguments for `love.plot`. We pass it a formula, our entire data frame, a list of treatment status, that we are using weighting, our weights, and that the estimand is ATT:

```
love.plot(bal.tab(balance_formula,
                  data = dat,
                  treat.list = treated,
                  method = "weighting",
                  weights = IPW,
                  estimand = "ATT"), theme=theme_bw())
```



Again, big improvement in balance. And now how do we use these weights to estimate an effect? Weighted least squares regression:

```
#unweighted
summary(lm(educ ~ abd, data = dat))
```

```
##
## Call:
## lm(formula = educ ~ abd, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4158 -1.8203 -0.4158  2.1797  8.5842
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.4158     0.1722  43.072  < 2e-16 ***
## abd          -0.5954     0.2180  -2.731  0.00647 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.876 on 739 degrees of freedom
## Multiple R-squared:  0.00999,    Adjusted R-squared:  0.00865
## F-statistic: 7.457 on 1 and 739 DF,  p-value: 0.00647
```

```r
#weighted by ipw
summary(lm(educ ~ abd, weight = IPW, data = dat))
```

```
##
## Call:
## lm(formula = educ ~ abd, data = dat, weights = IPW)
##
## Weighted Residuals:
##    Min     1Q Median     3Q    Max
## -9.017 -1.780 -0.460  2.002 10.765
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.4992     0.1739  43.132  < 2e-16 ***
## abd          -0.7255     0.2201  -3.296  0.00103 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.904 on 739 degrees of freedom
## Multiple R-squared:  0.01448,    Adjusted R-squared:  0.01315
## F-statistic: 10.86 on 1 and 739 DF,  p-value: 0.001029
```