R Notebook: Compare Groups

Contents

Content with * is optional.

This tutorial will need the following packages:

- Lattice
- multcomp
- dplyr

Marketing analysts often investigate differences between groups.

- Group data by people: Do men or women subscribe to our service at a higher rate? Which demographic segment can best afford our product? Does the product appeal more to homeowners or renters?
- Group data by geography: Does Region A perform better than Region B?
- Group data by experiment manipulation: Did Ad Version A generate a higher conversion rate than Ad Version B.
- Group data by time: Did same-store sales increase after a promotion such as a mailer or a sale?

In all such cases, we compare one data group to another to identify an effect. This tutorial examines the kinds of comparisons that often arise in marketing.

1 Descriptives by group

We use data with consumer segmentation. We are interested in the effect of two newly developed ads (narrative vs. informative) and have collected data from N =300 respondents along with their *age, gender, number of children*, whether they *like* the ads, and *how much time* they spent on the page. In this data, each respondent is from one of our four consumer segments: *Suburb mix, Urban hip, Travelers, or Moving up.*

```
ad.df <- read.csv("Data_Compare_Groups.csv", stringsAsFactors = TRUE)
summary(ad.df)
```

```
##      condition        like     seconds_spent        age           gender
##  control  :159   likeNo :260   Min.   :  0.69   Min.   :19.00   Female:157
##  treatment:141   likeYes: 40   1st Qu.: 39.66   1st Qu.:33.00   Male  :143
##                                Median : 52.02   Median :39.50
##                                Mean   : 50.98   Mean   :41.17
##                                3rd Qu.: 61.41   3rd Qu.:48.00
##                                Max.   :114.28   Max.   :80.00
##       kids            segment
##  Min.   :0.00   Moving up : 70
##  1st Qu.:0.00   Suburb mix:100
##  Median :1.00   Travelers : 80
##  Mean   :1.27   Urban hip : 50
##  3rd Qu.:2.00
##  Max.   :7.00
```

```
str(ad.df)
```

```
## 'data.frame':    300 obs. of  7 variables:
##  $ condition    : Factor w/ 2 levels "control","treatment": 1 2 2 1 2 2 1 1 1 2 ...
##  $ like         : Factor w/ 2 levels "likeNo","likeYes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ seconds_spent: num  49.5 35.5 44.2 81 79.3 ...
##  $ age          : int  47 31 43 37 41 43 38 28 44 35 ...
##  $ gender       : Factor w/ 2 levels "Female","Male": 2 2 2 1 1 2 2 2 1 1 ...
##  $ kids         : int  2 1 0 1 3 4 3 0 1 0 ...
##  $ segment      : Factor w/ 4 levels "Moving up","Suburb mix",..: 2 2 2 2 2 2 2 2 2 2 ...
```

We are interested in how measures such as *seconds spent* and *like* vary for two versions of ads.

An ad hoc way to do this is with data frame indexing: find the rows that match some criterion and then take the mean of another statistic. For instance, to find out the mean total seconds spent for the narrative ads (treatment):

```
mean(ad.df$seconds_spent[ad.df$condition == "treatment"])
```

```
## [1] 55.01809
```

We could further narrow the cases to *Moving up* respondents who are in the treatment condition:

```
mean(ad.df$seconds_spent[ad.df$condition == "treatment" & ad.df$segment == "Moving up"])
```

```
## [1] 50.21652
```

1.1 *aggregate()*

When you want to find values for multiple groups, a more general way to do this is with *aggregate()*.

```
aggregate(ad.df$seconds_spent, list(ad.df$condition), mean)
```

1.2 Basic formula syntax

R provides a standard way to describe relationships among variables through *formula* specification. A formula uses the tilde (~) operator to separate *response variable* on the left from *explanatory variables* on the right. The basic form is:

y ~ x (simple formula)

This is used in many contexts in R, where the meaning of *response* and *explanatory* depends on the situation. For instance, in linear regression, the simple formula above would model *y* as a linear function of *x*. In this case of the *aggregate()* command, the effect is to aggregate *y* according to the levels of *x*.

Let us see it in practice. Instead of $aggregate(ad.dfseconds_spent, list(ad.df condition), mean)$, a general form is *aggregate(formula, data, FUN)*. In our example, we tell R to "take *seconds_spent* by *condition* within the data set *ad.df*, and apply *mean* function to each group".

```
aggregate(seconds_spent ~ condition, data = ad.df, mean)
```

1.3 Descriptives for two-way groups

A common task in marketing is cross-tabulation, separating customers into groups according to two (or more) factors. Formula syntax makes it easy to compute a cross tab just by specifying multiple explanatory variables:

y ~ x1 + x2 + ... (Multiple variable formula)

1.3.1 Means

Using this format with *aggregate()*, we can write:

```
aggregate(seconds_spent ~ segment + condition, data = ad.df, mean)
```

We now have a separate group for each combination of *segment* and *condition* and can begin to see how *seconds_spent* is related to both the *segment* and the *condition* variables.

We can assign the result to a data frame object and index:

```
agg.data <- aggregate(seconds_spent ~ segment + condition, data = ad.df, mean)
```

The *aggregate()* command allows us to compute functions of continuous variables, such as the *mean* of *seconds_spent* or *like* for any combination of factors (*segment*, *condition*, and so forth). This is a common task in marketing research that companies specialize in producing cross tabs.

1.3.2 Frequencies

We also want to know the *frequency* with which different combinations of *condition* and *like* occur. We can compute frequencies using *table(factor1, factor2, . . .)* to obtain one-way or multi-way counts:

```
table(ad.df$condition, ad.df$like)
```

```
##
##               likeNo likeYes
##   control       137      22
##   treatment     123      18
```

```
table(ad.df$segment, ad.df$like)
```

```
##
##                likeNo likeYes
##   Moving up       56      14
##   Suburb mix      94       6
##   Travelers       70      10
##   Urban hip       40      10
```

We can add together the counts to find their total. For instance, *kids* is a count variable; if a respondent reported 3 kids, that is a count of 3, and we could add together the counts to get the total number of children reported in each segment. We can use *aggregate(. . . , sum)*:

```
aggregate(kids ~ segment, data = ad.df, sum)
```

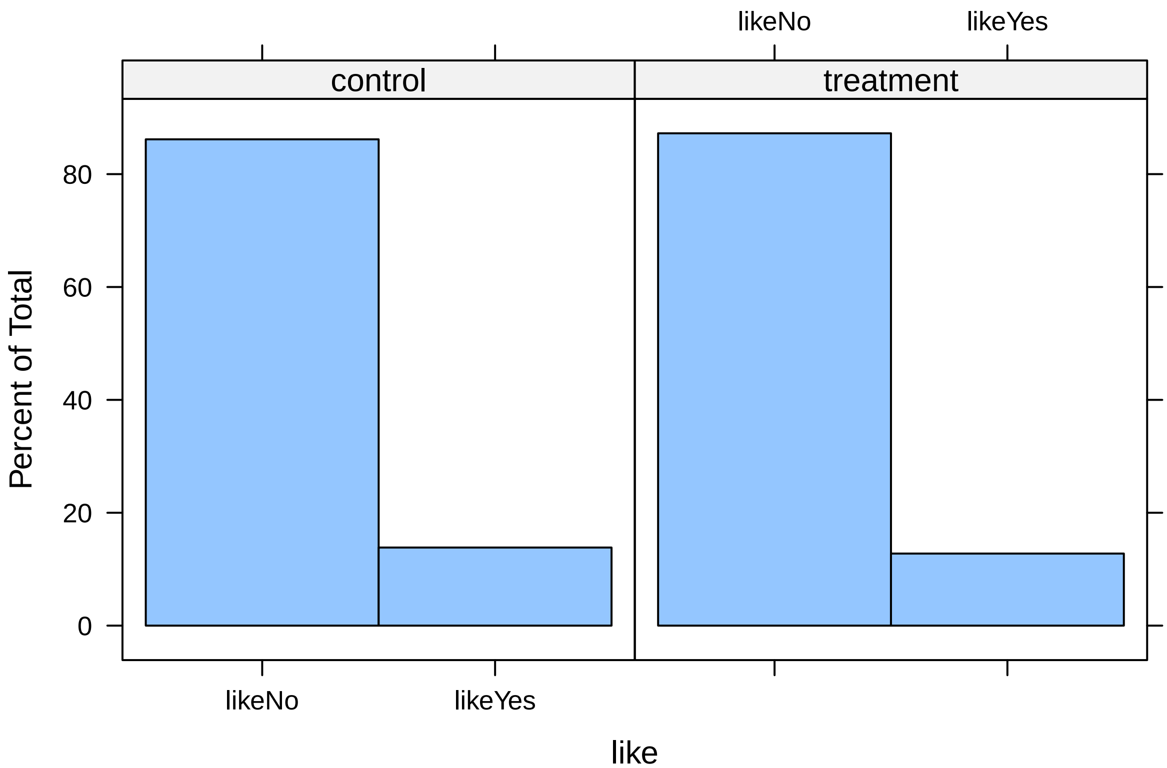
The result shows that "Urban up" respondents reported a total of 55 kids, while the "Traveller" reported none.

2 Visualization by group

2.1 Visualizing frequencies and proportions

The *Lattice* package provides a useful solution: *histogram (formula, data, type)*. It understands formula notions, including *conditioning* (" | ") on a factor, which means separating the plot into multiple panes based on the factor.

```
library(lattice)
histogram(~ like | condition, data = ad.df)
```

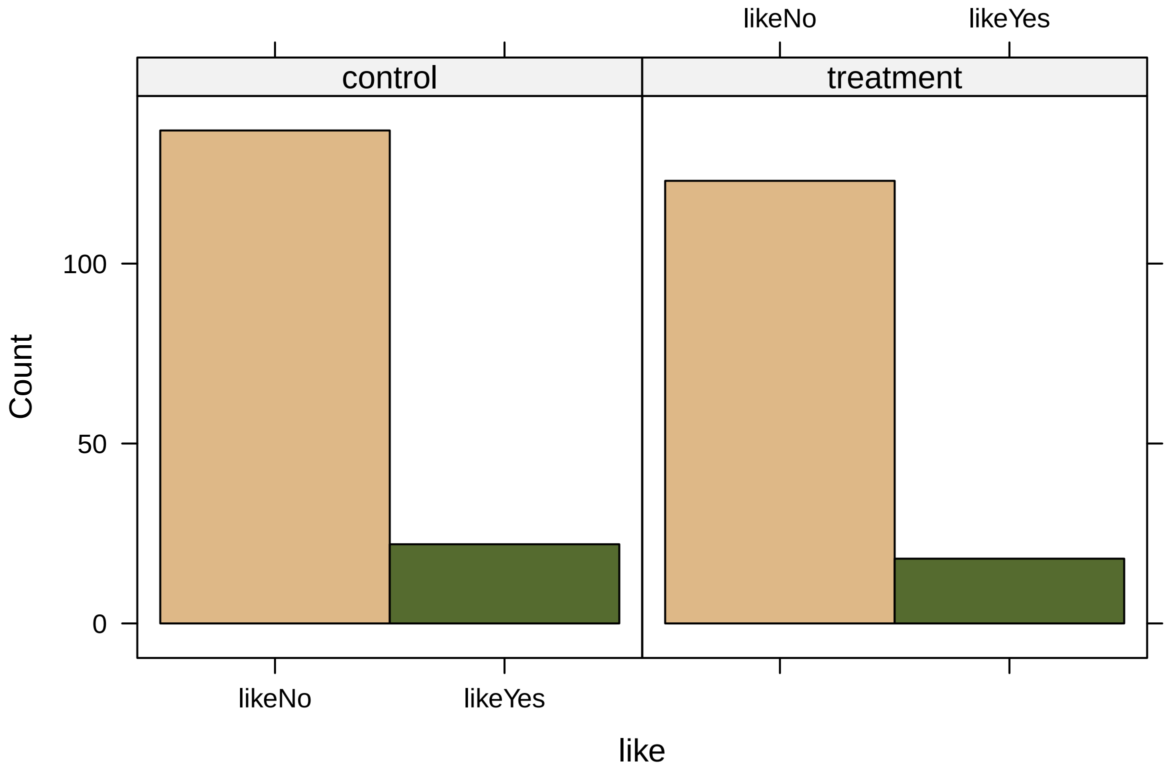


The result shows that the two versions of ads generate similar likes.

You will notice there is no response variable before the tilde (~) in this formula, only the explanatory variable (*condition*) after it. *histogram()* by default, assumes that we want to plot the *proportion* of people at each level of *like*. We condition the plot on *condition*, telling *histogram* to produce a separate histogram for each segment.

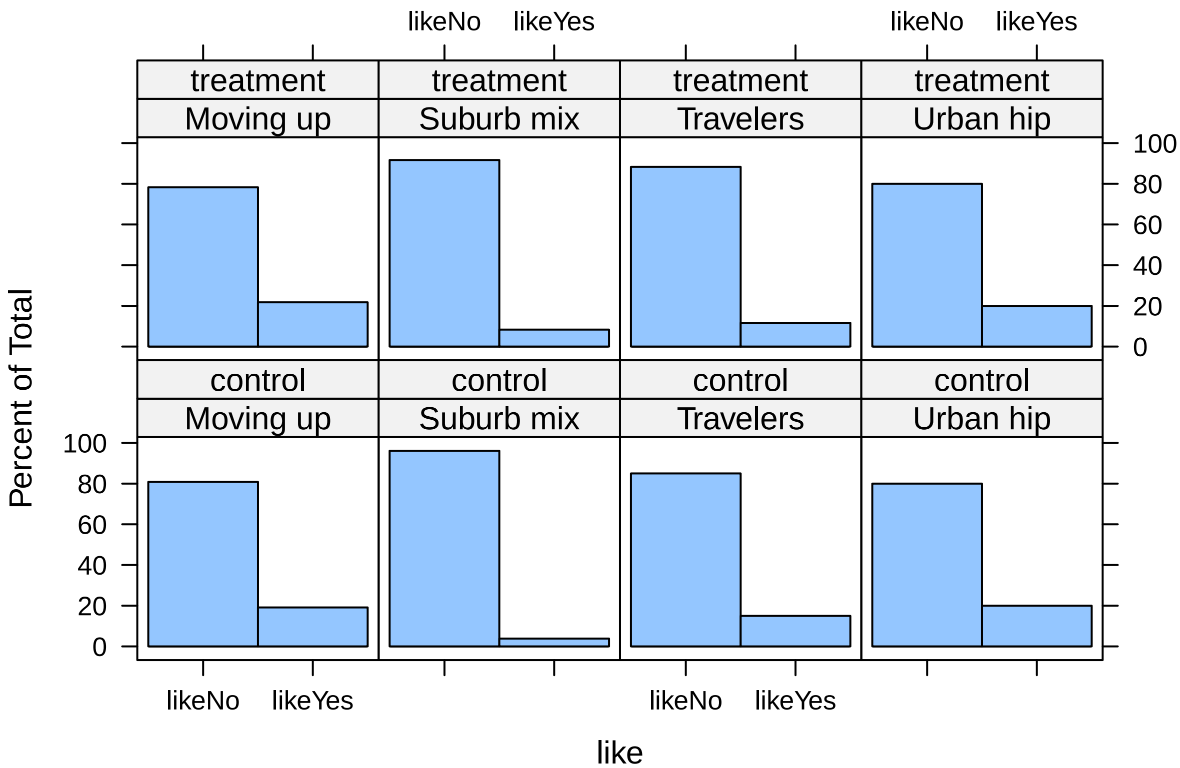
The default in *histogram()* is to plot *proportions* within each group so that the values are relative to the group size. If we want actual *counts* instead, we can include the argument *type = "count"*.

```
histogram(~ like | condition, data = ad.df, type = "count",
                    col = c("burlywood", "darkolivegreen") # add colours
)
```



We can add conditions on multiple factors by using "+". For instance, what is the proportion of subscribers within each segment by home ownership?

```
histogram(~ like | segment + condition, data = ad.df)
```

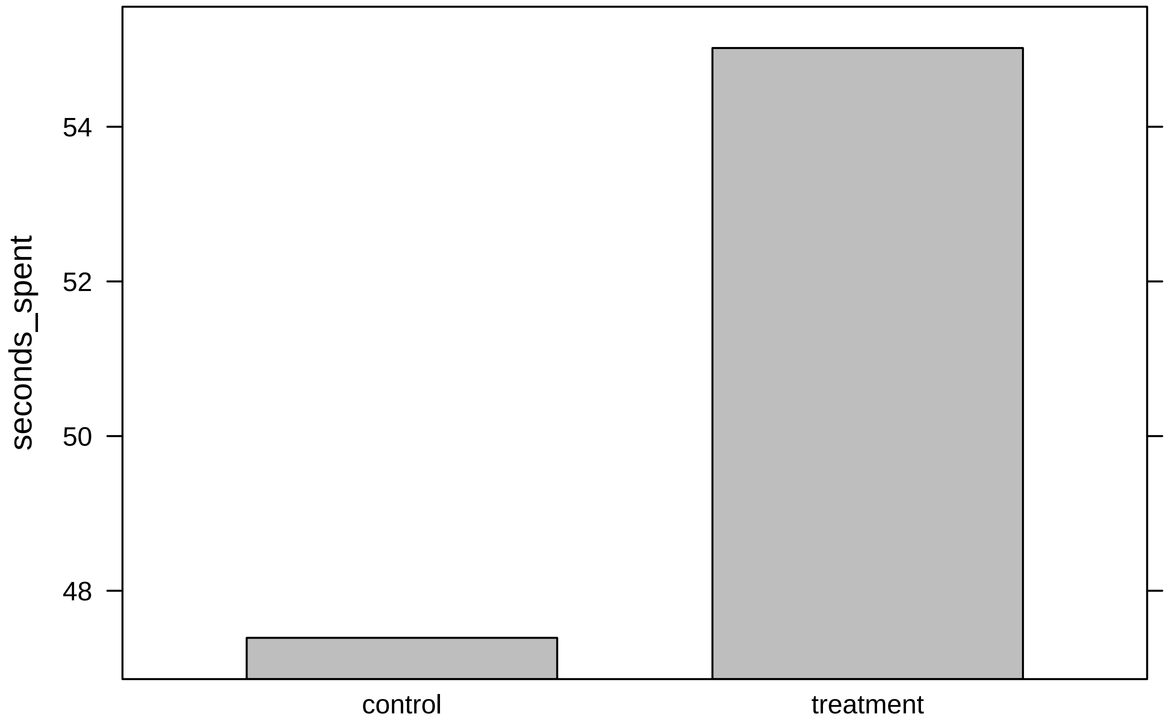


The result tells us the difference in like according to ads condition within segment is small. This implies that the two versions of the ads generate little difference in liking.

2.2 Visualizing continuous data

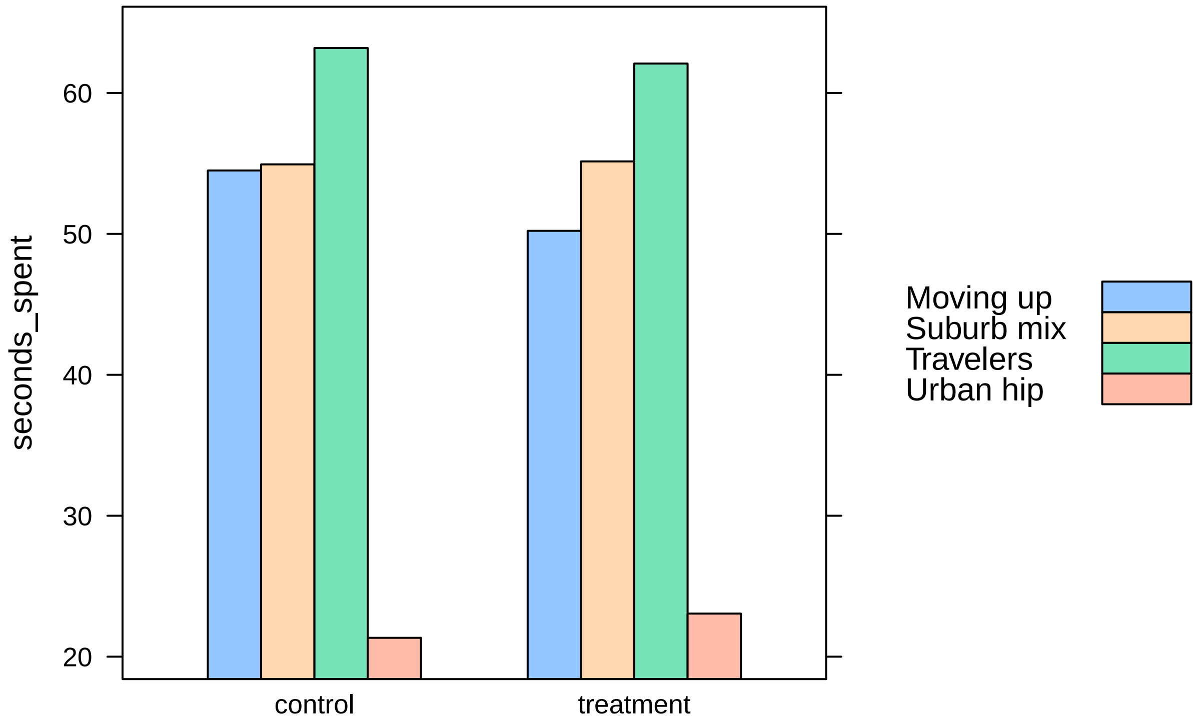
In the previous section, we saw how to plot counts and proportions. What about continuous data? How would we plot *total second spent* by *condition* in our data? A simple way is to use *aggregate()* to find the mean total time spent and then use *barchart()* from the *lattice* package to plot the computed values:

```
ad.mean <- aggregate(seconds_spent ~ condition, data = ad.df, mean)
library(lattice)
barchart(seconds_spent ~ condition, data = ad.mean, col = "grey")
```



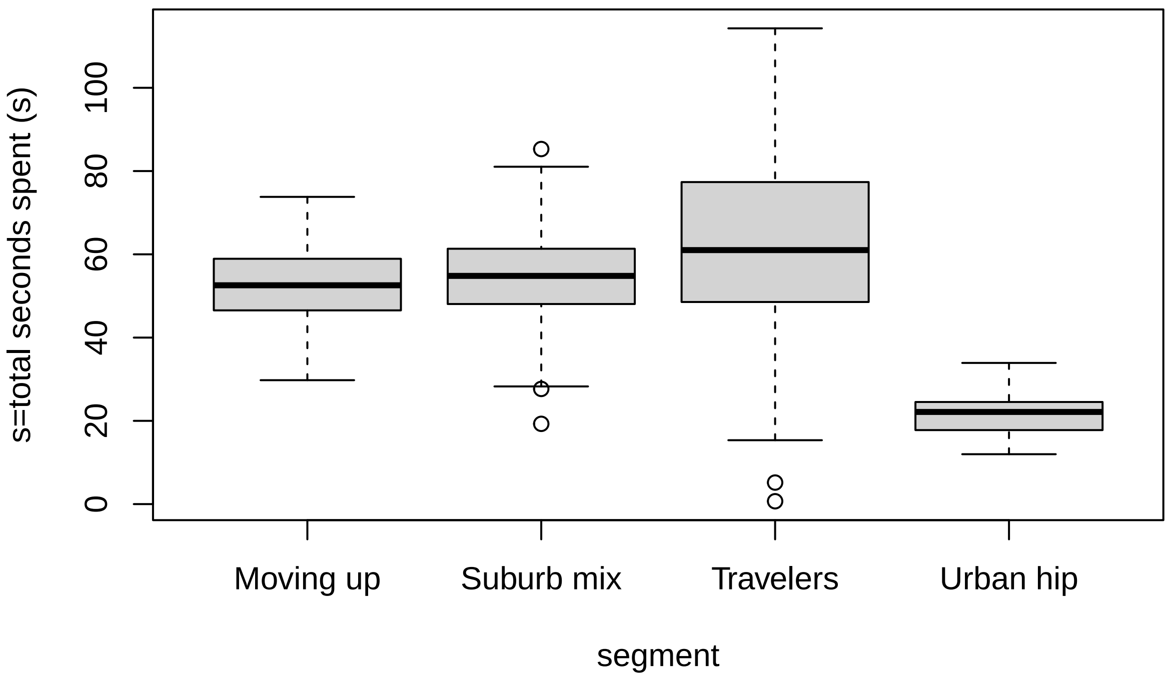
How do we split this out further by segment? First, we have to aggregate the data to include both factors in the formula. Then we tell *barchart()* to use *segment* as a grouping variable by adding the argument *groups=factor*.

```
ad.seconds.agg <- aggregate (seconds_spent ~ condition + segment, data = ad.df, mean)
barchart(seconds_spent ~ condition, data = ad.seconds.agg ,groups = segment, auto.key=TRUE)
```



A more informative plot for comparing values of continuous data, like *seconds_spent* for different groups, is the *boxplot*. A boxplot is better than a bar chart because it shows more about the *distributions* of values.

```
boxplot(seconds_spent ~ segment, data= ad.df,
        ylab = "s=total seconds spent (s)")
```

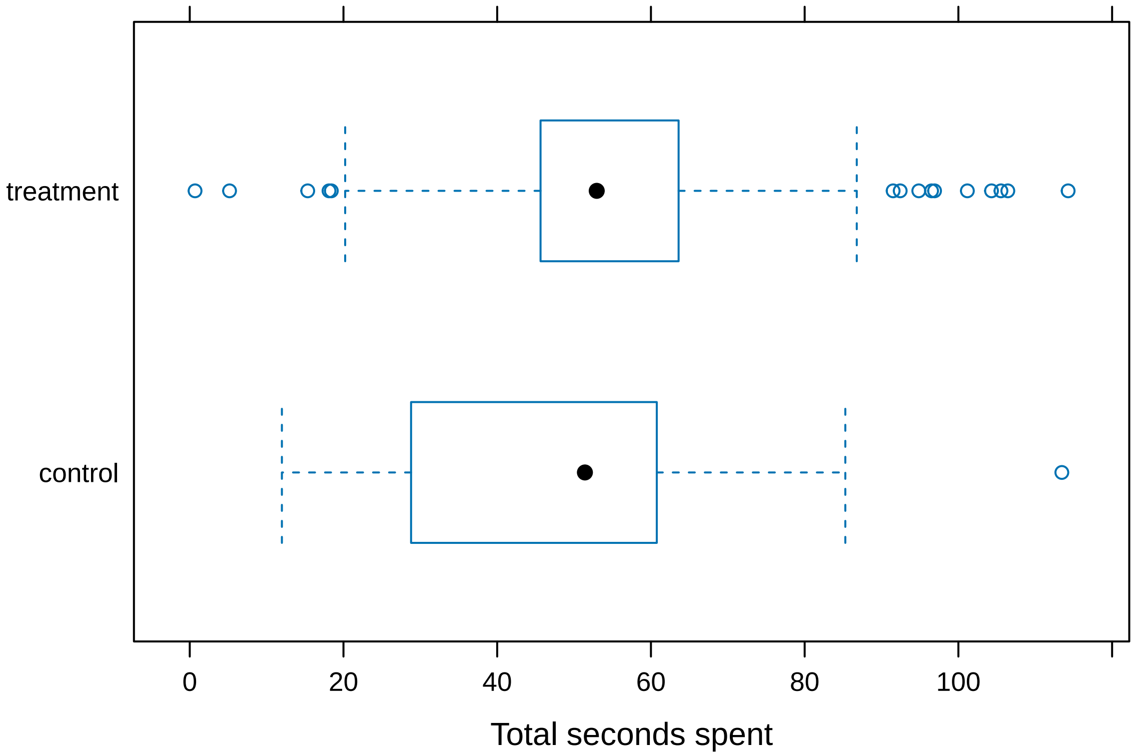


The result shows that the total seconds spent for "Travelers" is higher and also has a greater range, with a few "Travelers" spending very few seconds. The range of seconds spent for "Urban hip" is much less and tighter.

2.2.1 bwplot() *

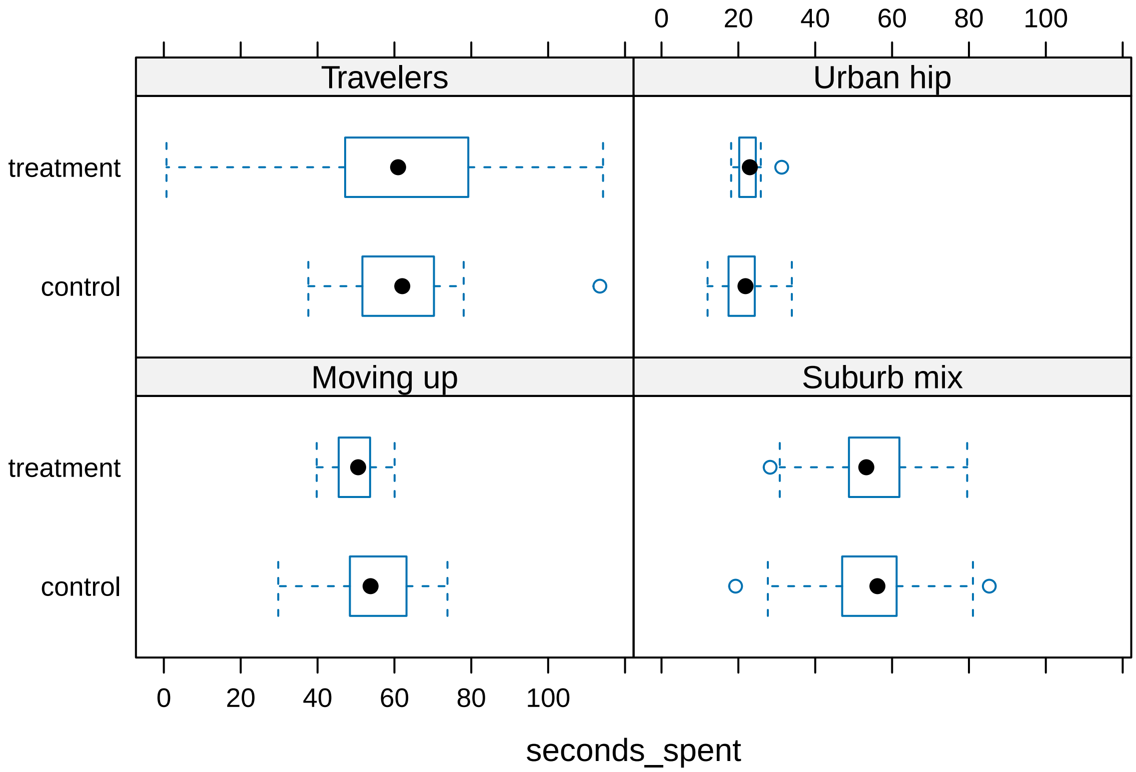
An even better option for box-and-whisker splits is the *bwplot()* command from the *lattice* package, which provides better-looking charts and allows multi-factor conditioning. One point of caution is that *bwplot()* uses the model formula in a direction opposite than you may expect; you write *condition* ~ *seconds_spent*. We plot a horizontal box-and-whiskers for seconds_spent by segment as follows:

```
bwplot(condition ~ seconds_spent, data = ad.df, horizontal = TRUE, xlab = "Total seconds spent")
```



We can break out treatment as a conditioning variable:

```
bwplot(condition ~ seconds_spent | segment, data = ad.df, horizontal = TRUE, xlab = "seconds_spent")
```



The conditioned plot for total seconds spent by segment and ads conditions shows that the Travelers segment has a much wider distribution of total seconds spent among those who are in the treatment group than those who are in the control group.

3 Statistical tests

In addition to summarize the differences between groups using group averages and cross tabs as described above, a good analyst is able to use *statistical tests* to determine whether differences are real or might instead be due to the minor variation ("noise") in the data. We should focus on statistical tests that help to identify the real difference.

3.1 Testing group frequencies: *chisq.test()*

chi-square test is used with frequency counts such as those produced by *table*. A chi-square test determines whether the frequencies in cells are significantly different from what one would expect on the basis of their total counts. In R, we use the *chisq.test()* command. In general, *chisq.test()* operates on a *table*.

Is liking behavior independent from conditions? That is, in our data, are respondents just as likely to like, regardless of which version of the ad they see? We construct a two-way table and test it:

```
table(ad.df$like, ad.df$condition)
```

```
##
##           control treatment
##   likeNo      137       123
##   likeYes      22        18
```

```r
chisq.test(table(ad.df$like, ad.df$condition))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(ad.df$like, ad.df$condition)
## X-squared = 0.010422, df = 1, p-value = 0.9187
```

The null hypothesis, in this case, is that the factors are unrelated. i.e., the counts in the cells are as one might expect from the marginal proportions. Based on the high *p*-value, we cannot reject the null hypothesis and conclude that the factors are unrelated and that like is independent of the condition in our data. There is no *relationship* between like and condition.

3.2 Testing group means: *t.test()*

A t-test compares the mean of one sample against the mean of another sample (or against a specific value such as 0). The important point is that it compares the *mean* for exactly *two* sets of data. For instance, in the data, we might ask whether the total seconds spent are different between the two conditions of ads.

We test difference in *seconds_spent* between two groups (treatment vs. condition), using *t.test(formula, data)*:

```r
t.test(seconds_spent ~ condition, data = ad.df)
```

```
##
##  Welch Two Sample t-test
##
## data:  seconds_spent by condition
## t = -3.3297, df = 286.45, p-value = 0.0009832
## alternative hypothesis: true difference in means between group control and group treatment is not equ
## 95 percent confidence interval:
##  -12.135137  -3.118392
## sample estimates:
##   mean in group control mean in group treatment
##                47.39132                55.01809
```

There are several important pieces of information in the output of *t.test()*

- *t statistic* is -3.12, with a *p*-value of 0.0010. This means that the null hypothesis of *no difference* in seconds_spent by conditions is rejected. It suggests that people who see the ad version in the treatment condition spend a longer time.
- 95% interval confidence interval for the difference is -12.14 to -3.12. We can have 95% confidence that the group difference is between those values.
- the sample means for our data: mean total second spent is 47.39132 for the control condition, and 55.01809 for the treatment condition.

What about the difference within the Travelers segment? We can use the filter *data = subset(data, condition)* to select just Travelers and repeat the test:

```r
t.test(seconds_spent ~ condition, data = subset(ad.df, segment == "Travelers"))
```

```
##
##  Welch Two Sample t-test
##
## data:  seconds_spent by condition
## t = 0.22758, df = 52.758, p-value = 0.8209
## alternative hypothesis: true difference in means between group control and group treatment is not equ
## 95 percent confidence interval:
```

```
##  -8.624575 10.831909
## sample estimates:
##   mean in group control mean in group treatment
##                63.18900                62.08533
```

The confidence interval of –8.62 to 10.83 includes 0, and the p-value is 0.82. Thus we conclude that there is not a significant difference in mean seconds spent between the two conditions among those Travelers in our data.

3.3 Testing multiple-group means: ANOVA

An ANOVA compares the means of multiple groups. The null hypothesis is that there is no difference among multiple means.

An ANOVA can handle a single factor (known as *one-way* ANOVA), two factors (*two-way*), and higher orders, including interactions among factors.

The basic R commands for ANOVA are *aov(formula, data)* to set up the model, followed by *anova(model)* to display a standard ANOVA summary.

For instance, we want to answer the question: are seconds spent related to conditions, segment membership, or both? We can model *seconds_spent* as a response to *condition*:

```r
ad.aov.con <- aov(seconds_spent ~ condition, data = ad.df)
anova(ad.aov.con)
```

There is a significant variance in *seconds_spent* between two conditions (Same conclusion as we did from t-test)

To test if seconds_spent varies by *both* condition and segment, we can add both factors into the ANOVA model to test this:

```r
anova(aov(seconds_spent ~ segment + condition, data = ad.df)) # combine two commands
```

The results indicate that when we try to explain the total seconds spent differences by both *segment* and *condition*, the segment is a significant predictor, but the condition is *not* a significant predictor. Yet the previous results said that it *was* significant. What is the difference? It means segment and condition are not independent, and the effect is captured sufficiently by segment membership alone. *condition* accounts for a little more than what can be explained by *segment*.

3.3.1 Visualize ANOVA result

A good way to visualize the results of an ANOVA is to plot confidence intervals for the group means. We use the *multcomp* (multiple comparison) package, and its *glht(model)* (general linear hypothesis) command.

The default *aov()* model has an intercept term (corresponding to one segment) and all other segments are relative to that. This may be difficult for decision-makers or clients to understand, so we find it preferable to remove the intercept by adding "0" to the model formula:

```r
library(multcomp)
```

```
## Warning: package 'multcomp' was built under R version 4.3.3

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 4.3.3

## Loading required package: survival

## Loading required package: TH.data

## Warning: package 'TH.data' was built under R version 4.3.3
```

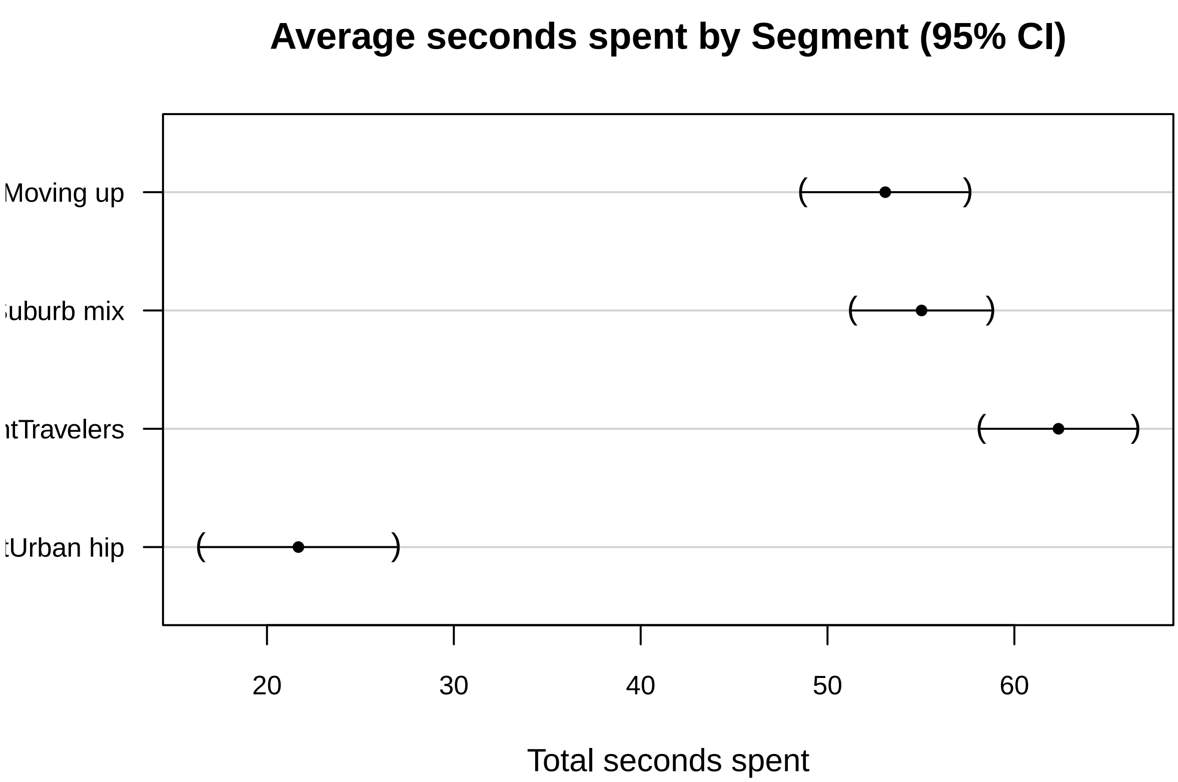
```
## Loading required package: MASS
##
## Attaching package: 'TH.data'
## The following object is masked from 'package:MASS':
##
##     geyser
```

```
ad.aov <- aov (seconds_spent ~ 0 + segment, data = ad.df)
glht(ad.aov)
```

```
##
##   General Linear Hypotheses
##
## Linear Hypotheses:
##                           Estimate
## segmentMoving up == 0      53.09
## segmentSuburb mix == 0     55.03
## segmentTravelers == 0      62.36
## segmentUrban hip == 0      21.68
```

With the intercept removed, *glht()* gives us the mean values for each segment. We *plot()* that, using the *par(mar =...)* command to add some extra margins for long-axis labels:

```
plot(glht(ad.aov),
     xlab = "Total seconds spent", main = "Average seconds spent by Segment (95% CI)", cex.axis = 0.8)
```



```r
# cex.axis = 0.8 makes the axis labels smaller to 80%.
```

The dots show the mean for each segment, and the bars reflect the confidence interval. We can see confidence intervals for the mean seconds spent in each segment. It is clear that the average seconds spent by Urban hip segment members is substantially lower than the other three groups.

3.4 Testing group means: lm()

You can also compare groups by using *lm()* regression when the response variable is continuous, such as total second spent.

The grouping variables (factors) have to be recoded as dummy variables (i.e. variables with 0 and 1 as values). For instance, factor "condition" (with "control" and "treatment") should be recoded as "dummy_condition: 0 = control, 1 = treatment.

Factor with more than two levels (n) should be recoded with multiple dummy variables (n-1). For instance, Segment has four levels and could be recoded with three dummy variables: dummy_s, dummy_u, dummy_u, and: Suburb mix: dummy_s = 1, dummy_u =0, dummy_t =0 Urban hip: dummy_s = 0, dummy_u = 1, dummy_t=0 Travelers: dummy_s=0, dummy_u=0, dummy_t=1 Moving up: dummy_s=0, dummy_u=0, dummy_t=0 (reference level)

```r
library(dplyr)
```

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

## The following object is masked from 'package:MASS':
##
##     select

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

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

```r
ad.df.reg <- ad.df %>%
  mutate(dummy_condition = ifelse(condition == "treatment",1,0),
         dummy_s = ifelse(segment == "Suburb mix",1,0),
         dummy_u = ifelse(segment == "Urban hip",1,0),
         dummy_t = ifelse(segment == "Travelers",1,0))
```

Dummy variables should then enter the model as predictors.

```r
regression <- lm(seconds_spent ~ dummy_condition, data = ad.df.reg)
summary(regression)
```

```
##
## Call:
## lm(formula = seconds_spent ~ dummy_condition, data = ad.df.reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.33 -11.02   0.97  12.10  66.07
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)       47.391      1.563  30.327  < 2e-16 ***
## dummy_condition    7.627      2.279   3.346 0.000925 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.7 on 298 degrees of freedom
## Multiple R-squared:  0.03621,    Adjusted R-squared:  0.03297
## F-statistic:  11.2 on 1 and 298 DF,  p-value: 0.0009251
```

```
regression <- lm(seconds_spent ~ dummy_s + dummy_u + dummy_t, data = ad.df.reg)
summary(regression)
```

```
##
## Call:
## lm(formula = seconds_spent ~ dummy_s + dummy_u + dummy_t, data = ad.df.reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.671  -7.059  -0.442   6.281  51.919
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   53.091      1.769  30.017  < 2e-16 ***
## dummy_s        1.943      2.306   0.842 0.400214
## dummy_u      -31.409      2.740 -11.463  < 2e-16 ***
## dummy_t        9.270      2.422   3.828 0.000158 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.8 on 296 degrees of freedom
## Multiple R-squared:  0.4601, Adjusted R-squared:  0.4546
## F-statistic: 84.08 on 3 and 296 DF,  p-value: < 2.2e-16
```

3.5 Difference-in-Difference

```
panel.df.raw <- read.csv("Data_Panel.csv", stringsAsFactors = TRUE)
str(panel.df.raw)
```

```
## 'data.frame':    70 obs. of  3 variables:
##  $ market: Factor w/ 7 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year  : int  2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 ...
##  $ profit: num  13428 -18997 -112 26458 30083 ...
```

Create a dummy variable to indicate the time when the treatment started. Let's assume that treatment began in 2014. In this case, years before 2014 will have a value of 0, and after 2014 will have 1.

```
library(dplyr)
panel.df<-panel.df.raw %>%
  mutate(time = ifelse(panel.df.raw$year >= 2014, 1, 0))
```

We then create a dummy variable to identify the group exposed to the treatment. In this example, markets 5, 6, and 7 were the treatment group (=1). Markets 1-4 were not influenced (=0).

```
panel.df<-panel.df %>%
  mutate(treatment = ifelse(panel.df$market == "E" |
                              panel.df$market== "F" |
```

```
                                    panel.df$market == "G", 1, 0))
```

Create an interaction term between time and treated. We will call this interaction 'did'.

```
panel.df<-panel.df %>%
  mutate(did = time * treatment)
str(panel.df)
```

```
## 'data.frame':    70 obs. of  6 variables:
##  $ market   : Factor w/ 7 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year     : int  2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 ...
##  $ profit   : num  13428 -18997 -112 26458 30083 ...
##  $ time     : num  0 0 0 0 1 1 1 1 1 1 ...
##  $ treatment: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ did      : num  0 0 0 0 0 0 0 0 0 0 ...
```

3.5.1 Estimating the DID estimator

```
didreg <- lm(profit ~ treatment + time + did, data = panel.df)
summary(didreg)
```

```
##
## Call:
## lm(formula = profit ~ treatment + time + did, data = panel.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -97675 -16228   1167  13928  68071
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3581       7382   0.485   0.6292
## treatment      17760      11276   1.575   0.1200
## time           22895       9530   2.402   0.0191 *
## did           -25195      14557  -1.731   0.0882 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29530 on 66 degrees of freedom
## Multiple R-squared:  0.08273,    Adjusted R-squared:  0.04104
## F-statistic: 1.984 on 3 and 66 DF,  p-value: 0.1249
```

The coefficient for 'did' is the differences-in-differences estimator. The effect is marginally significant at 10%, with the treatment having a negative effect.

4 Takeaways

When describing and visualizing data for groups:

- *aggregate()* is more powerful; it understands formula models and produces a reusable, indexable object with its result
- Frequency of occurrence can be found with *table()*.
- Charts of proportions and occurrence by a factor are well suited to the *lattice* package *histogram()* command

- Plot for continuous data by factor may use *barchart()*, or even better, box-and-whiskers plots with *boxplot()*. The *lattice* package extends such plots to multiple factors using formula specification and the *bwplot()* command.

When performing statistical tests on differences by group:

- *chisq.test()* find confidence intervals, and perform hypothesis tests on table data, respectively.
- A *t.test()* is a common way to test for differences between the means of two groups (or between one group and a fixed value)
- ANOVA is a more general way to test for differences in mean among several groups that are identified by one or more factors. The basic model is fit with *aov()* and common summary statistics are reported with *anova()*
- The *anova()* command is also useful to compare two or more ANOVA or other linear models, provided they are nested models
- Plotting a *glht()* object from the *multicomp* package is a good way to visualize confidence intervals for ANOVA models.
- linear regression *lm()* is a more general way to examine the difference among groups. Dummy coding is important for grouping variables
- Difference-in-Difference is tested by examining the coefficient of the interaction term between treatment and time.