6

Comparing Groups: Statistical Tests

In Chap. 5 we saw how to break out data by groups and inspect it with tables and charts. In this chapter we continue our discussion and address the question, "It looks different, but is it really different?" This involves our first inferential statistical procedures: chi-square, *t*-tests, and analysis of variance (ANOVA). In the final section, we introduce a Bayesian approach to compare groups.

6.1 Data for Comparing Groups

In this chapter, we continue with the data from Chap. 5. If you saved it at that time, you could load it again with a command such as:

```
> load("~/segdf-Rintro-Ch5.RData")      # modify directory as needed
> summary(seg.df)
      age           gender         income             kids         ownHome
 Min.   :19.26   Female:157   Min.   : -5183   Min.   :0.00   ownNo :159
 1st Qu.:33.01   Male  :143   1st Qu.: 39656   1st Qu.:0.00   ownYes:141
...
```

Alternatively, you could create the data following the procedure in Sect. 5.1. Or download it from this book's website:

```
> seg.df <- read.csv("http://goo.gl/qw303p")
> summary(seg.df)
      age           gender         income             kids         ownHome
 Min.   :19.26   Female:157   Min.   : -5183   Min.   :0.00   ownNo :159
 1st Qu.:33.01   Male  :143   1st Qu.: 39656   1st Qu.:0.00   ownYes:141
...
```


C. Chapman, E.M. Feit, *R for Marketing Research and Analytics*, Use R!,
DOI 10.1007/978-3-319-14436-8_6

6.2 Testing Group Frequencies: `chisq.test()`

Much of the work we do in marketing analytics and marketing research involves summarizing the differences between groups using group averages and cross tabs as we described in Sect. 5.2. However, a good analyst is able to use *statistical tests* to determine whether differences are real or might instead be due to minor variation ("noise") in the data. In the rest of the book, we largely focus on statistical tests that help to identify real differences.

One of the simplest statistical tests is the *chi-square* test, which 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 our segment data, we might ask whether there are equal numbers of respondents in each segment, given a marginal count of N=300 observations. In R, we use the `chisq.test()` command. One thing to remember is that in general `chisq.test()` operates on a *table* (such as produced by `table()`). To see how this works, let's look at the process using simple data before we tackle the question for our segments. Experimenting with simple data is always a good idea when trying a new command.

For the first example, we create a table where the data comprises 95 observations of the numbers 1–4 and where the counts of each are almost, but not quite identical. We then test this with `chisq.test()`:

```
> tmp.tab <- table(rep(c(1:4), times=c(25,25,25,20)))
> tmp.tab
 1  2  3  4
25 25 25 20

> chisq.test(tmp.tab)

  Chi-squared test for given probabilities

data:  tmp.tab
X-squared = 0.7895, df = 3, p-value = 0.852
```

In this code, we generate 95 observations of 1:4, compile those into a table, and then test that table for chi-square independence. The test evaluates the likelihood of seeing such a result under the *null hypothesis* that the data were randomly sampled from a large population where the values 1:4 are *equally distributed*, given a marginal count of $N = 95$ observations. The *p-value* of 0.852 tells us that there is an estimated 85 % chance of seeing a data set with differences similar to or greater than those in our table, if the null hypothesis is true. We conclude that under the assumptions of the chi-square test, our table does not suggest real differences in frequency between the four cells. Put another way, this data shows no evidence that the groups in the population are of unequal size, under the assumption of random sampling.

Compare that to the following, which differs from the code above by a single character—we change the number of observations of "4" from 20 to 10:

```
> tmp.tab <- table(rep(c(1:4), times=c(25,25,25,10)))
> tmp.tab
 1  2  3  4
25 25 25 10

> chisq.test(tmp.tab)

  Chi-squared test for given probabilities

data:  tmp.tab
X-squared = 7.9412, df = 3, p-value = 0.04724
```

In this case, we could conclude from the *p*-value of 0.047 that we can reject the null hypothesis of no difference between the cells with "95 % confidence." In other words, the data in this sample suggests that the distribution of the values 1:4 is likely to be unequal in the larger population, assuming the data are a random sample. In general, a *p*-value less than 0.10 or 0.05 suggests that there is a difference between groups.

As an aside, there are disagreements among statisticians about the meaning of null hypotheses and the value of traditional significance testing. We do not advocate classical significance testing in particular, but report the methods here because they are widely used in marketing to gauge the strength of evidence in a data set. We believe the classical methods are imperfect but nevertheless useful and important to know. For review and discussion of the controversies and alternatives, see [28, 80, 95]. In Sect. 6.6 we introduce Bayesian methods that do not use this kind of null hypothesis.

In the results above, if we had a smaller sample we would not get the same result for the significance test even if the relative proportion of customers in each group was the same. Significance tests are sensitive to both the observed difference and the sample size. To see this, we can create data with the same proportions but one fifth as many observations by dividing tmp.tab by 5.

```
> tmp.tab <- tmp.tab/5
> tmp.tab

1 2 3 4
5 5 5 2
> chisq.test(tmp.tab)

  Chi-squared test for given probabilities

data:  tmp.tab
X-squared = 1.5882, df = 3, p-value = 0.6621

Warning message:
In chisq.test(tmp.tab) : Chi-squared approximation may be incorrect
```

This shows a non-significant result—no evidence of a real difference in group sizes—even though the proportion of people in the "4" group is the same as in

the larger sample above where the result was significant. This highlights one of the cautions about statistical significance testing: it is dependent on sample size as well as on the real effect.

Returning to our simulated segment data, which has a $N = 300$ observations, we ask whether the segment sizes are significantly different from one another (assuming that our 300 customers are a random sample of a larger population). We use the same procedure as above, combining `chisq.test()` and `table()` into one command:

```
> chisq.test(table(seg.df$Segment))

  Chi-squared test for given probabilities

data:  table(seg.df$Segment)
X-squared = 17.3333, df = 3, p-value = 0.0006035
```

The answer to our question is "yes, there are differences in segment size." That is, with $p = 0.0006$, our sample does not support the hypothesis that there is an identical number of customers in each segment.

Is subscription status independent from home ownership, as we hypothesized when we plotted the data in Sect. 5.2? That is, in our simulated data, are respondents just as likely to subscribe or not, without regard to home ownership status (and conversely, are they just as likely to own a home or not, independent of subscription status)? We construct a two-way table and test it:

```
> table(seg.df$subscribe, seg.df$ownHome)

         ownNo ownYes
  subNo    137    123
  subYes    22     18

> chisq.test(table(seg.df$subscribe, seg.df$ownHome))

  Pearson's Chi-squared test with Yates' continuity correction

data:  table(seg.df$subscribe, seg.df$ownHome)
X-squared = 0.0104, df = 1, p-value = 0.9187
```

The null hypothesis in this case is that the factors are unrelated, i.e., that 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 home ownership is independent of subscription status in our data. Although people in general have a low subscription rate—and thus there are many more non-subscribers than subscribers in both groups—there is no *relationship* between subscription rate and home ownership.

We should note two options for `chisq.test()`. First, for 2×2 tables, `chisq.test()` defaults to using *Yates' correction*, which adjusts the chi-square statistic in light of the fact that the assumption of continuous data is imperfect when data comes from a lumpy binomial distribution. If you want the results to match

traditional values such as calculation by hand or spreadsheet, turn that off with `correct=FALSE`:

```
> chisq.test(table(seg.df$subscribe, seg.df$ownHome), correct=FALSE)

  Pearson's Chi-squared test

data:  table(seg.df$subscribe, seg.df$ownHome)
X-squared = 0.0741, df = 1, p-value = 0.7854
```

Second, `chisq.test()` can calculate confidence intervals using a simulation method, where it compares the observed table to thousands of simulated tables with the same marginal counts. The *p*-value indicates the proportion of those simulations with differences between the cell counts and marginal proportions at least as large as the ones in the observed table. We do that for 10,000 simulations using the `sim` and `B` arguments as follows:

```
> chisq.test(table(seg.df$subscribe, seg.df$ownHome), sim=TRUE, B=10000)

  Pearson's Chi-squared test with simulated p-value (based on 10000
  replicates)

data:  table(seg.df$subscribe, seg.df$ownHome)
X-squared = 0.0741, df = NA, p-value = 0.8596
```

The test statistics and *p*-values change slightly across these commands, but the overall conclusion is the same, namely that the factors are independent.

6.3 Testing Observed Proportions: `binom.test()`

When we are dealing with observations that have only two values, we can consider them to be a binomial (two-valued) variable. We illustrate this by taking a brief break from marketing data. On the day of Superbowl XLVIII in 2014, played in the New York City area, Chris took a walk in Manhattan and observed 12 groups of Seattle fans and 8 groups of Denver fans.

Suppose we assume the observations are a random sample of a binomial value (either Seattle or Denver fandom). Is the observed value of 60 % Seattle fans significantly different from equal representation (which would be 50 % each)? We use `binom.test(successes, trials, probability)` to test the likelihood of randomly observing 12 cases out of 20 in one direction, if the true likelihood is 50 %:

```
> binom.test(12, 20, p=0.5)

  Exact binomial test

data:  12 and 20
number of successes = 12, number of trials = 20, p-value = 0.5034
alternative hypothesis: true probability of success is not equal to 0.5
```

```
95 percent confidence interval:
 0.3605426 0.8088099
sample estimates:
probability of success
                   0.6
```

Based on our data, the 95 % confidence interval is 36–81 %, which includes the null hypothesis value of 50 %. Thus, we conclude that observing 60 % Seattle fans in a sample of 20 does not conclusively demonstrate that there are more Seattle fans in the larger group of fans roaming New York. We could also interpret the p-value ($p = 0.5034$) as being non-significant, i.e., as failing to support the idea that the results are different from the null hypothesis.

6.3.1 About Confidence Intervals

We have mentioned *confidence intervals* several times, and should take a moment to discuss them because they are widely misunderstood. Our definition of a 95 % confidence interval is this: it is the range of possible estimates that we would expect to see 95 % of the time if we repeatedly estimate a statistic using *random samples* of the *same sample size* under the assumption that the *true value* in an *infinite* or very large population is the same as our current estimate. In other words, it is the best guess of the range of possible answers we would expect with repeated random samples. When the confidence interval excludes the null hypothesis (such as a probability of 0.5 for equal chances, or a mean difference of 0 for no difference between groups), then the result is said to be *statistically significant*.

There are many misunderstandings of confidence intervals and statistical significance. Confidence intervals (CIs) do *not* express "how confident we are in the answer" because they do not reflect the degree of confidence in the assumptions. For example, true random sampling is rare, so the presumption of random sampling is usually not completely justified; but that additional uncertainty is not reflected in the CI. CIs are often misunderstood to imply that "the true value lies in the CI range," when in fact it is the other way around; *if* the true value is what we obtained, then we would expect additional estimates to fall within this CI 95 % of the time under further rounds of random sampling. The CI is about *estimates*, not about the true value. Additionally, statistical significance does not imply practical importance or the meaningfulness of a result; a tiny difference can be statistically significant with a large sample even when it is not actionable or interpretable as a business matter.

In practice, we suggest that before interpreting a result, make sure it is statistically significant for some level of confidence interval (95 %, or possibly 90 % or 99 % depending on how sensitive the matter is). If it is not significant, then your evidence for the result is weak, and you should not interpret it. In that case, either say that, ignore the result, or collect more data. If the result *is* significant, then proceed with your interpretation and reporting (taking care with how you describe "confidence").

Interpret results in light of their importance, not their statistical significance (once it has been established). We recommend to report—and when appropriate, to chart—confidence intervals whenever feasible rather than reporting single point estimates. By reporting CIs, one presents a more complete and accurate description to stakeholders.

Note that this discussion applies to the interpretation of significance in classical statistics (which covers most of this book, and is what practitioners mostly use). We briefly review the Bayesian alternative to confidence intervals (known as *credible intervals*) in Sect. 6.6.2 below. In general, the cautions expressed above do not directly apply to Bayesian models (there are different considerations), yet the practical recommendations about interpretation and reporting are consistent.

There is a general function in R to determine the confidence intervals for a statistical model (when appropriate): `confint()`, which we use in the next section.

6.3.2 More About `binom.test()` and Binomial Distributions

Now that we understand confidence intervals, let's look at `binom.test()` again. What if we had observed 120 out of 200 to be Seattle fans, the same proportion as before but in a larger sample?

```
> binom.test(120, 200, p=0.5)
...
number of successes = 120, number of trials = 200, p-value = 0.005685
...
95 percent confidence interval:
 0.5285357 0.6684537
```

With 120/200 cases, the confidence interval no longer includes 50 %. If we had observed this, it would be evidence for a preponderance of Seattle fans. Correspondingly, the p-value is less than 0.05, indicating a statistically significant difference.

With R, we can ask much more about the distribution. For example, what are the odds that we would observe 8–12 Seattle fans out of 20, if the true rate is 50 %? We use the density estimate for a binomial distribution across the range of interest and sum the point probabilities:

```
> sum(dbinom(8:12, 20, 0.5))
[1] 0.736824
```

If we observe 20 fans, and the true split is 50 %, there is a 73.7 % chance that we would observe between 8 and 12 fans (and thus a $1 - p$ or 27.3 % chance of observing fewer than 8 or more than 12).

An "exact" binomial test (the classical method) may be overly conservative in its estimation of confidence intervals [2]. One alternative method is to use `binom.confint(, method="agresti-coull")`, available in the `binom` package [35] (you may need to install that package):

```
> library(binom)
> binom.confint(12, 20, method="ac")  # same as "agresti-coull"
         method  x  n mean     lower     upper
1 agresti-coull 12 20  0.6 0.3860304 0.7817446
```

With the Agresti–Coull method, the confidence interval is slightly smaller but still includes 50 %. The `binom` package also computes several other variants on binomial tests, including a Bayesian version.

Finally, Chris also observed that among the 20 groups, 0 had a mixture of Seattle and Denver fans (as inferred from their team clothing). Based on that observation, what should we conclude is the *most likely* proportion of groups that comprise mixed fans? We use the Agresti–Coull method because exact tests have no confidence interval for 0 % or 100 % observations:

```
> binom.confint(0, 20, method="ac")
         method x  n mean      lower     upper
1 agresti-coull 0 20    0 -0.0286844 0.1898096
```

The negative lower bound may be ignored as an artifact, and we conclude that although Chris observed 0 cases, the occurrence of mixed fandom groups is likely to be somewhere between 0 and 19 %.

6.4 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 segment data we might ask whether household income is different among those who own a home and those who do not.

Before applying any statistical test or model, it is important to examine the data and check for skew, discontinuities, and outliers. Many statistical tests assume that the data follows a normal distribution or some other smooth continuous distribution; skewness or outliers violate those assumptions and might lead to an inaccurate test.

One way to check for non-normal distributions is to plot the data with a boxplot or histogram. We have already plotted `income` above (Figs. 5.7, 5.8, and 5.9) and thus skip that step. Additionally, we can check histograms for income overall as well as by home ownership:

```
> hist(seg.df$income)  # not shown
> with(seg.df, hist(income[ownHome=="ownYes"]))  # not shown
> with(seg.df, hist(income[ownHome=="ownNo"]))   # not shown
```

We omit those figures for brevity. Overall, in these histograms and in the boxplots above, `income` is approximately normally distributed (as it should be, given the data generation procedure, Sect. 5.1).

Now we are ready to test whether home ownership overall is related to differences in income, across *all* segments, using `t.test(formula, data)`. We write the formula using `income` as the response variable to be modeled on the basis of `ownHome` as the explanatory variable:

```
> t.test(income ~ ownHome, data=seg.df)

  Welch Two Sample t-test

data:  income by ownHome
t = -3.2731, df = 285.252, p-value = 0.001195
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -12080.155  -3007.193
sample estimates:
 mean in group ownNo mean in group ownYes
            47391.01             54934.68
```

There are several important pieces of information in the output of `t.test()`. First we see that the *t statistic* is −3.2, with a *p*-value of 0.0012. This means that the null hypothesis of *no difference* in income by home ownership is rejected. The data suggests that people who own their homes have higher income.

Next we see that the 95 % confidence interval for the difference is −3,007 to −12,080. If these are representative data of a larger population, we can have 95 % confidence that the group difference is between those values. Finally, we see the sample means for our data: mean income is $47,391 for the rent (`ownNo`) condition, and $54,935 for the ownership condition.

What about the difference within the Travelers segment? In Fig. 5.9, we saw that household income appeared to have a wider distribution among members of the Travelers segment who own homes than those who do not. Does that also reflect a difference in the mean income for the two groups? We use the filter `data=subset(data, condition)` to select just Travelers and repeat the test:

```
> t.test(income ~ ownHome, data=subset(seg.df, Segment=="Travelers"))

  Welch Two Sample t-test

data:  income by ownHome
t = 0.2656, df = 53.833, p-value = 0.7916
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8508.993 11107.604
sample estimates:
 mean in group ownNo mean in group ownYes
            63188.42             61889.12
```

The confidence interval of −8,508 to 11,107 includes 0, and thus we conclude—as evidenced in the *p*-value of 0.79—that there is not a significant difference in mean income among those Travelers in our data who own homes and who don't.

We might be puzzled: we saw in the first *t*-test that there *is* a significant difference in income based on home ownership, but in the second test that there's *no* significant difference within Travelers. Any difference must lie largely outside the Travelers group.

How can we locate where the difference lies? A *t*-test across all segments will not work because there are four segments and a *t*-test only compares two groups. We could test income within each segment, one at a time, but this is not a good idea because multiple tests increase the likelihood of finding a spurious difference (a "Type I error"). To track down the difference, we need a more robust procedure that handles multiple groups; we turn to that next.

6.5 Testing Multiple Group Means: ANOVA

An ANOVA compares the means of multiple groups. Technically, it does this by comparing the degree to which groups differ as measured by variance in their means (from one another), relative to the variance of observations around each mean (within each group). Hence the importance of *variance* in the name. More casually, you can think of it as testing for difference among multiple means, assuming that the groups have similar variance.

An ANOVA can handle single factors (known as *one-way* ANOVA), two factors (*two-way*), and higher orders including interactions among factors. A complete discussion of ANOVA would take more space than we have here, yet we use it to address our question from the previous section: which factors are related to differences in mean income in the segment data? Specifically, is income related to home ownership, or to segment membership, or both?

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. We look at income by home ownership first, and assign the `aov()` model to an object so we can use it with `anova()`. `aov()` uses the standard formula interface to model `income` as a response to `ownHome`:

```
> seg.aov.own <- aov(income ~ ownHome, data=seg.df)
> anova(seg.aov.own)
Analysis of Variance Table

Response: income
           Df     Sum Sq    Mean Sq F value   Pr(>F)
ownHome     1 4.2527e+09 4252661211  10.832 0.001118 **
Residuals 298 1.1700e+11  392611030
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1  1
```

The value of `Pr(>F)` for `ownHome` is the *p*-value and reflects that there is significant variation in `income` between those who do and do not own their own homes.

(This is a slightly different test but the same conclusion that we obtained from the *t*-test in Sect. 6.4).

What about income by segment? We model that and save the `aov` object:

```
> seg.aov.seg <- aov(income ~ Segment, data=seg.df)
> anova(seg.aov.seg)
Analysis of Variance Table

Response: income
            Df     Sum Sq    Mean Sq F value    Pr(>F)
Segment      3 5.4970e+10 1.8323e+10  81.828 < 2.2e-16 ***
Residuals 296 6.6281e+10 2.2392e+08
...
```

The value of `Pr(>F)` is very close to zero, confirming that income varies significantly by segment. (If you're wondering, `2.2e-16` means $2.2 * 10^{-16}$ and is the smallest non-zero number that R will typically report in Mac OS X. It is the value of the R constant `.Machine$double.eps` that expresses the tolerance of floating point differences.)

If income varies by *both* home ownership and segment, does that mean that a more complete model should include both? We can add both factors into the ANOVA model to test this:

```
> anova(aov(income ~ Segment + ownHome, data=seg.df))
Analysis of Variance Table

Response: income
            Df     Sum Sq    Mean Sq F value Pr(>F)
Segment      3 5.4970e+10 1.8323e+10 81.6381 <2e-16 ***
ownHome      1 6.9918e+07 6.9918e+07  0.3115 0.5772
Residuals 295 6.6211e+10 2.2444e+08
...
```

The results indicate that when we try to explain income differences in income by both `Segment` and `ownHome`, segment is a significant predictor ($p \ll 0.01$) but home ownership is *not* a significant predictor. Yet the previous results said that it *was* significant. What's the difference? What is happening is that segment and home ownership are not independent, and the effect is captured sufficiently by segment membership alone. Home ownership accounts for little more over and above what can be explained by `Segment`.

Could it be that home ownership is related to income in some segments but not in others? This would be represented in our model by an *interaction* effect. In a model formula, "+" indicates that variables should be modeled for main effects only. We can instead write ":" for an interaction or "*" for both main effect and interaction. We test main effects and interaction of home ownership and segment:

```
> anova(aov(income ~ Segment * ownHome, data=seg.df))
Analysis of Variance Table

Response: income
                Df     Sum Sq    Mean Sq F value Pr(>F)
```

```
Segment            3 5.4970e+10 1.8323e+10 81.1305 <2e-16 ***
ownHome            1 6.9918e+07 6.9918e+07  0.3096 0.5784
Segment:ownHome    3 2.6329e+08 8.7762e+07  0.3886 0.7613
Residuals        292 6.5948e+10 2.2585e+08
...
```

Again, segment is a significant predictor, while home ownership and the interaction of segment with home ownership are not significant. In other words, segment membership is again the best predictor on its own. We discuss interaction effects further in Chap. 7.

6.5.1 Model Comparison in ANOVA*

Another capability of the `anova()` command is to compare two or more models, using the syntax `anova(model1, model2, ...)` We can compare the `aov()` model with segment alone vs. the model with both segment and income:

```
> anova(aov(income ~ Segment, data=seg.df),
+       aov(income ~ Segment + ownHome, data=seg.df))
Analysis of Variance Table

Model 1: income ~ Segment
Model 2: income ~ Segment + ownHome
  Res.Df        RSS Df Sum of Sq      F Pr(>F)
1    296 6.6281e+10
2    295 6.6211e+10  1  69918004 0.3115 0.5772
```

This tells us that Model 2—which includes both segment and home ownership—is not significantly different in overall fit from Model 1. If it were better, the null hypothesis of no difference would be rejected, as shown by a p-value ("Pr(>F)") less than 0.05.

It is essential to note that model comparison as performed by the `anova()` command *only* makes sense in the case of nested models. In this context, a model A is *nested* within another model B when one or more parameters of B can be fixed or removed to yield model A. In the present case, `income ~ Segment` is nested within `income ~ Segment + ownHome` because we can remove `ownHome` and arrive at the former model. Because they are nested, the two models may be compared by `anova()` or other functions that perform likelihood comparisons.

The model `income ~ Segment` is *not* nested within `income ~ subscribe + ownHome` because no amount of removing or fixing parameters in the latter model will produce the former. Thus, those two models could not be compared by `anova()` in a meaningful way. If you try to compare them, R may produce some output but it is not generally interpretable.

The question of how to compare non-nested models is one we do not tackle in depth in this book, although it recurs in our discussion of structural models in Chap. 10.

If you wish to learn more about the issues and methods for general model comparison, a good place to start is to review the literature on the Akaike information criterion (AIC) and Bayesian information criterion (BIC). We review BIC briefly in Sect. 11.3.5.

6.5.2 Visualizing Group Confidence Intervals

A good way to visualize the results of an ANOVA is to plot confidence intervals for the group means. This will reveal more about whether the differences are substantial in magnitude or not. We use the `multcomp` (multiple comparison) package and its `glht(model)` (general linear hypothesis) command [79]. You may need to install the "multcomp" package on your system.

Let's take a look at what `glht()` does. We assign an `aov()` to an object and inspect it with `glht()`:

```
> library(multcomp)
> seg.aov <- aov(income ~ Segment, data=seg.df)
> glht(seg.aov)

   General Linear Hypotheses

Linear Hypotheses:
                       Estimate
(Intercept) == 0          53091
SegmentSuburb mix == 0     1943
SegmentTravelers == 0      9123
SegmentUrban hip == 0    -31409
```

There is a problem: the default `aov()` model has an intercept term (corresponding to the Moving up 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 "`-1`" to the model formula:

```
> seg.aov <- aov(income ~ -1 + Segment, data=seg.df)
> glht(seg.aov)

   General Linear Hypotheses

Linear Hypotheses:
                       Estimate
SegmentMoving up == 0     53091
SegmentSuburb mix == 0    55034
SegmentTravelers == 0     62214
SegmentUrban hip == 0     21682
```

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

```
> par(mar=c(6,10,2,2))   # adjusts margins to preserve axis labels
> plot(glht(seg.aov),
+      xlab="Income", main="Average Income by Segment (95% CI)")
```

The result is Fig. 6.1. The dot shows the mean for each segment, and bars reflect the confidence interval.

In Fig. 6.1 we see confidence intervals for the mean income of each segment. It is clear that the average income of Urban hip segment members is substantially lower than the other three groups.

6.5.3 Variable Selection in ANOVA: Stepwise Modeling*

Building models iteratively by adding and removing variables is a common task that can be automated with the `step(model)` command. This performs *stepwise* model selection by testing models one at time while changing the variables

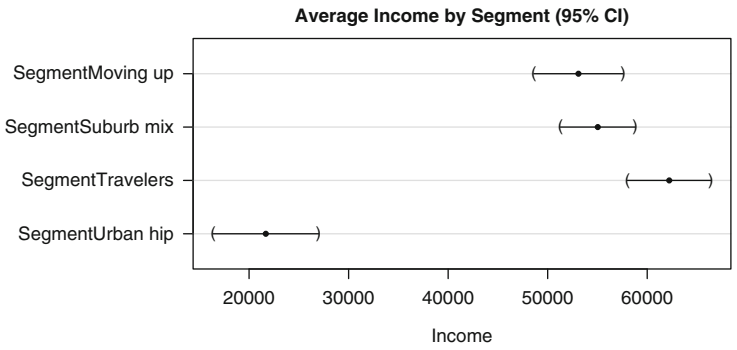

Fig. 6.1. Confidence intervals for income by segment, from an analysis of variance model with `aov()` and `glht()`.

in the model to see whether the change improves the model. There are options for both *backward* (starting with a larger set of variables and progressively cutting them) and *forward* (adding variables) procedures. The `step()` command uses the AIC to compare models on the basis of overall fit balanced with model complexity [3].

We perform a backward stepping procedure here (the default direction) by specifying a complete main effect model using the formula shorthand "`response ~ .`" The "`.`" is shorthand for "all other variables (except the response variable)." By default this models all main effects without interactions. Higher order effects in this case may be added with superscript notation, such as "`.^2`" for two-way interactions, but it is usually good to avoid such indiscriminate interaction modeling.

For our `aov()` model for income, the command to run the stepwise procedure for main effects and save the resulting best model is:

```
> seg.aov.step <- step(aov(income ~ ., data=seg.df))
Start:  AIC=5779.17
income ~ age + gender + kids + ownHome + subscribe + Segment

            Df  Sum of Sq        RSS    AIC
- age        1 4.7669e+06 6.5661e+10 5777.2
- ownHome    1 1.0337e+08 6.5759e+10 5777.6
- kids       1 1.3408e+08 6.5790e+10 5777.8
- subscribe  1 1.5970e+08 6.5816e+10 5777.9
- gender     1 2.6894e+08 6.5925e+10 5778.4
<none>                   6.5656e+10 5779.2
- Segment    3 1.9303e+10 8.4959e+10 5850.5

Step:  AIC=5777.19
income ~ gender + kids + ownHome + subscribe + Segment
... [several steps] ...
Step:  AIC=5772.02
income ~ Segment

          Df Sum of Sq        RSS    AIC
<none>                6.6281e+10 5772.0
- Segment  3 5.497e+10 1.2125e+11 5947.2
```

We see that `step()` started by modeling income with all six other variables, went through several steps of removing variables, and concluded with the "best" model as `income ~ Segment`.

We examine the result of `step()`, which was saved in a model object, using the standard `anova()` command:

```
> anova(seg.aov.step)
Analysis of Variance Table

Response: income
           Df     Sum Sq    Mean Sq F value    Pr(>F)
Segment     3 5.4970e+10 1.8323e+10  81.828 < 2.2e-16 ***
Residuals 296 6.6281e+10 2.2392e+08
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1  1
```

Stepwise procedures are not a panacea and must be used with caution, although they are sometimes helpful for model exploration. In more general cases—where there may be dozens, hundreds, or thousands of available variables—variable selection is better informed by procedures such as a lasso [152] or random forest [19] procedure. We examine random forest models in Chap. 11.

6.6 Bayesian ANOVA: Getting Started*

This is an advanced section that is primarily recommended for readers who have some familiarity with the principles of Bayesian analysis and seek an introduction to Bayesian models in R. We do *not* provide a comprehensive overview of the methods,

and assume that the reader is generally familiar with Bayesian concepts such as a *prior*, *posterior*, and *posterior sampling*.

For other readers, we attempt to give enough context to make the concepts approachable. Although this may be insufficient for a real project, it introduces how such models work and demonstrates the steps involved. We refer you to Sect. 6.7 for additional references.

6.6.1 Why Bayes?

We suggest analysts consider Bayesian analyses instead of traditional ("frequentist") statistical models when possible. Bayesian analysis is often a more direct way to tackle the questions we usually want to know: "Is this hypothesis likely to be true?", "How much confidence should I have?", and "What are the most likely values?" A Bayesian analysis does not take refuge in the double and triple negatives of traditional models ("we failed to reject the null hypothesis that there is no difference between the models"). Instead, it answers, "Given these data, how likely is the difference?"

Despite the advantages, there are reasons Bayesian analyses are not more common: there are fewer Bayesian teachers, texts, and practitioners; many Bayesian texts are dense with formulas; and the field is rapidly developing and some contentious issues have not been settled. Perhaps most importantly, available software packages are designed to make traditional models easy to run and that ease has not yet been brought to many areas of Bayesian practice. For an analyst, it may be easier and more productive to use traditional models in day-to-day work. Happily, Bayesian and traditional methods often lead to the same business conclusions (although not always).

R is on the forefront of making Bayesian methods more widely available. This is made possible by the many contributors to R, and by the R language itself which is well suited for the iterated analyses that Bayesian methods require. In this section, we demonstrate a starting point for a Bayesian version of ANOVA.

6.6.2 Basics of Bayesian ANOVA*

There are many options in R for Bayesian analyses (see the Bayesian task view on CRAN: `http://cran.r-project.org/web/views/`). The `MCMCpack` package is a robust, fast, and powerful Bayesian kit. However, we opt here to use the `BayesFactor` package for its simplicity. In particular, `BayesFactor` has sensible defaults for weakly informative prior probabilities [116, 139] and makes model comparison easy. You will need to install the `BayesFactor` package for the following code.

We use the `lmBF(formula, data)` command to specify our ANOVA model as a linear model for `income` modeled by the `Segment` factor:

```
> set.seed(96761)
> library(BayesFactor)
> seg.bf1 <- lmBF(income ~ Segment, data=seg.df)
```

We set a pseudorandom number seed because this function will take *draws* from the *posterior distribution*. What does that mean? Briefly, a common way to estimate a Bayesian model is to do repeated assessments of how well a proposed model fits the data.

To understand this we must consider the concept of a *parameter*. We have not used the term yet, but a statistical model estimates one or more parameters that define the presumed distribution. For example, a *t*-test compares the mean of two groups, and the parameter it estimates is the difference between the means. An ANOVA model can also be used to estimate the mean. It was confidence in the estimation of that parameter that we plotted in Sect. 6.5.2.

Common Bayesian models operate by selecting initially random values for model parameters (such as the mean for a segment). The process then retains the parameter according to the likelihood that it fits the data and prior expectation (an estimated starting point, if we have one), and iterates that process thousands or even millions of times. The retained estimates are the *draws* from the posterior distribution for the parameters, while the final estimated distribution of them is the *posterior distribution*. The end result is a large sample of possible parameters and their likelihoods, or in other words, an outline of the most likely parameters for a given model. Again, see Sect. 6.7 for more.

After fitting the model for `income ~ Segment`, we might inspect it directly. However, instead of starting to interpret a model, it is preferable to have a sense that it is an *adequate* model. So we first compare it to the alternative we considered in Sect. 6.5.1, which modeled `income ~ Segment + ownHome`. We would then interpret the Segment-only model if it fits the data better (or fits just as well but is simpler).

Model comparison in `BayesFactor` is performed by using the "/" operator to find the ratio of the models' Bayes Factors. We have the first model `seg.bf1` from above, and now fit the second model with two factors that we wish to compare:

```
> seg.bf2 <- lmBF(income ~ Segment + ownHome, data=seg.df)
  |============================================================| 100%
> seg.bf1 / seg.bf2
Bayes factor analysis
--------------
[1] Segment : 6.579729 1.62%

Against denominator:
  income ~ Segment + ownHome
---
Bayes factor type: BFlinearModel, JZS
```

This tells us that the ratio of Bayes Factors for model 1 (`~ Segment`) vs. model 2 (`~ Segment + ownHome`) is 6.58. This means that model 1 is the preferable model by a factor of 6.5.

To find the model parameters and their credible ranges, we use the `posterior(model, index, draws)` command to draw 10,000 samples of the possible parameters from model 1:

```
> seg.bf.chain <- posterior(seg.bf1, 1, iterations = 10000)
  |============================================================| 100%
```

The draws are known as a *chain* because they are estimated by a Markov chain process; we skip those details (see [61]).

Before we examine the estimates, we should inspect whether the draws *converged* to stable values such that the estimates are reliable. In `BayesFactor`, we simply call `plot()` on the chain object. We select columns 1:6 from the draws because there are six parameters we care about: the population mean and variance (*mu* and *sigma*) and the estimates of means for the four segments:

```
> plot(seg.bf.chain[, 1:6])   # check console: may need <Return> to see all
```

The charts for the first three parameters are shown in Fig. 6.2; we omit the other three charts because they are nearly identical. We interpret the charts as follows. On the left, we see the estimated parameter values (*Y* axis) plotted against the draw sequence (*X* axis). These form a fat but straight line, which means the estimates varied around a stable central point; thus, they converged. (If they had not converged, the plot would show erratic variations up or down, or would spread out increasingly rather than being straight.)

On the right, we see a density plot of the values. The density shape is approximately normal, which matches the assumption of the regression model. Thus, the charts confirm that the model was stable and converged (note that these don't mean the model is *useful*, only that it achieved a stable estimate).

6.6.3 Inspecting the Posterior Draws*

We now examine the parameters as expressed in our posterior draw chain. A simple `summary()` of the chain shows us the estimates:

```
> summary(seg.bf.chain)

Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

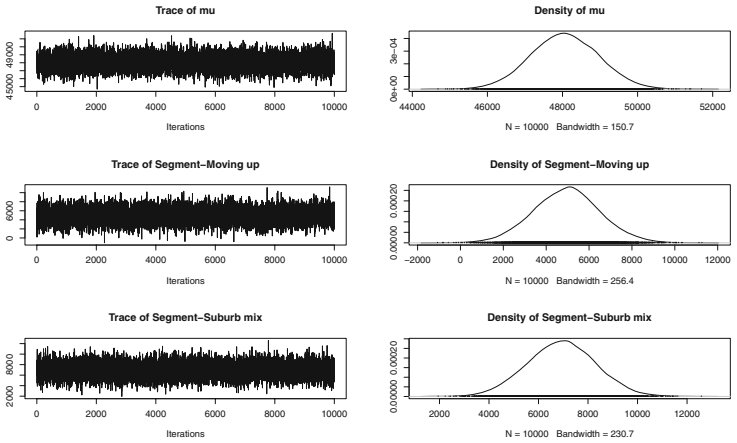

Fig. 6.2. Trace plot for draws from the posterior distribution of a Bayesian ANOVA for income by segment, for the first three parameters. The *left-hand charts* show trace convergence; *right-hand charts* show the posterior distributions for the parameters.

```
                          Mean        SD  Naive SE Time-series SE
mu                    4.806e+04 8.969e+02 8.969e+00      8.804e+00
Segment-Moving up     4.951e+03 1.548e+03 1.548e+01      1.548e+01
Segment-Suburb mix    6.927e+03 1.373e+03 1.373e+01      1.373e+01
Segment-Travelers     1.398e+04 1.487e+03 1.487e+01      1.518e+01
Segment-Urban hip    -2.586e+04 1.777e+03 1.777e+01      1.956e+01
sig2                  2.259e+08 1.856e+07 1.856e+05      1.856e+05
g_Segment             2.138e+00 3.359e+00 3.359e-02      3.359e-02

2. Quantiles for each variable:

                           2.5%        25%        50%        75%      97.5%
mu                    4.631e+04  4.745e+04  4.805e+04  4.868e+04  4.982e+04
Segment-Moving up     1.925e+03  3.916e+03  4.968e+03  5.961e+03  8.054e+03
Segment-Suburb mix    4.243e+03  5.996e+03  6.934e+03  7.857e+03  9.608e+03
Segment-Travelers     1.104e+04  1.297e+04  1.399e+04  1.499e+04  1.690e+04
Segment-Urban hip    -2.934e+04 -2.703e+04 -2.586e+04 -2.466e+04 -2.239e+04
sig2                  1.923e+08  2.128e+08  2.249e+08  2.378e+08  2.647e+08
g_Segment             3.765e-01  7.949e-01  1.298e+00  2.284e+00  8.738e+00
```

The first section of the summary ("1. Empirical mean and ...") gives arithmetic central tendency estimates for the 10,000 draws of each of the parameters in the chain: the mean of each parameter, the standard deviation of that estimate across the 10,000 draws, and so forth. The second result ("Quantiles ...") is what we prefer to use instead; it reports the actual observed quantiles for each of the parameters.

Note that the model estimates an overall `mu` that is the best guess for the population mean regardless of segment effects, and then estimates each segment as a deviation from that. However, for many purposes, it is more useful to have direct estimates for the mean of each segment rather than its deviation. To estimate the direct values for each segment, we add the population value (`mu`) to the deviations for each segment.

However, we cannot simply do that with the *aggregate* numbers here by adding the `mu` row to each of the other rows. Why not? Because the best estimates of segment totals are found *within* each draw; we need to compute segment values at that level and then summarize those estimates. Luckily that is easy to do in R.

To see how, let's examine the chain object:

```
> head(seg.bf.chain)
...
          mu Segment-Moving up Segment-Suburb mix Segment-Travelers  ...
[1,] 48055.75         4964.3105           6909.032          13983.21  ...
[2,] 47706.52         6478.1497           7816.873          12160.32  ...
[3,] 48362.90         5228.0718           6654.030          12565.87  ...
[4,] 49417.43         5300.9543           7249.228          12218.89  ...
...
```

We see rows (10,000 in all) for the draws, and columns for the estimates for each segment. By indexing the chain, we confirm that it is arranged as a matrix:

```
> seg.bf.chain[1:4, 1:5]
          mu Segment-Moving up Segment-Suburb mix Segment-Travelers  ...
[1,] 48055.75          4964.310           6909.032          13983.21  ...
[2,] 47706.52          6478.150           7816.873          12160.32  ...
[3,] 48362.90          5228.072           6654.030          12565.87  ...
[4,] 49417.43          5300.954           7249.228          12218.89  ...
```

This means that simple math will work to find within-draw estimates for each row. We do this by adding column 1, the population estimate, to each of the other columns 2–5. We test this first on rows `1:4` only:

```
> seg.bf.chain[1:4, 2:5] + seg.bf.chain[1:4, 1]
     Segment-Moving up Segment-Suburb mix Segment-Travelers Segment-Urban hip
[1,]          53020.06           54964.78          62038.95          22199.20
[2,]          54184.67           55523.40          59866.84          21251.18
[3,]          53590.97           55016.93          60928.77          23914.93
[4,]          54718.38           56666.66          61636.32          24648.35
```

It works, so now we compute that total for all rows and assign the result to a new object. Then we get quantiles from that object as the overall best estimates of segment income:

```
> seg.bf.chain.total <- seg.bf.chain[, 2:5] + seg.bf.chain[, 1]
> seg.bf.ci <- t(apply(seg.bf.chain.total, 2,
+                      quantile, pr=c(0.025, 0.5, 0.975)))
> seg.bf.ci
                         2.5%      50%    97.5%
Segment-Moving up  49582.08 53020.98 56522.05
Segment-Suburb mix 52039.66 54988.99 57867.29
Segment-Travelers  58799.46 62048.33 65355.62
Segment-Urban hip  17992.85 22216.26 26450.56
```

In the `apply()` command, we applied the `quantile()` function to the columns with the probabilities that we wanted for a 95 % credible interval. Then we transposed the result with `t()` to be more readable (treating the segments as "cases").

Those values are the best estimates of the 95 % credible range for the estimate of *average income* as modeled by segment, under the assumptions of our model.

6.6.4 Plotting the Bayesian Credible Intervals*

We can plot the credible intervals from the previous section using the capability of the `ggplot2` package to plot error bars. Install the "ggplot2" package if needed. The `ggplot2` commands work best with data frames, so we coerce our credible interval object `seg.bf.ci` to a data frame and add a column for segment names:

```
> library(ggplot2)
> seg.bf.df <- data.frame(seg.bf.ci)
> seg.bf.df$Segment <- rownames(seg.bf.df)
```

Now we construct the chart in three steps. We add elements corresponding to the values of segment quartiles in the summary data frame:

```
> p <- ggplot(seg.bf.df, aes(x=Segment, y=X50., ymax=X97.5., ymin=X2.5.))
```

We add points for the *y* values (the estimated median in this case), and add the 2.5 % and 97.5 % quartiles as "error bars" (which are automatically associated with the names `ymax` and `ymin` as we set above):

```
> p <- p + geom_point(size=4) + geom_errorbar(width=0.2) + ylab("Income")
```

Finally we draw that plot object while adding a title and flipping the plot coordinates so the segments are nicely on the left:

```
> p + ggtitle("95% CI for Mean Income by Segment") + coord_flip()
```

The result is Fig. 6.3, a chart that is easy to explain yet comes from a powerful underlying Bayesian model.

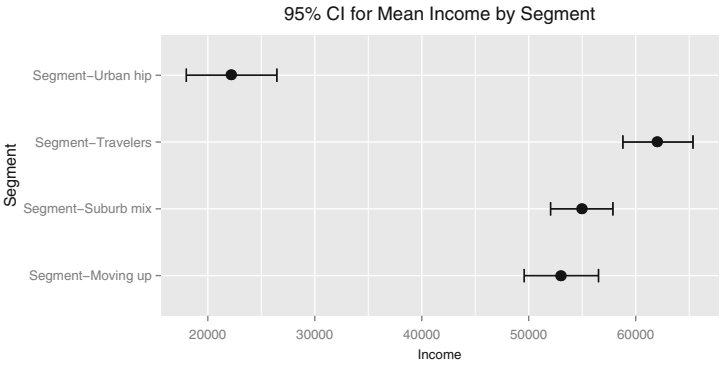

Fig. 6.3. Using `ggplot2` to plot the credible intervals for income by segment from the Bayesian posterior draws.

You might notice that the Bayesian results in Fig. 6.3 are not all that different from the classical results in Fig. 6.1. This is to be expected because they come from the same data. In fact, if the model is exactly correct and the population is infinite, then as the sample size approaches infinity, the Bayesian and classical confidence intervals will be the same.

In that case, why one would want to use the Bayesian approach? One answer will come in Chaps. 7 and 13 when we introduce *hierarchical* methods that are more flexibly modeled in a Bayesian framework. Another answer is that data are never infinite, and in our opinion Bayesian models more directly address confidence in models for the data you actually have.

As you can see, R provides powerful capability for Bayesian analysis. R's open-source structure has made it easier for the software to keep pace with a rapidly evolving field. If you run into limitations with existing packages, you can use R's programming language to accomplish tasks (as we did here to compute posterior draws for total segment income).

6.7 Learning More*

t-tests and ANOVA are nothing more than flavors of general linear models, which we cover in more depth in Chap. 7. In the R domain, there are many books on linear models. A readable text that focuses on understanding basic models and getting them right is Fox and Weisberg's *An R Companion to Applied Regression* [51].

For categorical data analysis, which we briefly sampled with our discussion of binomial distribution and chi-square tests, the best starting place—although not specific to R—is Agresti's *An Introduction to Categorical Data Analysis* [1].

Readings on Bayesian data analysis vary tremendously in mathematical prerequisites and authors' styles. Kruschke's *Doing Bayesian Data Analysis* [94] is a textbook that uses R and builds intuition from the ground up with only high-school level mathematics. It is a lengthy and thorough exposition of Bayesian thinking. A standard text that moves faster with more mathematics is Gelman et al., *Bayesian Data Analysis* [61]. For advanced marketing applications, especially hierarchical linear models and choice models, a standard text is Rossi, Allenby, and McCulloch's *Bayesian Statistics and Marketing* [137].

We presented charts in this chapter using the `lattice` and `ggplot2` packages. Each of them is described in detail in an eponymous book: Sarkar's *Lattice* [141] and Wickham's *ggplot2* [162].

6.8 Key Points

This chapter introduced formal statistical tests in R. Following are some of the important lessons.

To perform statistical tests on differences by group:

- `chisq.test()` (Sect. 6.2) and `binom.test()` (Sect. 6.3) find confidence intervals and perform hypothesis tests on tables and proportion data, respectively. The `binom` package offers options such as Agresti–Coull and Bayesian versions of binomial tests that may be more informative and robust than standard exact binomial tests (Sect. 6.3).
- 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) (Sect. 6.4).
- 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()` (Sect. 6.5).
- The `anova()` command is also useful to compare two or more ANOVA or other linear models, provided that they are nested models (Sect. 6.5.1).
- Stepwise model selection with `step()` is one way to evaluate a list of variables to select a well-fitting model, although we recommend that it be used with caution as other procedures may be more appropriate (Sect. 6.5.3).

We reviewed a few advanced topics for statistical models and data visualization:

- Plotting a `glht()` object from the `multcomp` package is a good way to visualize confidence intervals for ANOVA models (Sect. 6.5.2).
- A relatively straightforward starting point for Bayesian ANOVA and other linear models is the `BayesFactor` package (Sect. 6.6).
- Bayesian models should be evaluated for the stability and distribution of their estimated parameters using trace and density plots (Sect. 6.6).
- Credible intervals (and other types of intervals) may be plotted with the `ggplot2` option to add `geom_errorbar()` lines for groups (Sect. 6.6.4).