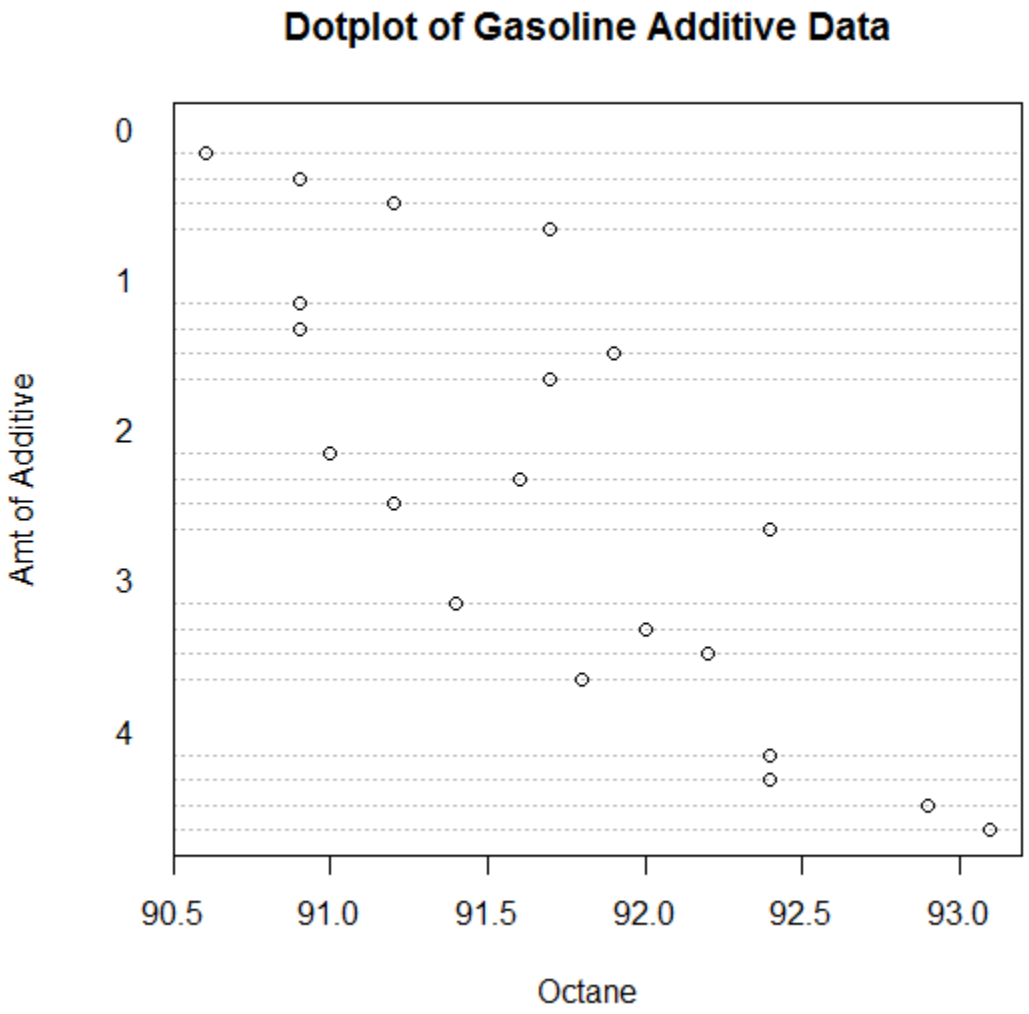
Output from gasadd1.R

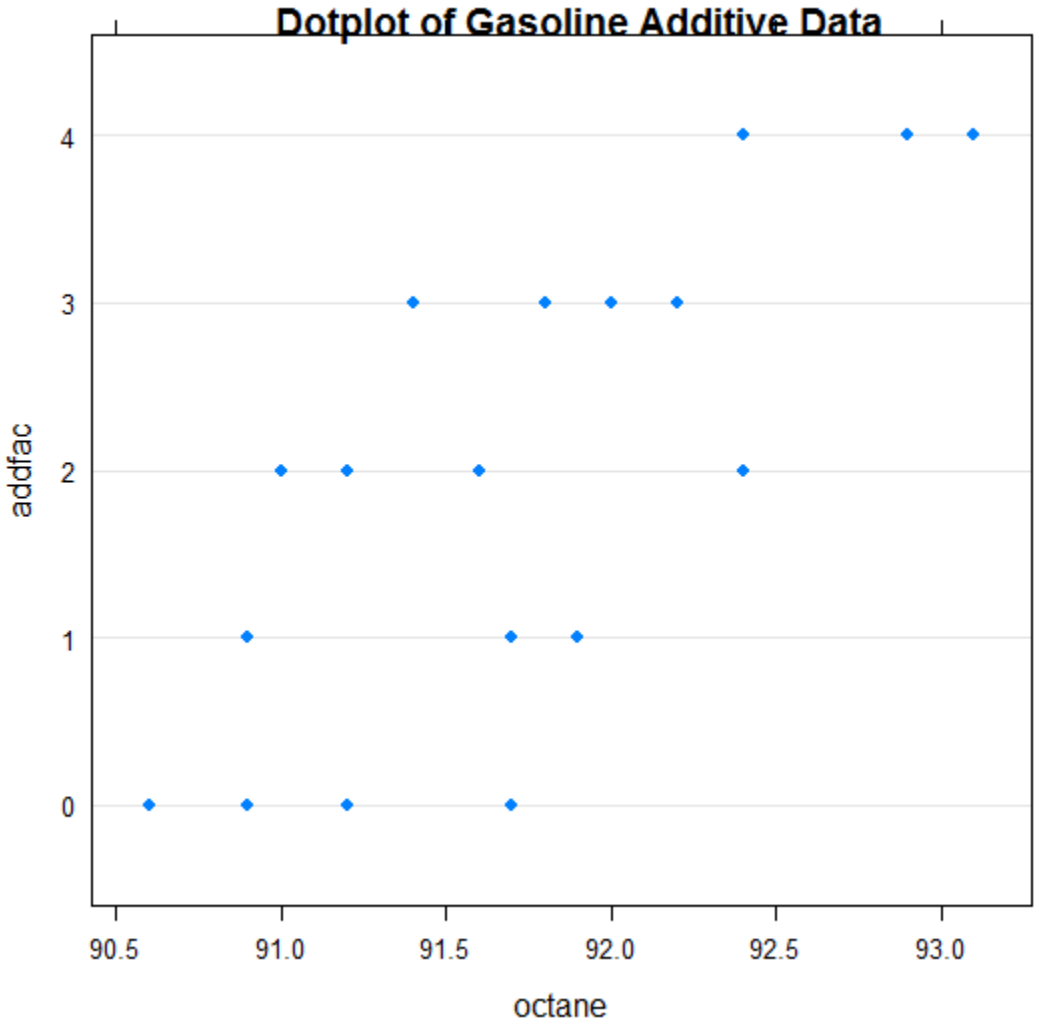
```
> library(nlme) # In R there are user-contributed "packages" of function
>               # that do extend the basic capabilities of R with all sorts of
>               # different statistical methods. Some packages are built into the
>               # basic distribution of R (nlme is one such package), others
>               # have to be downloaded and installed. Either way, the functions
>               # and capabilities in a given package are not available in an R
>               # session until the pakcage has been loaded with the command
>               # library(package_name).
>               # The nlme package is a big one with lots of tools for
>               # exploring grouped or clustered data, and for fitting
>               # linear and nonlinear mixed effect models. For this example,
>               # we are just fitting a simple one-way analysis of variance
>               # model, which is a linear model (not a mixed model at all)
>               # but there are couple of capabilities in the nlme package
>               # that are still convenient and useful here.
>
> # other useful packages used in this script:
> library(MASS)
> library(car)
> library(lsmeans)
>
> gasdata <- read.table(file="gasadd.dat",header=T) # read the data from an
>                                                   # external file and grab the
>                                                   # variable names from the
>                                                   # 1st line of that file.
>                                                   # The data are placed into
>                                                   # a "data frame", which I've
>                                                   # called gasdata
> head(gasdata) #look at the first few rows of gasdata
  add octane
1   0   91.7
2   0   91.2
3   0   90.9
4   0   90.6
5   1   91.7
6   1   91.9
>
> gasdata$addfac <- factor(gasdata$add, ordered=T)
>                       # by default a numeric variable is not treated when
>                       # read in, so make a copy of the add variable that
>                       # R will now treat as a factor, with levels that
>                       # are assumed to be ordered (rather than qualitative)
>
>
> attach(gasdata) # this makes the variables within gasdata available
The following object(s) are masked from 'gasdata (position 3)':

    add, addfac, octane
The following object(s) are masked from 'gasdata (position 4)':

    add, addfac, octane
The following object(s) are masked from 'gasdata (position 5)':

    add, addfac, octane
```

```
The following object(s) are masked from 'gasdata (position 10)':

    add, addfac, octane
The following object(s) are masked from 'gasdata (position 11)':

    add, octane
>                 # so you can refer to them as add or octane rather than
>                 # gasadd$add or gasadd$octane
>
> dotchart(octane,groups=addfac) # plot the octane values with a dotplot
> title(main="Dotplot of Gasoline Additive Data",xlab="Octane",
+   ylab="Amt of Additive") # titles and x and y axis labels can be added
>                           # with the title() function or the main, xlab
>                           # and ylab arguments can be added to the original
>                           # dotchart() function call
```

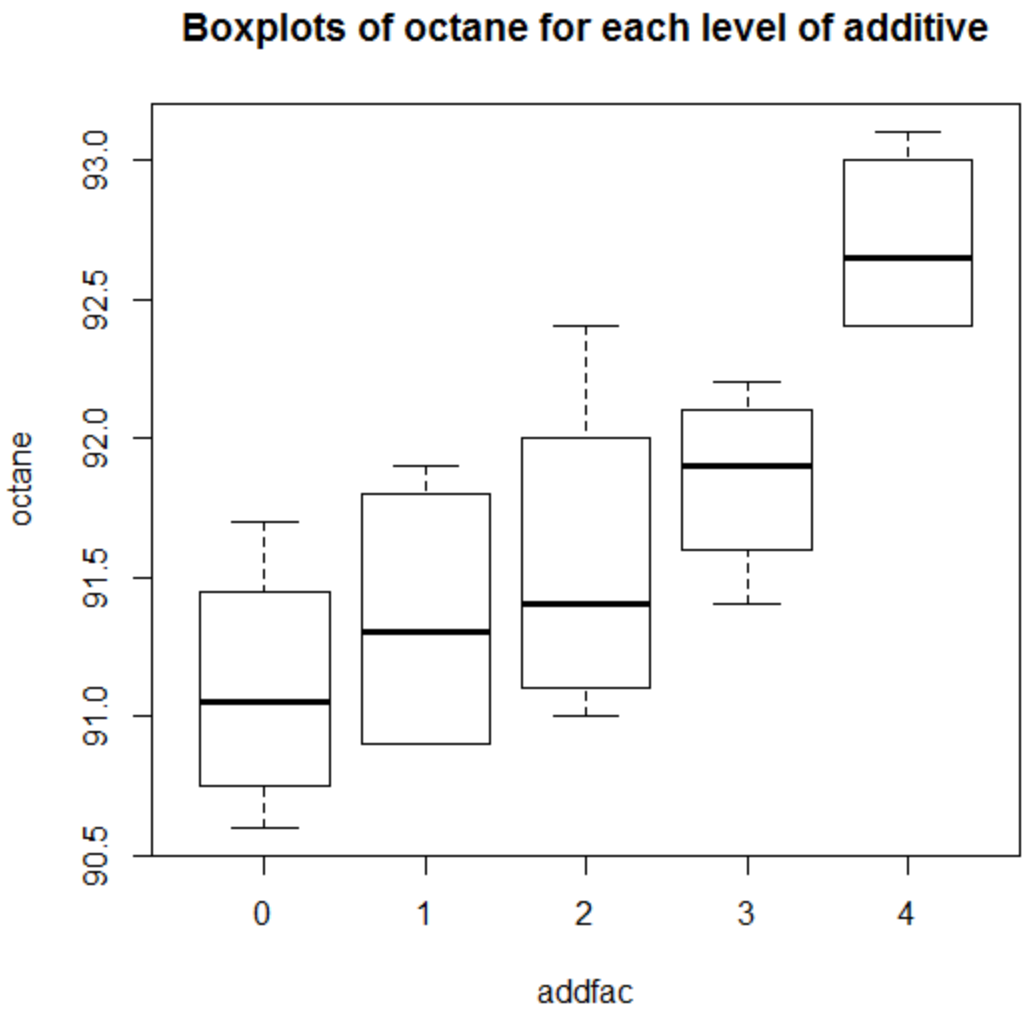



```
>
> # I like the dotplot produced by default when you plot the data after they
> # have been turned into a groupedData object. Such an object is a data frame
> # but with extra information attached to it that comes in the form of a
> # formula that identifies the response variable, (octane below), the
> # covariate or explanatory variable that we want to regard as the "primary"
> # covariate (for plotting and modeling purposes) and any grouping structure.
> # Below the formula is "octane~1|addfac". Here the "~" plays the role of an
> # equal sign, octane is the response (or y variable), 1 represents an
```

```
> # intercept or constant term, and what comes after the vertical bar is
> # the factor that identifies groups.
>
> gasdata2 <- groupedData(octane~1|addfac, gasdata)
>
> plot(gasdata2) # R is an "object-oriented" computing language. That means that
>                # quantities that we specify are treated as one of many
>                # different types of objects, and functions that operate
>                # on those quantities figure out what to do based upon what
>                # kind of object that quantity is. E.g., the plot function
>                # figure out how to plot its argument based upon the argument's
>                # object type. In this case, gasdata2 is a groupedData
>                # object, and the default way to handle it is to produce
>                # a dotplot.
> title(main="Dotplot of Gasoline Additive Data")
```




```
>
> # side-by-side boxplots are also useful:
>
> plot(octane~addfac,data=gasdata) # this shows another use of the plot function
>                                  # it can take a formula as argument and
>                                  # figure out what to do
> title(main="Boxplots of octane for each level of additive")
```




```
>
> # fit a one way anova model using the aov() function. The aov() function
> # is just a wrapper for the lm() function, which fits linear models of all
> # sorts of varieties (regression models, anova models, ancova models).
> # That is, aov() calls lm() where the model fitting is actually done. But
> # this is convenient because the default reporting of a fitted lm is not as
> # suitable for our purposes (not as suitable for an anova model) as the
> # default reporting of a fitted aov model
> m1 <- aov(octane~addfac,gasdata)
> m1    # this "prints" the fitted model
Call:
   aov(formula = octane ~ addfac, data = gasdata)

Terms:
                  addfac Residuals
Sum of Squares     6.108     3.370
Deg. of Freedom        4        15

Residual standard error: 0.4739902
Estimated effects are balanced
> summary(m1) # this summarizes the fitted model.
            Df Sum Sq Mean Sq F value Pr(>F)
addfac       4  6.108  1.5270   6.797 0.0025 **
Residuals   15  3.370  0.2247
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> coefficients(m1) # this gives the actual fitted model parameters
(Intercept)    addfac.L    addfac.Q    addfac.C    addfac^4
91.71000000  1.17004273  0.34743961  0.18973666  0.03585686
>
> summary.lm(m1) # this summarizes the fitted model treating it as generic lm

Call:
aov(formula = octane ~ addfac, data = gasdata)

Residuals:
   Min     1Q Median     3Q    Max
-0.550 -0.375  0.000  0.350  0.850

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 91.71000    0.10599 865.291  < 2e-16 ***
addfac.L     1.17004    0.23700   4.937 0.000179 ***
addfac.Q     0.34744    0.23700   1.466 0.163291
addfac.C     0.18974    0.23700   0.801 0.435865
addfac^4     0.03586    0.23700   0.151 0.881757
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.474 on 15 degrees of freedom
Multiple R-squared: 0.6444,     Adjusted R-squared: 0.5496
F-statistic: 6.797 on 4 and 15 DF,  p-value: 0.002499

>                # not necessarily an aov model. This is an easy way to get some
>                # additional useful output.
>
>
>
> # now estimate the mean for each level of additive:
> lsmeans(m1, specs = ~ addfac)
$`addfac lsmeans`
 addfac lsmean        SE df lower.CL upper.CL
      0  91.10 0.2369951 15 90.59486 91.60514
      1  91.35 0.2369951 15 90.84486 91.85514
      2  91.55 0.2369951 15 91.04486 92.05514
      3  91.85 0.2369951 15 91.34486 92.35514
      4  92.70 0.2369951 15 92.19486 93.20514

>
>
> # a contrast can be tested using the linearHypothesis function from the
> # car package. We need to define contrasts we are interested and then
> # stack them up as the rows of a matrix which we'll call hypmat1
> # A command like
> # x <- 3
> # assigns 3 to the variable x and produces no output
> # If we put parentheses around that assignment, the assignment is done
> # but in addition, R prints out the value of the variable after the
> # assignment has been executed:
> # (x <- 3)
> # Still assigns 3 to x, but also prints out 3
```

```
>
> ( hypmat1 <- matrix( c(0,1,0,0,0,  0,0,1,0,0,  0,0,0,1,0,  0,0,0,0,1),
+     nrow=4,byrow=T ) )
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    0    0    0
[2,]    0    0    1    0    0
[3,]    0    0    0    1    0
[4,]    0    0    0    0    1
>
> linearHypothesis(m1,hypothesis.matrix=hypmat1) # main effect of additive
Linear hypothesis test

Hypothesis:
addfac.L = 0
addfac.Q = 0
addfac.C = 0
addfac^4 = 0

Model 1: restricted model
Model 2: octane ~ addfac

  Res.Df   RSS Df Sum of Sq      F   Pr(>F)
1     19 9.478
2     15 3.370  4     6.108 6.7967 0.002499 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> ( hypmat2 <- hypmat1[2:4,] )
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    1    0    0
[2,]    0    0    0    1    0
[3,]    0    0    0    0    1
> linearHypothesis(m1,hypothesis.matrix=hypmat2) # nonlinear effects of add
Linear hypothesis test

Hypothesis:
addfac.Q = 0
addfac.C = 0
addfac^4 = 0

Model 1: restricted model
Model 2: octane ~ addfac

  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     18 4.002
2     15 3.370  3     0.632 0.9377  0.447
>
> # fit simple linear regression of octane on amount of additive
> m2 <- lm (octane~add,data=gasdata)
> summary(m2)

Call:
lm(formula = octane ~ add, data = gasdata)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7100 -0.3875 -0.0600  0.3825  0.7300
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 90.97000    0.18262 498.139  < 2e-16 ***
add          0.37000    0.07455   4.963 0.000101 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4715 on 18 degrees of freedom
Multiple R-squared: 0.5778,     Adjusted R-squared: 0.5543
F-statistic: 24.63 on 1 and 18 DF,  p-value: 0.0001006

>
> options(op) #reset contrasts to what they were to start (the default)
>
```