An Analysis of Adverstising.cvs (with quadratic terms)

Levi Lopes de Lima

2025-07-29

Describing the lab

We revisit the data set *Advertising.csv* available here and perform the corresponding analysis following the guidelines suggested in (James et al. 2013, vol. 112, chap. 3). This data set displays *200* observations of the *sales* (in thousands of units) of a certain good together with the corresponding expenditures (in thousands of dollars) distributed between three types of media (*TV*, *radio* and *newspaper*), thus forming a **design matrix** of size *200x4*. The central problem here is to check whether any of these media somehow influences the sales and to what extent. We examine this by first applying **backward selection** to the fitted least squares linear model (which confirms that spending with *newspaper* fails to be statistically significant) and then incorporating interactions of the remaining regressors (*TV* and *radio*) up to second order, as we find that this quadratic model is statistically more significant than the original strictly linear model. In particular, this quadratic enhancement of the model, although not as interpratable as the linear version, in principle allows us to numerically optimize the **mean response** for *sales* for a given value of the total expenditure *TV+radio*, at least if we remain in the **interpolation range** covered by the data, thus providing a substantial gain in predictive power.

All the math needed to understand the analysis may be found in (Lima 2023), which is available here.

A glimpse at the theory

Before embarking into the data analysis proper, we find it convenient to recall the theoretical underpinnings of a **linear regression model**, which relies on the system of equations

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \mathrm{e}_i, \quad i = 1, \cdots, n,$$

corresponding to n observations of p features (regressors, predictors, independent/explanatory variables, etc.) of an underlying population which can be arranged in the $n \times (p+1)$ **design matrix**

$$\mathfrak{x} = \left(\begin{array}{c|c} & \mathbf{x}_1 \\ \mathbf{1} & \vdots \\ & \mathbf{x}_n \end{array} \right)$$

where each $\mathbf{x}_i = (x_{i1}, \cdots, x_{ip})$ is a row p-vector (representing the outcome of the i^{th} measurement) and $\mathbf{1}$ is the column n-vector whose entries all equal 1. If we suspect that these features explain a response y_i (regressand, dependent/explained variable, etc.) which has also been observed, thus yielding an n-vector $\mathbf{y}$, we may pose ourselves the (exceedingly important!) problem of *predicting* the response at some unobserved regressor by *best fitting* a (possibly non-linear) functional dependence, say $\mathbf{y} = F(\mathfrak{x})$, to the available data $(\mathfrak{x}, \mathbf{y})$. The simplest choice is to postulate that F is **linear**, thus yielding the system above, where the error $\mathbf{e}$ entails the unobserved noise and the unknown parameter vector $\beta = (\beta_0, \beta_1, \cdots, \beta_p)$ should be estimated somehow. If we view $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ as a cloud of points in $\mathbb{R}^p \times \mathbb{R} = \mathbb{R}^{p+1}$, the **least squares** best fitting proposal leads to

$$\widehat{\beta} := \mathrm{argmin}_\beta \|\mathbf{y} - \mathfrak{x}\beta\|^2$$

as the natural estimator for β. Under the assumption that the **design matrix** $\mathfrak{x}$ has full column-rank ($= p+1$), which in particular implies the *low-dimensional* condition $p < n$, one sees that

$$\widehat{\beta} = (\mathfrak{x}^T\mathfrak{x})^{-1}\mathfrak{x}^T\mathbf{y},$$

which turns out to be the natural candidate for estimating β.

But how good is $\widehat{\beta}$ as an estimator? To ponder on this question, first posed by Gauss in his writings on the applications of the method, we follow the usual statistical route and view the observations $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ as **i.i.d.** copies of the same $\mathbb{R}^p \times \mathbb{R}$-valued random vector $(\mathscr{X}, \mathscr{Y})$ whose distribution is of course unknown. Under mild conditions on the random error (zero mean plus homoscedasticity) we are able to check that $\widehat{\beta}$ is an **unbiased** estimator for β so in particular $\mathsf{x}^T\widehat{\beta}$ is an **unbiased** estimator for the **mean response**

$$\mathsf{x}^T\beta = \mathbb{E}(\mathscr{Y}|_{\widetilde{\mathscr{X}}=\mathsf{x}}),$$

where $\widetilde{\mathscr{X}} = (1, \mathscr{X})$ and $\mathsf{x} = (1, \mathsf{x}_1, \cdots, \mathsf{x}_n)$.

Clearly, this solves in a satisfactory manner the problem of characterizing $\widehat{\beta}$ as a good **point** estimator, but the problem remains of evaluating its fluctuation around the true parameter β. With this goal in mind, we proceed by further assuming **normality** of the error, $\mathbf{e} \sim \mathcal{N}_n(\vec{0}, \sigma^2 \mathrm{I})$, and looking at

- the **fitted vector** $\widehat{\mathbf{y}} = \mathfrak{x}\widehat{\beta}$, which lies in $C(\mathfrak{x}) \subset \mathbb{R}^n$, the column-space of $\mathfrak{x}$ (recall that $C(\mathfrak{x})$ is assumed to have *maximal* dimension $p+1$);
- the **residual vector**, $\hat{\mathbf{e}} := \mathbf{y} - \widehat{\mathbf{y}}$, the orthogonal projection of $\mathbf{y}$ (and hence of $\mathbf{e}$ as well) onto the orthogonal complement of $C(\mathfrak{x})$ in $\mathbb{R}^n$.

Note that $\widehat{\mathbf{y}} = H\mathbf{y}$, where

$$H = \mathfrak{x}(\mathfrak{x}^T\mathfrak{x})^{-1}\mathfrak{x}^T$$

is the **hat matrix**, a symmetric and idempotent matrix defining the orthogonal projection from $\mathbb{R}^n$ onto $C(\mathfrak{x})$. Since $\hat{\mathbf{e}} = (\mathrm{I} - H)\mathbf{y}$, the **residual sum of squares** $RSS := \|\hat{\mathbf{e}}\|^2$ may be interpreted as a measure of how much variation in $\mathbf{y}$ is left unexplained by the linear regression model, which provides the fitted vector $\widehat{\mathbf{y}} \in C(\mathfrak{x})$. In any case, from the normality of $\mathbf{e}$ we deduce without much difficulty that:

- $\hat{\mathbf{e}} \sim \mathcal{N}_{n-p-1}(\vec{0}, \sigma^2 \mathrm{I})$;
- $\widehat{\beta} \sim \mathcal{N}_n(\beta, \sigma^2 \mathfrak{s})$, where $\mathfrak{s} := (\mathfrak{x}^T\mathfrak{x})^{-1}$ is the inverse of the **Gram matrix** $\mathfrak{x}^T\mathfrak{x}$ (in particular, the **covariance matrix** of $\widehat{\beta}$ is $\mathrm{cov}(\widehat{\beta}) = \sigma^2\mathfrak{s}$);
- $(n-p-1)\frac{\widehat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1}$, where $\widehat{\sigma}^2 := \frac{\|\hat{\mathbf{e}}\|^2}{n-p-1}$ is an **unbiased** estimator for the unknown error variance σ^2;
- $\{\widehat{\beta}, \hat{\mathbf{e}}\}$ and $\{\widehat{\beta}, \widehat{\sigma}^2\}$ are both independent.

From this, the standard inferential package for the least squares estimate $\widehat{\beta}$ is readily available. In particular, this provides:

- **confidence intervals** for the parameters:

$$\beta_i \in \left[\widehat{\beta}_i \mp \mathfrak{t}_{n-p-1,\delta/2}\widehat{\sigma}\sqrt{\mathfrak{s}_{ii}}\right] \text{ with prob. } 1-\delta,$$

  as displayed in the standard R summary;

- **confidence intervals** for the mean response:

$$\mathrm{x}^T\beta \in \left[\mathrm{x}^T\widehat{\beta} \mp \mathfrak{t}_{n-p-1,\delta/2}\widehat{\sigma}\sqrt{\mathrm{x}^T\mathfrak{s}\mathrm{x}}\right] \text{ with prob. } 1-\delta,$$

  where x is as above;

- all the standard **hypothesis tests** commonly employed (and used below) to carry out **variable selection** in a least squares model, at least if we work in the *low-dimensional* regime determined by $p \ll n$.

With these theoretical preliminaries out of the way, we now turn to the data…

Loading the packages

```
library(tidyverse)
library(ellipse)
library(patchwork)
library(knitr)
library(kableExtra)
library(crayon)
```

Viewing the data

```
data_adv <- read.csv("Advertising.csv")
str(data_adv)
```

```
## 'data.frame':    200 obs. of  5 variables:
##  $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ TV       : num  230.1 44.5 17.2 151.5 180.8 ...
##  $ radio    : num  37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...
##  $ newspaper: num  69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...
##  $ sales    : num  22.1 10.4 9.3 18.5 12.9 7.2 11.8 13.2 4.8 10.6 ...
```

Note that the variable *X* is an integer that merely labels the observations so it should not be included in the analysis. To check this, let us look at the data from the proper perspective (putting observations along the lines and features along the columns and isolating the first few observations, thus in alignment with the way the design matrix $\mathfrak{X}$ was defined…

```
head(data_adv)
```

```
##   X    TV radio newspaper sales
## 1 1 230.1  37.8      69.2  22.1
## 2 2  44.5  39.3      45.1  10.4
## 3 3  17.2  45.9      69.3   9.3
## 4 4 151.5  41.3      58.5  18.5
## 5 5 180.8  10.8      58.4  12.9
## 6 6   8.7  48.9      75.0   7.2
```

…which confirms the redundancy of *X*.

Since the ultimate goal is to understand how advertisement in *TV*, *radio* and *newspaper* explain the response (*sales*), let us first have an idea of how much has been spent in each medium. Let us check this with a simple piping which includes a new column with the total expenditure.

```
data_with_total_sum <- data_adv %>% mutate(total=TV+radio+newspaper) %>% select(TV, radio,newspaper,total)
summary(data_with_total_sum)
```

```
##        TV             radio          newspaper          total      
##  Min.   :  0.70   Min.   : 0.000   Min.   :  0.30   Min.   : 11.7  
##  1st Qu.: 74.38   1st Qu.: 9.975   1st Qu.: 12.75   1st Qu.:123.5  
##  Median :149.75   Median :22.900   Median : 25.75   Median :207.3  
##  Mean   :147.04   Mean   :23.264   Mean   : 30.55   Mean   :200.9  
##  3rd Qu.:218.82   3rd Qu.:36.525   3rd Qu.: 45.10   3rd Qu.:281.1  
##  Max.   :296.40   Max.   :49.600   Max.   :114.00   Max.   :433.6
```

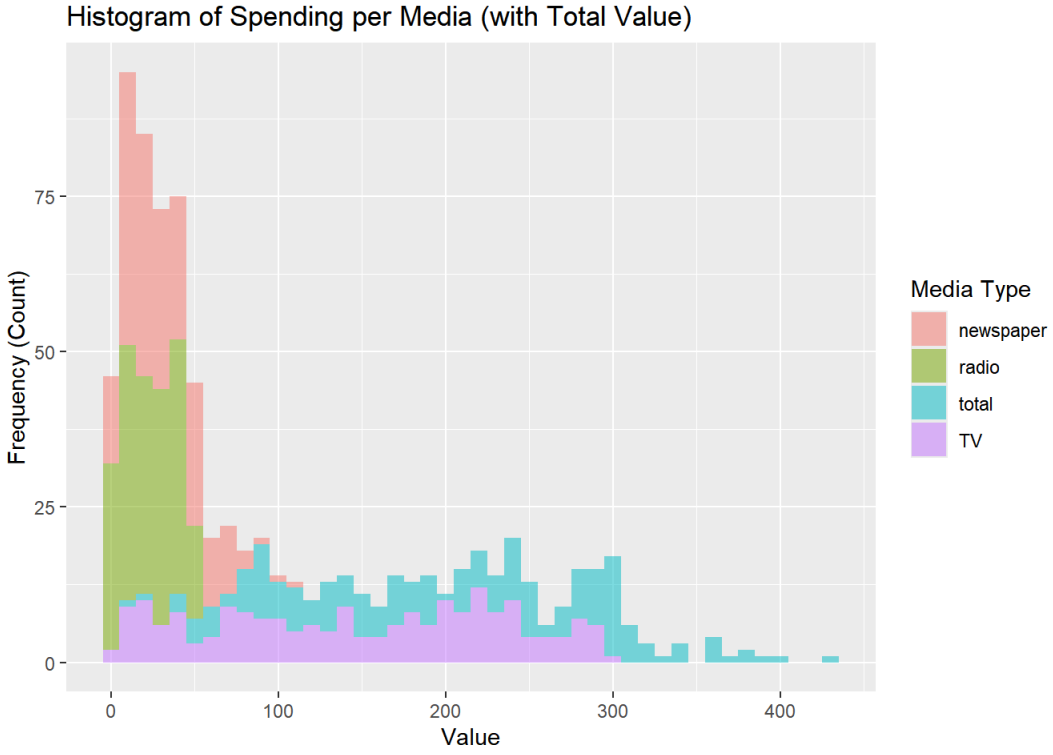
Thus, on average much more has been spent in TV than in radio and newspaper taken together! We may graphically confirm this with the appropriate plots.

First we need to reshape the data so as to have a simultaneous visualization:

```
data_long <- data_with_total_sum %>%
  pivot_longer(cols = c(TV, radio, newspaper, total),
               names_to = "media_type",
               values_to = "value")
```

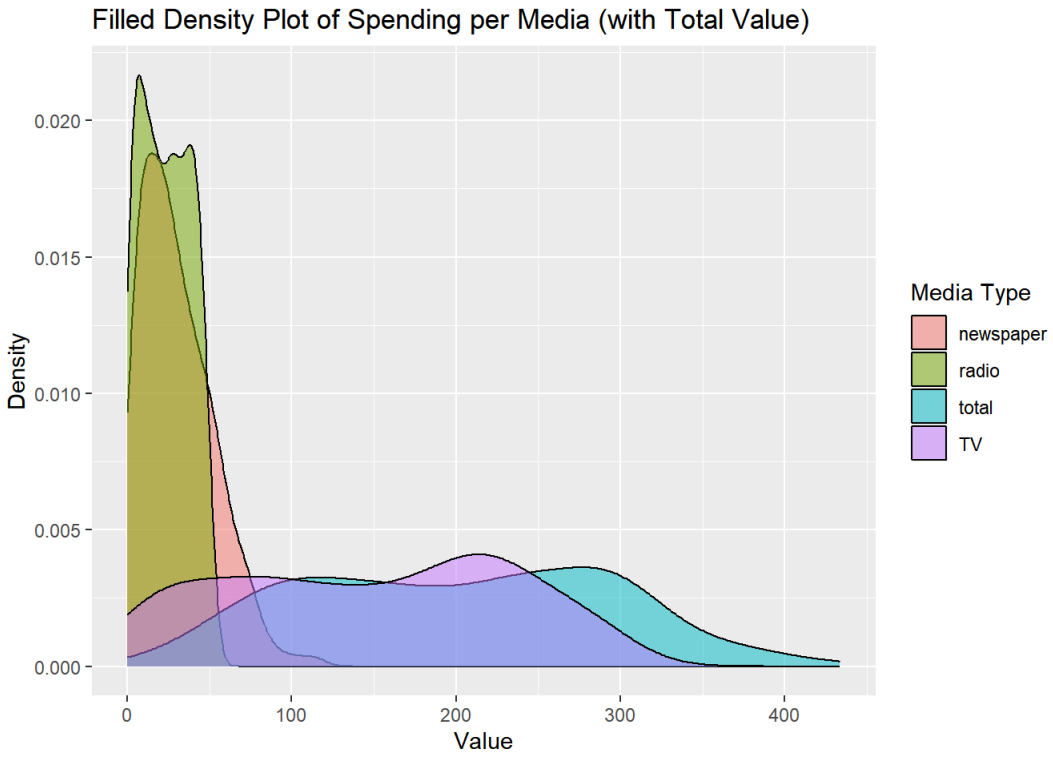
We may start with histograms…

```
ggplot(data_long, aes(x = value, fill = media_type)) +
  geom_histogram(binwidth = 10,alpha = 0.5) +
  labs(title = "Histogram of Spending per Media (with Total Value)",
       x = "Value",
       y = "Frequency (Count)",
       fill = "Media Type")
```



…and then refine this by looking at the corresponding density plots:

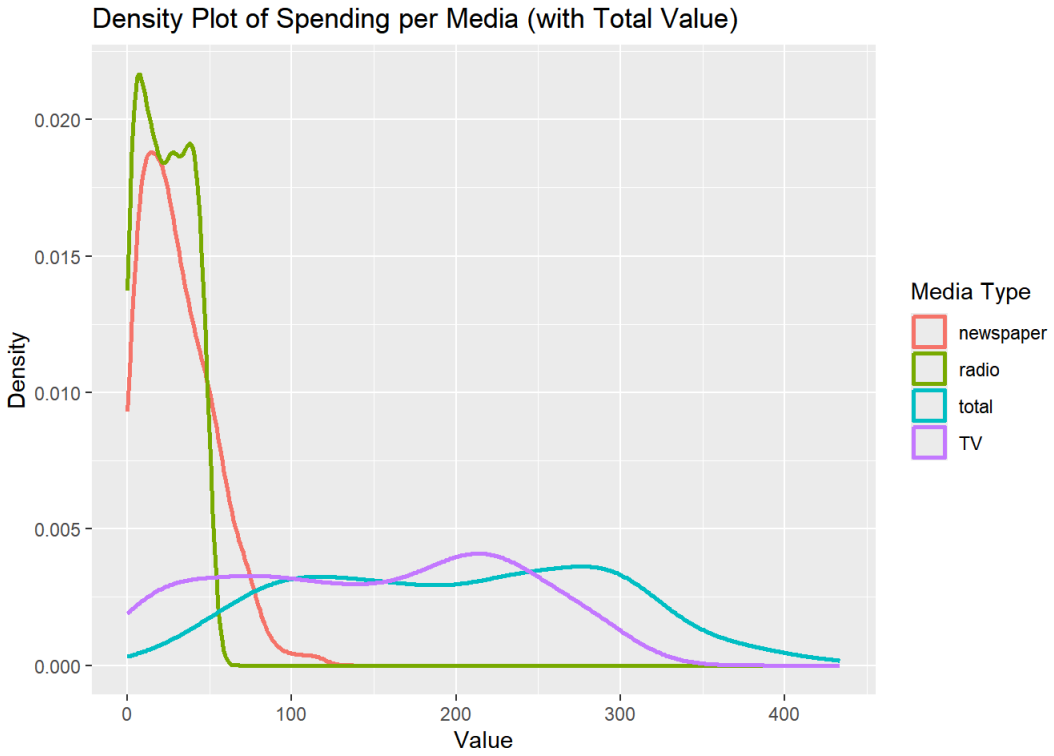
```
# Plot the long data as a density function
ggplot(data_long, aes(x = value, fill = media_type)) +
  geom_density(alpha = 0.5) +
  labs(title = "Filled Density Plot of Spending per Media (with Total Value)",
       x = "Value",
       y = "Density",
       fill = "Media Type")
```



Note that any of the four colored regions has unit area, as expected.

The same density plot with the colored fillings removed:

```
ggplot(data_long, aes(x = value, color = media_type)) +
  geom_density(linewidth = 1) + # Use linewidth to make lines thicker
  labs(title = "Density Plot of Spending per Media (with Total Value)",
       x = "Value",
       y = "Density",
       color = "Media Type")
```



Fitting a multiple regression model to the data

The preliminary analysis gives no clue as to whether the spending should be concentrated in some sub-collection of media (for instance, *TV* and *radio*). Since this delicate decision is our ultimate concern here, let us fit a multiple least squares model to the data based on the linear expression…

$$\mathbf{y}_{:\text{sales}} = \beta_0 + \beta_1 x_{1:\text{TV}} + \beta_2 x_{2:\text{radio}} + \beta_3 x_{3:\text{newspaper}} + \mathbf{e}$$

```
model_total <- lm(sales ~ TV+radio+newspaper, data=data_adv)
```

… and see what happens.

Before embarking into the analysis of the corresponding summary, we first observe that the fitting already allows us to access the residual vector $\hat{\mathbf{e}}$, which, as we have seen above, is the most important element in the least squares inference analysis (as it is the appropriate orthogonal projection of both the response $\mathbf{y}_{:\text{sales}}$ and the error $\mathbf{e}$, which are directly inaccessible).

The whole summary with the residual entries may be accessed with `residuals(model_total)` but this is useless for a human: it is a vector with *200* entries…

```
str(residuals(model_total))
```

```
##  Named num [1:200] 1.576 -1.938 -3.008 0.902 -0.289 ...
##  - attr(*, "names")= chr [1:200] "1" "2" "3" "4" ...
```

In fact, we may now access a whole array of sample information associated to the model with the same syntax.

```
names(model_total)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

For instance,…

```
print(coefficients(model_total))
```

```
##  (Intercept)           TV        radio    newspaper
##  2.938889369  0.045764645  0.188530017 -0.001037493
```

…prints the estimated coefficient vector $\beta = (\widehat{\beta}_0, \widehat{\beta}_1, \widehat{\beta}_2, \widehat{\beta}_3)$ of the fitted model (compare with the much more informative summary below).

Returning to $\hat{\mathbf{e}}$, we may appeal to a few descriptive plots in order to probe its structure. We first construct the appropriate data.frame:

```
residuals <- residuals(model_total)
res<-data.frame(residuals)
```

As a check we may summarize this data.frame…

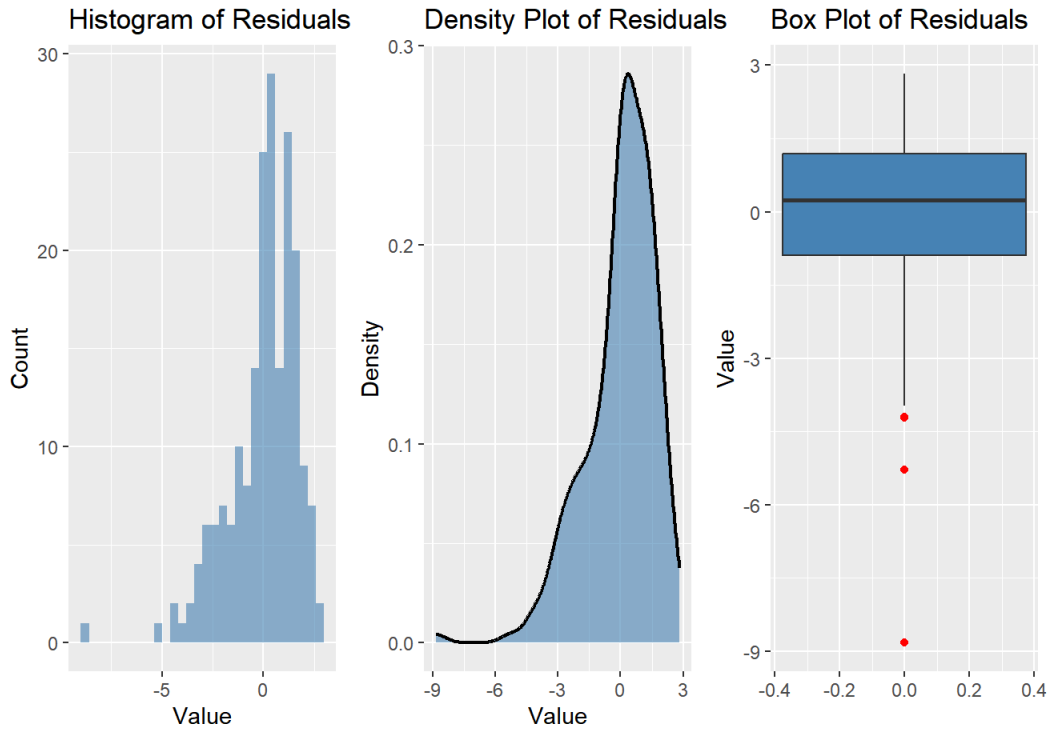
```
summary(res)
```

```
##    residuals
##  Min.   :-8.8277
##  1st Qu.:-0.8908
##  Median : 0.2418
##  Mean   : 0.0000
##  3rd Qu.: 1.1893
##  Max.   : 2.8292
```

…which furnishes a brief summary of the **residuals**. Note that the vanishing of the mean is an indication of normality and in fact theoretically expected (due to the projection property mentioned earlier). On the other hand, the median is non-zero and the first and third quantiles are far from being symmetric about the origin, which might be problematic …

Let us then pass to a graphical analysis, starting with a histogram, a density plot and a box plot of the residuals displayed side by side…

```
# Histogram
hist_plot <- ggplot(res, aes(x = residuals)) +
  geom_histogram(binwidth = .4, alpha = 0.6, fill = "steelblue") +
  labs(title = "Histogram of Residuals",
       x = "Value",
       y = "Count")
# Density Plot
density_plot <- ggplot(res, aes(x = residuals)) +
  geom_density(linewidth = 0.8, fill = "steelblue", alpha = 0.6) +
  labs(title = "Density Plot of Residuals",
       x = "Value",
       y = "Density")
#Box Plot
box_plot <- ggplot(res, aes(y = residuals)) +
  geom_boxplot(outlier.colour="red",fill="steelblue") +
  labs(title = "Box Plot of Residuals",
       x = "",
       y = "Value")
#Combine the plots side-by-side using patchwork
hist_plot + density_plot + box_plot
```



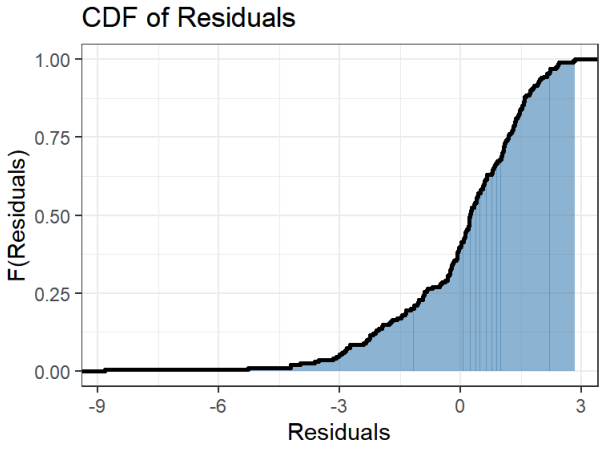
…which, taken together, resemble normality to a certain extent.

We may also compare the density plot with the related empirical **cumulative distribution function** (CDF). First, we calculate the data needed for the step plot…

```
ecdf_fun <- ecdf(res$residuals)
plot_data <- data.frame(
  x_start = knots(ecdf_fun),
  y_val = ecdf_fun(knots(ecdf_fun))
) %>%
  mutate(x_end = lead(x_start, default = max(x_start))) %>%
  filter(x_start != max(x_start))
```

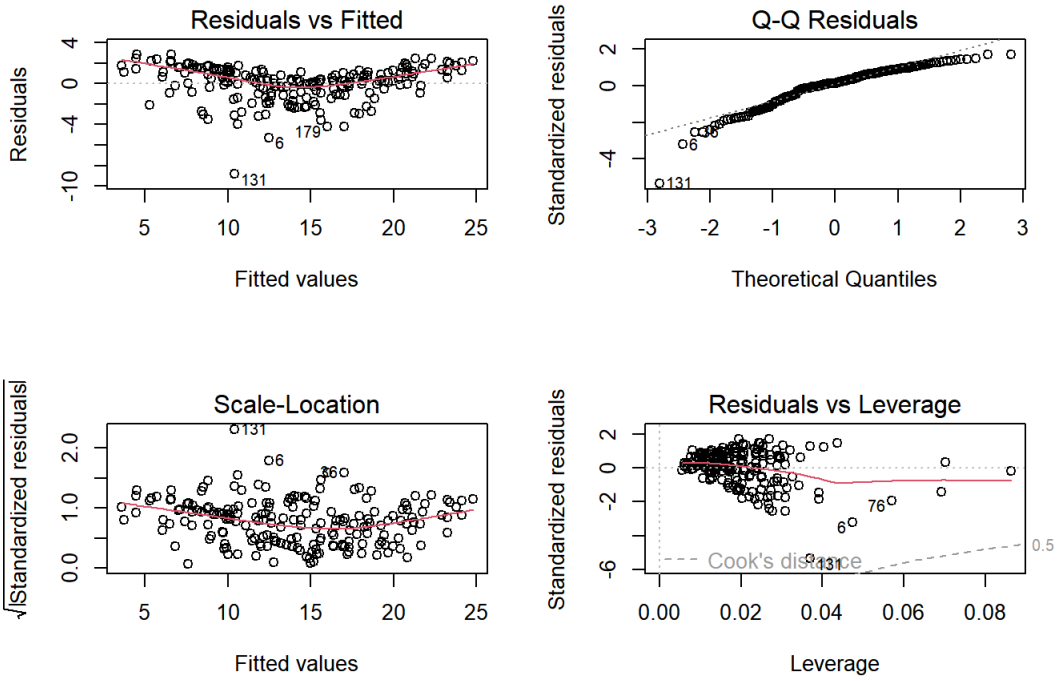
…so the CDF plot, which again resembles normality, may be obtained in the usual way.

```
ggplot() +
  geom_rect(data = plot_data,
            aes(xmin = x_start, xmax = x_end, ymin = 0, ymax = y_val),
            fill = "steelblue", alpha = 0.6) +
  geom_step(data = res, aes(x = residuals), stat = "ecdf", linewidth = 1.0) +
  labs(title="CDF of Residuals",
       y = "F(Residuals)", x="Residuals") +
  theme_bw()
```



Let us now complement this with the standard goodness-of-fit plots for testing the model assumptions beyond normality.

```
par(mfrow=c(2,2))
plot(model_total)
```



Thus, as far as linearity[1], error normality, homoscedasticity and outliers are concerned, the model has (reasonably!) passed the graphical tests.

Let us now play devil's advocate and check normality **numerically** with a well-known test available in R:

```
shapiro.test(residuals(model_total))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model_total)
## W = 0.91767, p-value = 3.939e-09
```

Thus, the rather small *p*-value indicates that the residuals normality has failed the **Shapiro-Wilk test**. We note, however, that the use of *p*-value-based tests to check normality of residuals is disputable for at least two reasons: i) the tests are not powerful enough for small samples when normality might be important; ii) they are too powerful for large samples, when normality is less important due to the Central Limit Theorem (CLT). Taking this account and noticing that the Q-Q plot above looks "almost" straight with only minor wiggles (and only near the extremities), we get

strong indication that our data are likely “normal enough.” Hence, we will stick to the graphical analysis above, and will assume that the residuals are normally distributed (which indirectly confirms the error normality of the model) for at least two reasons:

- the sample size is *n=200*, which allows us to confidently appeal to CLT;
- all the hypothesis tests used below to perform **variable selection** on the models we shall deal with are quite robust to small departures from normality.

Thus, let us proceed (with the fingers crossed!) and look at the model summary.

```
summary(model_total)
```

```
##
## Call:
## lm(formula = sales ~ TV + radio + newspaper, data = data_adv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

We may substantially improve the presentation by adding colors to the various blocks in this summary using the function *summary_with_colors()* below.

```
summary_colors <- c(
  block1 = "#e6f0ff",
  block2 = "#e6ffe6",
  block3 = "#fff5e6",
  block4 = "#f9e6ff"
)

summary_with_colors <- function(model) {
  summary_text <- capture.output(summary(model))
  call_start <- grep("^Call:", summary_text)
  residuals_start <- grep("^Residuals:", summary_text)
  coeffs_start <- grep("^Coefficients:", summary_text)
  footer_start <- grep("^Residual standard error:", summary_text)

  block1_text <- summary_text[call_start:(residuals_start - 1)]
  block2_text <- summary_text[residuals_start:(coeffs_start - 1)]
  block3_text <- summary_text[coeffs_start:(footer_start - 1)]
  block4_text <- summary_text[footer_start:length(summary_text)]

  wrap_in_div <- function(text, color) {
    text_collapsed <- paste(text, collapse = "\n")
    sprintf(
      '<div style="background-color: %s; border: 1px solid #ddd; border-radius: 5px; padding: 10px; margin-bottom: 10px;"><p
re style="margin: 0;">%s</pre></div>',
      color,
      htmltools::htmlEscape(text_collapsed)
    )
  }
```

```
  html_output <- paste(
    wrap_in_div(block1_text, summary_colors["block1"]),
    wrap_in_div(block2_text, summary_colors["block2"]),
    wrap_in_div(block3_text, summary_colors["block3"]),
    wrap_in_div(block4_text, summary_colors["block4"]),
    sep = "\n"
  )
  asis_output(html_output)
}

highlight_block_text <- function(text, block_number) {
  color <- summary_colors[block_number]
  sprintf(
    '<span style="background-color: %s; padding: 2px 5px; border-radius: 4px; font-family: monospace;">%s</span>',
    color,
    text
  )
}

print_summary_block <- function(model, block_number) {
  if (!(block_number %in% 1:4)) {
    stop("Invalid block_number. Please choose a number from 1 to 4.")
  }

summary_text <- capture.output(summary(model))
  call_start <- grep("^Call:", summary_text)
  residuals_start <- grep("^Residuals:", summary_text)
  coeffs_start <- grep("^Coefficients:", summary_text)
  footer_start <- grep("^Residual standard error:", summary_text)

  all_blocks <- list(
    "1" = summary_text[call_start:(residuals_start - 1)],
    "2" = summary_text[residuals_start:(coeffs_start - 1)],
    "3" = summary_text[coeffs_start:(footer_start - 1)],
    "4" = summary_text[footer_start:length(summary_text)]
  )

  selected_text <- all_blocks[[as.character(block_number)]]
  selected_color <- summary_colors[block_number]

  text_collapsed <- paste(selected_text, collapse = "\n")
  html_output <- sprintf(
      '<div style="background-color: %s; border: 1px solid #ddd; border-radius: 5px; padding: 10px;"><pre style="margin:
0;">%s</pre></div>',
      selected_color,
      htmltools::htmlEscape(text_collapsed)
  )
  asis_output(html_output)
}
```

```
summary_with_colors(model_total)
```

```
Call:
lm(formula = sales ~ TV + radio + newspaper, data = data_adv)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-8.8277 -0.8908  0.2418  1.1893  2.8292
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.938889   0.311908   9.422   <2e-16 ***
TV           0.045765   0.001395  32.809   <2e-16 ***
radio        0.188530   0.008611  21.893   <2e-16 ***
newspaper   -0.001037   0.005871  -0.177     0.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared:  0.8972,    Adjusted R-squared:  0.8956
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

Let us examine the various blocks of this colorful summary more closely.

- Of course, the `first block` is there just to remind us the formula used in fitting the least squares model.
- The `second block` gives a summary of the residuals which exactly matches the previous one and indicates a fairly symmetric distribution around the median (*~0.24*).
- Let us now look at the `last block` in the summary.

It first informs us the *rse*, the *residual standard error*, which is the square root of the *residual variance*[2] $\|\hat{\mathbf{e}}\|^2/196$, where $196 = 200 - 3 - 1$ counts the degrees of freedom (*df*) of the model.

```
print(df.residual(model_total))
```

```
## [1] 196
```

The *residual squared sum* $RSS := \|\hat{\mathbf{e}}\|^2$ may be extracted from the fitting as follows.

```
deviance(model_total)
```

```
## [1] 556.8253
```

Thus, the residual variance is…

```
deviance(model_total)/196
```

```
## [1] 2.840945
```

…and the rse is…

```
sqrt(deviance(model_total)/196)
```

```
## [1] 1.68551
```

…which matches the value given in the summary above.

- No essential difference between R^2 and adjusted R^2, with both assuring about 90% of explanation for the variation of the response, which is quite good!
- Finally, the rather small *p*-value for the *F*-test (against the null model, in which none of the predictors is statistically significant) shows that this null hypothesis should be rejected and at least one parameter is non-zero (with a very high significance level).

As an addendum, we may also obtain the same test via ANOVA: we first fit the null model…

```
model_null <- lm(sales ~ 1, data=data_adv)
```

…and then run the corresponding ANOVA, which compares this to *model_total*:

```
anova(model_null,model_total)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ 1
## Model 2: sales ~ TV + radio + newspaper
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    199 5417.1
## 2    196  556.8  3    4860.3 570.27 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the *F*-statistics in both cases is about the same (*F~570.3*), we are done.

- We may also directly obtain the relevant *F*-statistics in this test (*F*=570.3) by means of the textbook formula, but *anova()* clearly does the job; more on this below, when we compare nested models.
- The *F*-test in the last block above fails to provide statistical elements to decide which specific variables should be discarded, as it only detects the non-vanishing of at least one variable.

To ponder on this latter issue, we look at the summary's `third block`, in which the vanishing of each parameter estimate is tested separately. For our convenience, let us reproduce it here.

```
print_summary_block(model_total, 3)
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.938889   0.311908   9.422   <2e-16 ***
TV           0.045765   0.001395  32.809   <2e-16 ***
radio        0.188530   0.008611  21.893   <2e-16 ***
newspaper   -0.001037   0.005871  -0.177     0.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As explained elsewhere, under the corresponding null hypothesis the relevant statistics is

$$\frac{\text{Estimate}}{\text{Std. Error}} = \frac{\widehat{\beta}_i}{\widehat{\sigma}\sqrt{\mathfrak{s}_{ii}}},$$

whose observed values are displayed in the third column (under the head *t value*). Under error normality and homocesdaticity (which have been "empirically" verified above) this statistics is known to follow a $\mathfrak{t}$-distribution with 196 *df*s, so from the available *p*-values (in the last column) we see that only *newspaper* **fails** to be statistically significant for explaining the response. We may confirm this by looking at the 95% confidence intervals for the corresponding estimated parameters…

```
confint(model_total)
```

```
##                 2.5 %     97.5 %
## (Intercept)  2.32376228 3.55401646
## TV           0.04301371 0.04851558
## radio        0.17154745 0.20551259
## newspaper   -0.01261595 0.01054097
```

…which may be embedded in the previous block as its two last columns:

```
print_coeffs_with_ci <- function(model) {
  s <- summary(model)
  coeffs <- s$coefficients
  cis <- confint(model)
  get_stars <- function(p_value) {
    if (p_value < 0.001) return("***")
```

```
    if (p_value < 0.01)  return("**")
    if (p_value < 0.05)  return("*")
    if (p_value < 0.1)   return(".")
    return(" ")
  }
  output_lines <- "Coefficients:"
  col_names <- sprintf("%-12s %10s %11s %8s %10s %10s %9s %s",
                       "", "Estimate", "Std. Error", "t value", "Pr(>|t|)", "2.5 %",
                       "97.5 % ", "")
  output_lines <- c(output_lines, col_names)
  for (i in 1:nrow(coeffs)) {
    row_name <- rownames(coeffs)[i]
    line_data <- c(
      sprintf("%-12s", row_name),
      sprintf("%10.6f", coeffs[i, "Estimate"]),
      sprintf("%11.6f", coeffs[i, "Std. Error"]),
      sprintf("%8.3f",  coeffs[i, "t value"]),
      sprintf("%10s", format.pval(coeffs[i, "Pr(>|t|)"], digits = 2)),
      sprintf("%10.4f", cis[i, 1]),
      sprintf("%9.4f", cis[i, 2]),
      get_stars(coeffs[i, "Pr(>|t|)"])
    )
    output_lines <- c(output_lines, paste(line_data, collapse = " "))
  }

  output_lines <- c(output_lines, "---")
  output_lines <- c(output_lines, "Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1")
  text_collapsed <- paste(output_lines, collapse = "\n")
  color <- summary_colors["block3"]
  html_output <- sprintf(
      '<div style="background-color: %s; border: 1px solid #ddd; border-radius: 5px; padding: 10px;"><pre style="margin:
0;">%s</pre></div>',
      color,
      htmltools::htmlEscape(text_collapsed)
  )
  asis_output(html_output)
}
```

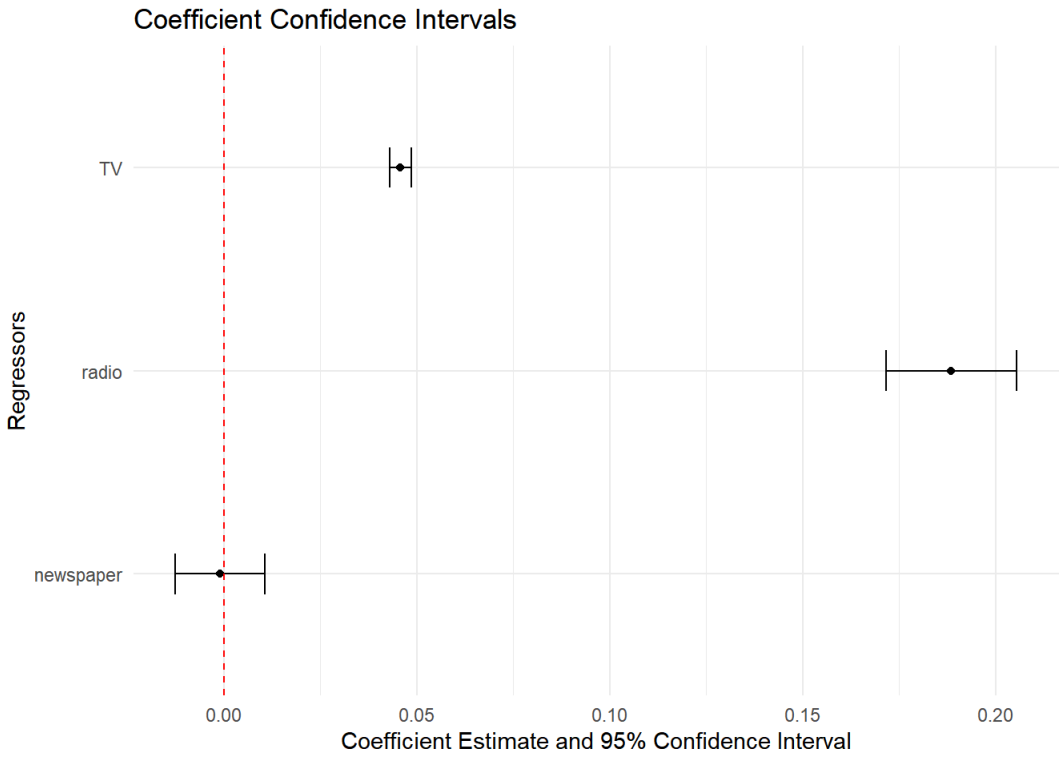
```
print_coeffs_with_ci(model_total)
```

```
Coefficients:
              Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)    2.938889    0.311908    9.422     <2e-16     2.3238    3.5540 ***
TV             0.045765    0.001395   32.809     <2e-16     0.0430    0.0485 ***
radio          0.188530    0.008611   21.893     <2e-16     0.1715    0.2055 ***
newspaper     -0.001037    0.005871   -0.177       0.86    -0.0126    0.0105
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We may also visualize the confidence intervals, with the one for the *intercept* excluded:

```
plot_coef_intervals <- function(model) {
  coefs <- as.data.frame(summary(model)$coefficients)
  conf_intervals <- confint(model)
  plot_data <- data.frame(
    term = rownames(coefs),
    estimate = coefs$Estimate,
    lower = conf_intervals[, 1],
    upper = conf_intervals[, 2]
  )

  plot_data <- plot_data[plot_data$term != "(Intercept)", ]#intercept excluded
```

```
  ggplot(plot_data, aes(x = term, y = estimate, ymin = lower, ymax = upper)) +
   geom_point() +
   geom_errorbar(width = 0.2) +
   geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
   coord_flip() +
   labs(
     title = "Coefficient Confidence Intervals",
     x = "Regressors",
     y = "Coefficient Estimate and 95% Confidence Interval"
   ) +
   theme_minimal()
}
plot_coef_intervals(model_total)
```



As expected, the only confidence interval containing 0 is the last one, which means that we may discard *newspaper* as a predictor if we wish. Besides, we note that the interval for *TV* is much closer to 0 than the one for *radio*, an information not available in the corresponding *p*-values, indicating that this latter variable is slightly more significant that the former one (this is already hinted at if we look at the respective t-statistics: *32.8* and *21.9*). In particular, we see that:

- one thousand dollars of increasing on TV advertising explains an **increasing** in sales of about *43* to *48* units, with an estimated value of *46*;
- one thousand dollars of increasing on radio advertising explains an **increasing** in sales of about *171* to *205* units, with an estimated value of *188*;
- one thousand dollars of increasing on newspaper advertising explains a variation in sales that goes from a **decreasing** of about *13* to an **increasing** of about *10* units, with an estimated net value of *1* (hence, virtually null).

Taken together, these items mean that spending in radio is about four times more efficient than spending in TV, with these variables certainly being significant for explaining sales, whereas spending in newspaper yields a practically null effect on sales. This detailed analysis illustrates the need to complement the *p*-value information with the construction of confidence intervals. Finally, although the summary confirms that it is statistically significant, in general the *intercept* β_0 is **not interpretable** and we follow this tradition here (this is the reason why its confidence interval has been excluded in the previous plot).

Following the suggestion above, we now simply discard *newspaper* and fit a model with the remaining predictors:

$$\mathbf{y}_{:\text{sales}} = \beta_0 + \beta_1 x_{1:\text{TV}} + \beta_2 x_{2:\text{radio}} + \mathbf{e}.$$

```
model_partial <- lm(sales ~ TV+radio, data=data_adv)
```

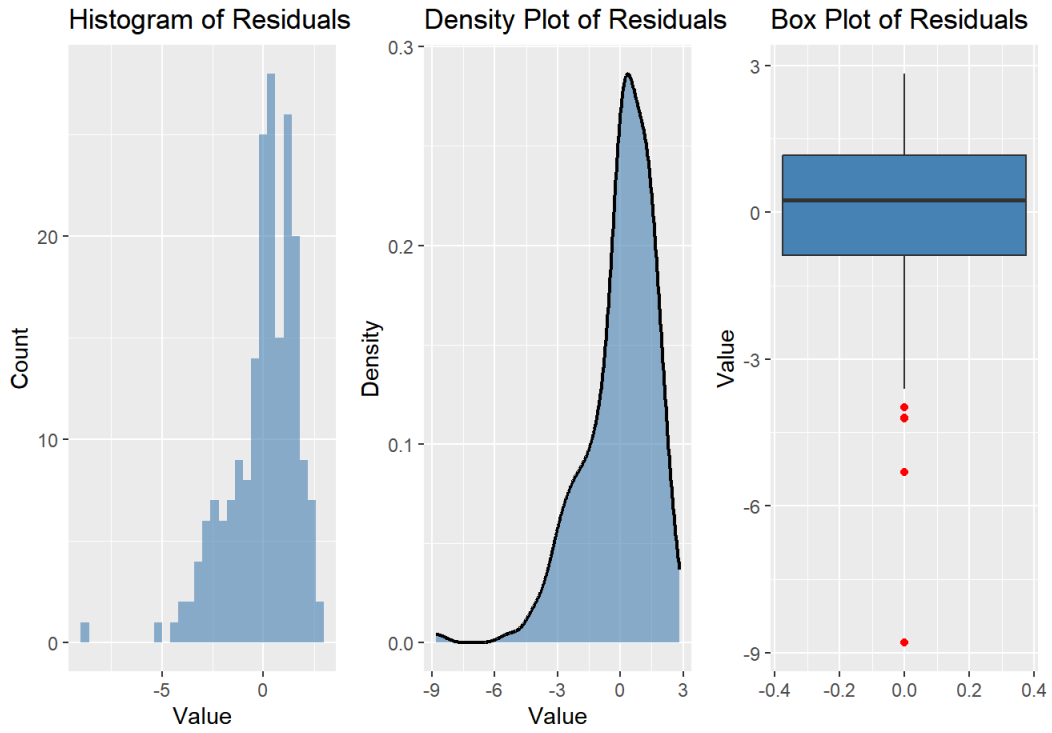
As usual, we now look at the corresponding residuals.

```
residuals_partial <- residuals(model_partial)
res_partial<-data.frame(residuals_partial)
summary(res_partial)
```

```
##  residuals_partial
##  Min.   :-8.7977
##  1st Qu.:-0.8752
##  Median : 0.2422
##  Mean   : 0.0000
##  3rd Qu.: 1.1708
##  Max.   : 2.8328
```

and the associated plots

```
#Data.frame
residuals_p <- residuals(model_partial)
res_p<-data.frame(residuals_p)
#Histogram
hist_plot_p <- ggplot(res_p, aes(x = residuals_p)) +
  geom_histogram(binwidth = .4, alpha = 0.6, fill = "steelblue") +
  labs(title = "Histogram of Residuals",
       x = "Value",
       y = "Count")
#Density Plot
density_plot_p <- ggplot(res_p, aes(x = residuals_p)) +
  geom_density(linewidth = 0.8, fill = "steelblue", alpha = 0.6) +
  labs(title = "Density Plot of Residuals",
       x = "Value",
       y = "Density")
#Box Plot
box_plot_p <- ggplot(res_p, aes(y = residuals_p)) +
  geom_boxplot(outlier.colour="red",fill="steelblue") +
  labs(title = "Box Plot of Residuals",
       x = "",
       y = "Value")
#Combine the plots side-by-side using patchwork
hist_plot_p + density_plot_p + box_plot_p
```



Let us also look at the model summary…

```
summary_with_colors(model_partial)
```

```
Call:
lm(formula = sales ~ TV + radio, data = data_adv)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-8.7977 -0.8752  0.2422  1.1708  2.8328 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.92110    0.29449   9.919   <2e-16 ***
TV           0.04575    0.00139  32.909   <2e-16 ***
radio        0.18799    0.00804  23.382   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 1.681 on 197 degrees of freedom
Multiple R-squared:  0.8972,    Adjusted R-squared:  0.8962 
F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

…with the third block adjusted to contain the *95%* confidence intervals…

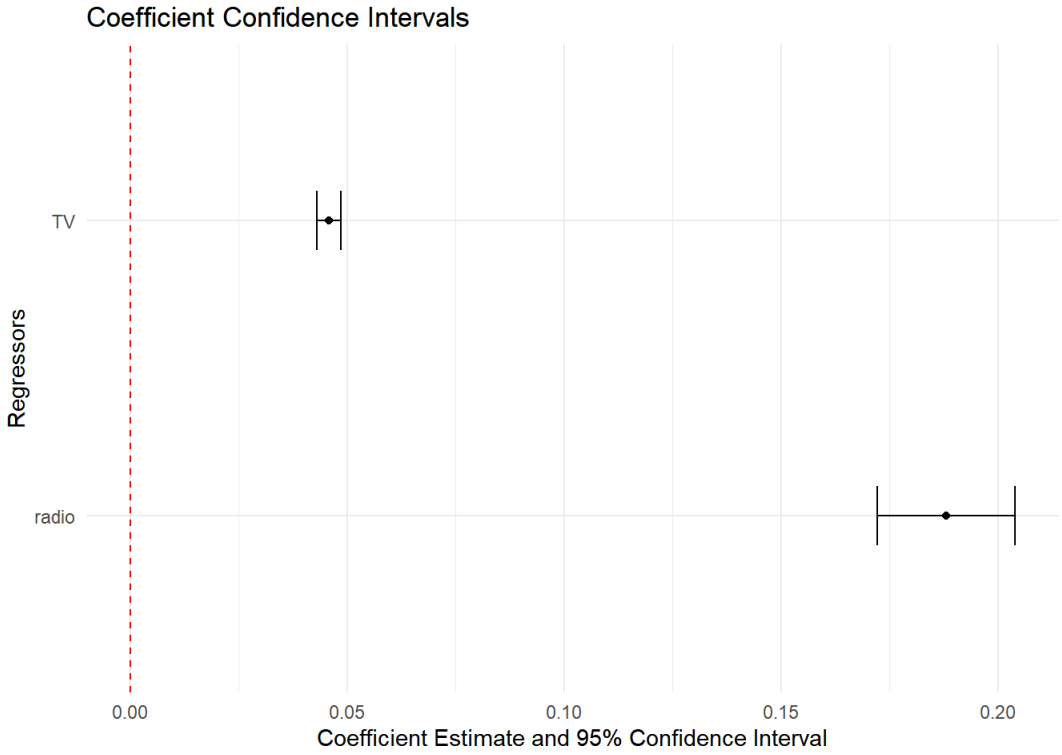
```
print_coeffs_with_ci(model_partial)
```

```
Coefficients:
              Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)    2.921100    0.294490    9.919     <2e-16     2.3403    3.5019 ***
```

```
TV            0.045755    0.001390   32.909     <2e-16     0.0430    0.0485 ***
radio         0.187994    0.008040   23.382     <2e-16     0.1721    0.2038 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…and the corresponding plots:

```
plot_coef_intervals(model_partial)
```



Now we see that:

- one thousand dollars of increasing on TV advertising explains an **increasing** in sales of about *43* to *48* units, with an estimated value of *46*;
- one thousand dollars of increasing on radio advertising explains an **increasing** in sales of about *172* to *204* units, with an estimated value of *188*;

Thus, no essential difference from the previous model (with *newspaper* included).

We may check this using ANOVA to compare the models (*model_partial* **nested** inside *model_total*).

```
anova(model_partial,model_total)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ TV + radio
## Model 2: sales ~ TV + radio + newspaper
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    197 556.91
## 2    196 556.83  1  0.088717 0.0312 0.8599
```

Noteworthy mentioning that the textbook formula for the *F*-statistics *F=0.0312* is in general given by

$$\frac{(RSS_{\text{Partial}} - RSS_{\text{Total}})/(p-q)}{RSS_{\text{Total}}/(n-p-1)} \sim F_{p-q,n-p-1} \quad \text{under the null,}$$

where $q < p$ is the number of remaining variables in the partial/null model. In our case, this reduces to…

```
{deviance(model_partial)-deviance(model_total)/{3-2}}/{deviance(model_total)/
```

```
    {200-3-1}}
```

```
## [1] 0.03122805
```

…from which the associated *p*-value may be obtained in the standard way:

```
p_value <- pf(0.03122805, 1, 196, lower.tail = FALSE)
print(p_value)
```

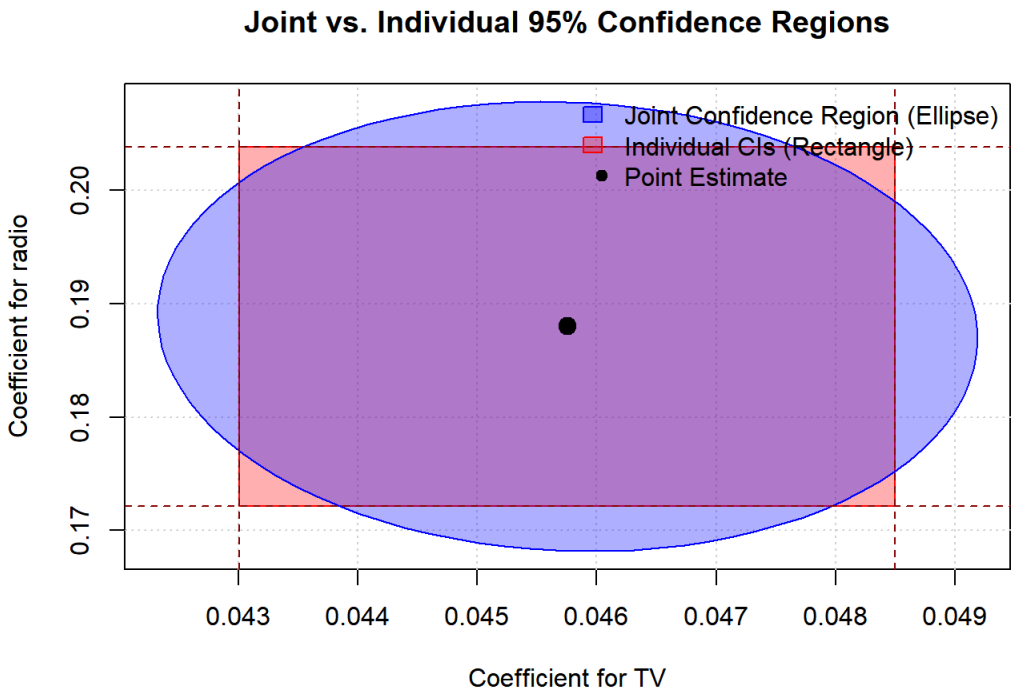
```
## [1] 0.859915
```

The large *p*-value (*~0.86*), obtained by either method, confirms that the null model (*model_partial*) should **not** be rejected, so we may safely discard *newspaper* from our original model.

As a final checking of *model_partial*, let us draw the 95% confidence ellipsis for the whole vector parameter *(TV,radio)*, with the central dot representing the estimate and the dashed lines indicating the end-points of the individual confidence intervals. For this we will use the `ellipse` package available in R, so the code below just tells how to juxtapose to the output of this package the rectangle corresponding to the confidence intervals.

```
params_to_plot <- c(2, 3)
conf_level <- 0.95

ellipse_coords <- ellipse(model_partial, which = params_to_plot, level = conf_level)

conf_intervals <- confint(model_partial, level = conf_level)
rect_x_bounds <- conf_intervals[params_to_plot[1], ]
rect_y_bounds <- conf_intervals[params_to_plot[2], ]

point_estimate <- coef(model_partial)[params_to_plot]

param_names <- names(point_estimate)

# Star plotting
plot(0, type = 'n',
     xlim = range(ellipse_coords[, 1], rect_x_bounds),
     ylim = range(ellipse_coords[, 2], rect_y_bounds),
     xlab = paste("Coefficient for", param_names[1]),
     ylab = paste("Coefficient for", param_names[2]),
     main = "Joint vs. Individual 95% Confidence Regions")

grid()

rect(xleft = rect_x_bounds[1],
     ybottom = rect_y_bounds[1],
     xright = rect_x_bounds[2],
     ytop = rect_y_bounds[2],
     col = rgb(1, 0, 0, alpha = 0.3),
     border = "red")

#alpha stands for semi-transparence
polygon(ellipse_coords,
        col = rgb(0, 0, 1, alpha = 0.3),
        border = "blue")

abline(v = rect_x_bounds, lty = 2, col = "darkred")
abline(h = rect_y_bounds, lty = 2, col = "darkred")

points(point_estimate[1], point_estimate[2],
       pch = 19,
       cex = 1.5,
       col = "black")

# Add a legend
```

```
legend("topright",
       legend = c("Joint Confidence Region (Ellipse)", "Individual CIs (Rectangle)", "Point Estimate"),
       fill = c(rgb(0, 0, 1, alpha = 0.3), rgb(1, 0, 0, alpha = 0.3), NA),
       border = c("blue", "red", NA),
       pch = c(NA, NA, 19),
       col = c(NA, NA, "black"),
       bty = "n")
```



Since the ellipse does **not** enclose the origin, the whole vector estimate should be considered statistically significant. Also, the axes of the ellipse are perfectly aligned with the coordinate axes, which indicates that the coefficient estimates are virtually uncorrelated. To directly check this alignment, we first recall that `ellipse` determines this random ellipsoidal region by means of the general expression

$$\mathcal{E}_{n,p,\delta}(\widehat{\beta};\widehat{\sigma}^2) = \left\{\beta' \in \mathbb{R}^{p+1}; (\widehat{\beta}-\beta')^t \mathfrak{s} (\widehat{\beta}-\beta') \leq (p+1)\widehat{\sigma}^2 \mathrm{f}_{p+1,n-p-1,\delta}\right\}.$$

On the other hand, the estimated covariance matrix $\mathrm{cov}(\widehat{\beta}) \approx \widehat{\sigma}^2 \mathfrak{s}$ is stored in `model_partial` as…

```
cov_matrix <- vcov(model_partial)
```

…so if we pass to the sub-matrix that really matters…

```
vars_of_interest <- c("TV", "radio")
sub_cov_matrix <- cov_matrix[vars_of_interest, vars_of_interest]
print(sub_cov_matrix)
```

```
##                 TV         radio
## TV     1.933089e-06 -6.126744e-07
## radio -6.126744e-07  6.464116e-05
```

…and use it to compute the tilting angle θ of our ellipse by means of the well-known formula…

```
theta <- atan(2*sub_cov_matrix[1,2]/(sub_cov_matrix[1,1]-sub_cov_matrix[2,2]))/2
print(theta)
```

```
## [1] 0.009769021
```

…we end up with a quite negligible value indeed.

Let us proceed with the analysis and construct **confidence** and **prediction** intervals (for the **mean response** and the **response**, respectively) of *model_partial*. First, let us estimate the **mean response**

$$\mathbb{E}\left(\mathbf{y}_{:\text{sales}}|_{(\mathbf{X}_{1:\text{TV}},\mathbf{X}_{2:\text{radio}})=(TV,radio)}\right),$$

which is the expected value of *sales* restricted to the sub-population defined by a given value of *(TV,radio)*. Thus, we are supposed to enter with the values of the pair *(TV,radio)* in a *function()* whose output should be the fitted value for *sales* and the endpoins of the corresponding confidence interval.

```
int_conf_partial <- function(new_TV, new_radio, model, conf_level = 0.95) {
  # The predict() function requires the new data to be in a data frame.
  # The column name ('TV') MUST match the predictor name in the model's formula.
  new_data_point <- data.frame(TV = new_TV, radio = new_radio)
  # Use predict() to calculate the confidence interval
  confidence_interval <- predict(
    model,
    newdata = new_data_point,
    interval = "confidence",
    level = conf_level
  )
return(confidence_interval)
}
```

…so that evaluating *int_conf_partial()* at a few values of *(TV,radio)* gives

```
int_conf_partial(new_TV = 350, new_radio = 0, model = model_partial)
```

```
##        fit      lwr      upr
## 1 18.93529 18.21195 19.65862
```

```
int_conf_partial(new_TV = 340, new_radio = 10, model = model_partial)
```

```
##        fit      lwr      upr
## 1 20.35768 19.73213 20.98323
```

```
int_conf_partial(new_TV = 290, new_radio = 60, model = model_partial)
```

```
##        fit      lwr      upr
## 1 27.46965 26.74656 28.19274
```

```
int_conf_partial(new_TV = 280, new_radio = 70, model = model_partial)
```

```
##        fit      lwr     upr
## 1 28.89204 28.05099 29.7331
```

```
int_conf_partial(new_TV = 10, new_radio = 340, model = model_partial)
```

```
##        fit      lwr      upr
## 1 67.29669 62.23472 72.35865
```

```
int_conf_partial(new_TV = 0, new_radio = 350, model = model_partial)
```

```
##        fit      lwr      upr
## 1 68.71908 63.49562 73.94254
```

We may conveniently arrange this in a table with as many multiple new data points as we wish. The idea is to inform the list of regressors in a data.frame (*new_data*) which appears as a parameter in a new function.

```
get_confidence_table <- function(new_data, model, conf_level = 0.95) {
  # This is to check if new_data is a data frame
  if (!is.data.frame(new_data)) {
    stop("Error: 'new_data' must be a data frame.")
  }

  confidence_intervals <- predict(
    model,
    newdata = new_data,
    interval = "confidence",
    level = conf_level
  )
# Combine and return
  result_table <- cbind(new_data, confidence_intervals)
return(result_table)
}
```

The function asks for a concrete data.frame, which we may provide by hand. After that, printing the table is straightforward.

```
my_new_points <- data.frame(
  TV =    c(350, 340 , 290, 280, 10, 0),
  radio = c(0,  10,  60,  70, 340, 350)
)
results_table <- get_confidence_table(new_data = my_new_points, model = model_partial)
print(results_table)
```

```
##    TV radio      fit      lwr      upr
## 1 350     0 18.93529 18.21195 19.65862
## 2 340    10 20.35768 19.73213 20.98323
## 3 290    60 27.46965 26.74656 28.19274
## 4 280    70 28.89204 28.05099 29.73310
## 5  10   340 67.29669 62.23472 72.35865
## 6   0   350 68.71908 63.49562 73.94254
```

The same table may be obtained if we prescribe instead the total spending (*new_spend*) and the spending in one of the media, say TV, so that the spending in the other medium, say radio, is the difference between these values. Of course, this is just a matter of redefining the data.frame and rewriting the code accordingly (but still using the same function as before!).

```
new_spend <- 350
TV <- c(350, 340 , 290, 280, 10, 0)
radio <- new_spend - TV
my_new_points_s <- data.frame(
  TV     ,
  radio
)
results_table_s <- get_confidence_table(new_data = my_new_points_s, model=model_partial)
print(results_table_s)
```

```
##    TV radio      fit      lwr      upr
## 1 350     0 18.93529 18.21195 19.65862
## 2 340    10 20.35768 19.73213 20.98323
## 3 290    60 27.46965 26.74656 28.19274
## 4 280    70 28.89204 28.05099 29.73310
## 5  10   340 67.29669 62.23472 72.35865
## 6   0   350 68.71908 63.49562 73.94254
```

…which matches the previous table.

Looking at the extreme values in this table we see that, with an overall expenditure of *350*, we have that:

- an expenditure of *350* (in thousands of dollars) in TV leads to an average sale of about *19.000* units;
- an expenditure of *0* in TV leads to a average sale of about *69.000* units;

How should we interpret this rather strange outcome? Does it mean that we should concentrate the spending in radio and simply forget about TV?

The problem here seems to be that in general it may be **too risky** to make estimates for the mean response for a range of values of the regressors lying very far from those covered by data (this is usually known as **extrapolation**). To check this, let us summarize the spending associated to *model_partial*, thus completely ignoring the spending with *newspaper* but including *sales*.

```
data_partial_sum <- data_adv %>% mutate(partial_sum=TV+radio) %>% select(sales, TV, radio,partial_sum)
summary(data_partial_sum)
```

```
##      sales             TV             radio         partial_sum
##  Min.   : 1.60   Min.   :  0.70   Min.   : 0.000   Min.   : 10.70
##  1st Qu.:10.38   1st Qu.: 74.38   1st Qu.: 9.975   1st Qu.: 90.62
##  Median :12.90   Median :149.75   Median :22.900   Median :175.55
##  Mean   :14.02   Mean   :147.04   Mean   :23.264   Mean   :170.31
##  3rd Qu.:17.40   3rd Qu.:218.82   3rd Qu.:36.525   3rd Qu.:243.30
##  Max.   :27.00   Max.   :296.40   Max.   :49.600   Max.   :332.70
```

We thus see that the extreme choices above correspond to values of *(TV,radio)* very far from the mean and the median provided by the data. Thus, a more realistic, data-driven estimate would be to restrict ourselves to **interpolation** (where we remain within the range covered by data) by choosing values around *(TV,radio)=(150,25)*,

```
int_conf_partial(new_TV = 150, new_radio = 25 , model = model_partial)
```

```
##        fit      lwr      upr
## 1 14.48418 14.24802 14.72034
```

which gives about *14.500* sales (on average) and remains quite close to the overall sales mean.

By keeping the total spending as *175*, we may try small variations around *TV=150*:

```
new_spend <- 175
TV <- c(140, 145 , 150, 155, 160)
radio <- new_spend - TV
my_new_points_s_n <- data.frame(TV,radio)
results_table_s_n <- get_confidence_table(new_data = my_new_points_s_n, model=model_partial)
print(results_table_s_n)
```

```
##    TV radio      fit      lwr      upr
## 1 140    35 15.90657 15.60597 16.20718
## 2 145    30 15.19537 14.93755 15.45320
## 3 150    25 14.48418 14.24802 14.72034
## 4 155    20 13.77298 13.53163 14.01433
## 5 160    15 13.06178 12.78991 13.33365
```

which turns out to be a quite satisfactory output (in the sense that the confidence intervals are much tighter).

Now let us modify the code above to obtain the corresponding prediction intervals, which estimate the response itself:

$$\mathbf{y}_{:\text{sales}} \Big| (\mathbf{x}_{1:\text{TV}}, \mathbf{x}_{2:\text{radio}}) = (TV, radio)$$

```
int_pred_partial <- function(new_TV, new_radio, model, conf_level = 0.95) {

  new_data_point <- data.frame(TV = new_TV, radio = new_radio)

  prediction_interval <- predict(
    model,
    newdata = new_data_point,
    interval = "prediction",
    level = conf_level
  )
```

```
  return(prediction_interval)
}
```

…so that using our realistic choice above would yield the prediction interval…

```
int_pred_partial(new_TV = 150, new_radio = 25, model = model_partial)
```

```
##        fit   lwr      upr
## 1 14.48418 11.16 17.80835
```

…which is much wider than the confidence interval for the mean response.

Of course, we may also provide a table of predicted values around *(TV,radio)=(150,25)* by slightly modifying the codes above, but this is left as an exercise.

A **summary** of the analysis so far:

- We started with *model_total*, a least squares model in which *sales* is supposedly explained by advertisements in *TV*, *radio* and *newspaper*, and which passed the usual tests of linearity, normality, homoscedasticity and outliers;
- We have seen, however, that *newspaper* fails to significantly explain *sales*, so we discarded it and considered *model_partial*, which uses only *TV* and *radio* as regressors;
- The models have been compared to one another via a nested *F*-test, with a large *p*-value justifying the reduction to the smaller model;
- Thus, we discarded *model_total*, considered only *model_partial* and proceeded to examine its explanatory/predictive power.

In fact, the procedure above constitutes the first step in applying **variable selection** by means of **backward selection**, an algorithm which can be organized as follows:

- Start with the full model (in our case, *model_total*), thus incorporating all the regressors in the data (in our case, *TV*, *radio* and *newspaper*);
- Run a least squares fitting for this full model and then discard the **least** significant regressor, namely, that associated to the parameter displaying the largest *p*-value in the corresponding t-test (in our case, *newspaper*). Preferably, this should be accompanied by an examination of the confidence intervals for the involved parameters in order to ensure that the chosen regressor has indeed the largest *p*-value in a statistically significant way;
- Fit the reduced model (in our case, *model_partial*) and iterate the previous step by removing one regressor at a time up to the point in which some pre-assigned **stopping rule** is reached (usually, when all *p*-values are strictly less than an acceptable significance level, say *0.05*).

Let us then extract the *p*-values of the nested models considered so far.

The can be done either separately for each model…

```
p.values.t <- function(model) {
  summary(model)$coefficients[, "Pr(>|t|)"]
}
p.values.t(model_total)
```

```
##  (Intercept)           TV        radio    newspaper
## 1.267295e-17 1.509960e-81 1.505339e-54 8.599151e-01
```

```
p.values.t(model_partial)
```

```
##  (Intercept)           TV        radio
## 4.565557e-19 5.436980e-82 9.776972e-59
```

… or in a more informative manner, by arranging the *p*-values in a table with colors demarcating the nested models:

```
model_list <- list(full_model = model_total, reduced_model = model_partial)
#
extract_pvals_simple <- function(model) {
  coef_table <- summary(model)$coefficients
 df <- data.frame(
  Regressor = rownames(coef_table),
    P_Value = coef_table[, "Pr(>|t|)"]
```

```
  )
rownames(df) <- NULL
return(df)
}
#
final_table <- map_dfr(model_list, extract_pvals_simple, .id = "Model")
#
final_table %>%
  # Conditionally format the P-Value column
  mutate(P_Value_Formatted = ifelse(P_Value < 0.001,
                                    sprintf("%.3e", P_Value),
                                    sprintf("%.4f", P_Value))) %>%
  select(Model, Regressor, P_Value_Formatted) %>%
  kable(format = "html",
        caption = "Model Comparison with P-Values",
        col.names = c("Model", "Regressor", "P-Value")) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  row_spec(which(final_table$Model == "full_model"),
           background = "#D9E3F4") %>%
  row_spec(which(final_table$Model == "reduced_model"),
           background = "#F4EAD9")
```

Model Comparison with P-Values

Model	Regressor	P-Value
full_model	(Intercept)	1.267e-17
full_model	TV	1.510e-81
full_model	radio	1.505e-54
full_model	newspaper	0.8599
reduced_model	(Intercept)	4.566e-19
reduced_model	TV	5.437e-82
reduced_model	radio	9.777e-59

This indeed shows that:

- removing *newspaper* from the full model (*model_total*) is justified (in fact, it is the only regressor whose coefficient has a *p*-value larger than *0.05*);
- the reduced model (*model_partial*) already meets the stopping rule (as all of its coefficients already have associated *p*-values less than *0.05*)

Thus, as far as **backward selection** is concerned, we must adopt *model_partial* as our final **linear** model and explore further its explanatory/predictive power (in alignment with what we already did above).

The effect of higher order interactions

The analysis above only takes into account the presence of **linear** terms in the regression model. And how about adding **higher order terms** in the main regressors (*TV* and *radio*) to the linear model? Here we examine this by starting with

$$\mathbf{y}_{:\text{sales}} = \beta_0 + \beta_1 x_{1:\text{TV}} + \beta_2 x_{2:\text{radio}} + \beta_{12} x_{12:\text{TV}\times\text{radio}} + \mathbf{e},$$

which incorporates the interaction regressor $x_{12} = x_1 x_2$ to *model_partial*.

We may run this in at least two different ways. First, we may manually construct a new data.frame using…

```
data_partial_int <- data_adv %>% mutate(int=TV*radio) %>% select(sales, TV, radio,int)
head(data_partial_int)
```

```
##   sales    TV radio     int
## 1  22.1 230.1  37.8 8697.78
## 2  10.4  44.5  39.3 1748.85
## 3   9.3  17.2  45.9  789.48
```

```
## 4  18.5 151.5  41.3 6256.95
## 5  12.9 180.8  10.8 1952.64
## 6   7.2   8.7  48.9  425.43
```

…and then fit the expanded model with the interaction variable *int* included as an independent regressor…

```
data_partial_int_1 <- lm(sales ~ TV + radio + int, data=data_partial_int)
print_coeffs_with_ci(data_partial_int_1)
```

```
Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)  6.750220    0.247871   27.233     <2e-16     6.2614    7.2391 ***
TV           0.019101    0.001504   12.699     <2e-16     0.0161    0.0221 ***
radio        0.028860    0.008905    3.241     0.0014     0.0113    0.0464 **
int          0.001086    0.000052   20.727     <2e-16     0.0010    0.0012 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…or simply use…

```
model_partial_int_2 <- lm(sales ~ TV*radio, data = data_adv)
# Here, 'TV*radio' is a shorthand for "TV + radio + TV:radio", thus including the first order terms already present in the l
inear model
print_coeffs_with_ci(model_partial_int_2)
```

```
Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)  6.750220    0.247871   27.233     <2e-16     6.2614    7.2391 ***
TV           0.019101    0.001504   12.699     <2e-16     0.0161    0.0221 ***
radio        0.028860    0.008905    3.241     0.0014     0.0113    0.0464 **
TV:radio     0.001086    0.000052   20.727     <2e-16     0.0010    0.0012 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…which gives the same results. Note, however, that by comparing this with the summary of *model_partial*, the strictly linear model,…

```
print_coeffs_with_ci(model_partial)
```

```
Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)  2.921100    0.294490    9.919     <2e-16     2.3403    3.5019 ***
TV           0.045755    0.001390   32.909     <2e-16     0.0430    0.0485 ***
radio        0.187994    0.008040   23.382     <2e-16     0.1721    0.2038 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…we see that the estimated coefficients for the main variables (*TV* and *radio*) have changed, which anticipates difficulties in interpreting this expanded model. Let us simply ignore this for the moment and fit the full model incorporating all possible (polynomial) interactions up to second order:

$$\mathbf{y}_{:\text{sales}} = \beta_0 + \beta_1 x_{1:\text{TV}} + \beta_2 x_{2:\text{radio}} + \beta_{12} x_{12:\text{TV}\times\text{radio}} + \beta_{11} x_{11:\text{TV}\times\text{TV}} + \beta_{22} x_{22:\text{radio}\times\text{radio}} + \mathbf{e}.$$

```
model_full_raw <- lm(sales ~ TV*radio + I(TV^2) + I(radio^2), data = data_adv)
summary_with_colors(model_full_raw)
```

```
Call:
lm(formula = sales ~ TV * radio + I(TV^2) + I(radio^2), data = data_adv)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.0027 -0.2859 -0.0062  0.3829  1.2100
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.194e+00  2.061e-01  25.202   <2e-16 ***
TV           5.099e-02  2.236e-03  22.801   <2e-16 ***
radio        2.654e-02  1.242e-02   2.136   0.0339 *
I(TV^2)     -1.098e-04  6.901e-06 -15.914   <2e-16 ***
I(radio^2)   1.861e-04  2.359e-04   0.789   0.4311
TV:radio     1.075e-03  3.479e-05  30.892   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6244 on 194 degrees of freedom
Multiple R-squared:  0.986, Adjusted R-squared:  0.9857
F-statistic:  2740 on 5 and 194 DF,  p-value: < 2.2e-16
```

We immediately see that, under the usual *95%* threshold, $x_{22:\text{radio}\times\text{radio}}$ fails to be significant (*p*-value *0.43*) whereas $x_{1:\text{radio}}$ came close to it (*p*-value *0.03*), but this might be merely an effect of correlations introduced in this full model due to the addition of the interaction terms. In this regard, we should have in mind that a **high correlation** between predictors can cause several problems in a regression model such as:

- **Unstable Coefficient Estimates:** The estimated coefficients for a main predictor, hereafter denoted by *main*, and its square *main^2*, which are obviously **correlated** to a high degree, can be very sensitive to small changes in the data;
- **Inflated Standard Errors:** Perhaps more importantly, the standard errors for the correlated predictors become large, which makes the confidence intervals for the coefficients very wide, so that some variable may be displayed as **not** statistically significant when it actually has a strong effect in explaining the model;
- **Difficulty in Interpretation:** The coefficients for *main* are supposed to represent the effect of a one-unit increase in *main* while holding all the remaining coefficients, in particular *main*^2, constant, which is at least logically awkward.

We may remedy this by using the *poly()* function, which, by default, doesn't just calculate *main* and *main^2* but instead creates a new set of variables that are orthogonal (i.e., perfectly uncorrelated) to each other and still retains the linear and quadratic information from the original *main* variables. This solves the problems mentioned above, as the new predictors are uncorrelated by design:

- **Stable and Independent Coefficients:** Because the terms are uncorrelated, the coefficient estimated for the linear component is independent of the coefficient estimated for the quadratic component. In particular, adding the quadratic term to the model will not change the coefficient of the linear term, which is a huge advantage for model building, as it allows us to test the contribution of each polynomial degree in a clean way;
- **Clearer Hypothesis Testing:** We can confidently use the *p*-value of the quadratic term to see if it significantly improves the model, without its statistical significance being obscured by its high correlation with the linear term.

With this theoretical discussion out of the way, we may run the suggested procedure as follows.

```
model_poly_orth <- lm(sales ~ TV:radio + poly(TV, 2) + poly(radio, 2), data = data_adv)
summary_with_colors(model_poly_orth)
```

```
Call:
lm(formula = sales ~ TV:radio + poly(TV, 2) + poly(radio, 2),
    data = data_adv)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.0027 -0.2859 -0.0062  0.3829  1.2100
```

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.027e+01  1.292e-01  79.508  < 2e-16 ***
poly(TV, 2)1     2.343e+01  1.208e+00  19.396  < 2e-16 ***
poly(TV, 2)2    -9.990e+00  6.278e-01 -15.914  < 2e-16 ***
poly(radio, 2)1  7.426e+00  1.239e+00   5.991 9.96e-09 ***
poly(radio, 2)2  4.939e-01  6.261e-01   0.789    0.431
TV:radio         1.075e-03  3.479e-05  30.892  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6244 on 194 degrees of freedom
Multiple R-squared:  0.986, Adjusted R-squared:  0.9857
F-statistic:  2740 on 5 and 194 DF,  p-value: < 2.2e-16
```

We should be aware, however, that using orthogonal polynomials changes the interpretation of the individual coefficients, as they no longer correspond to the likes of *main* and *main^2*. On the other hand, as it is clearly seen in the summaries above, the overall model fit (*R*-squared, *F*-statistic) and the fitted values of *model_poly_orth* are identical to their counterparts in *model_full_raw*, with its raw, collinear terms. We find it convenient to append a couple of brief remarks justifying these claims on theoretical grounds:

- when passing from one model to the other, the design matrix is right multiplied by a non-singular *(p+1)x(p+1)* matrix M, precisely $\mathfrak{r}_{\text{ortho}} = \mathfrak{r}_{\text{raw}} M$, and since the hat matrix of a model turns out to be insensitive to this kind of linear transformation, we conclude that both $\widehat{\mathbf{y}}$ and $\hat{\mathbf{e}}$ remain unchanged (which is nice for **goodness of fit**).
- since the vector of regressors $\mathrm{x} = (1, \mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_{12}, \mathrm{x}_{11}, \mathrm{x}_{22})$ used in doing predictions and the vector of estimated coefficients $\widehat{\beta} = (\widehat{\beta}_0, \widehat{\beta}_1, \widehat{\beta}_2, \widehat{\beta}_{12}, \widehat{\beta}_{11}, \widehat{\beta}_{22})$ lie in dual vector spaces, we easily see that when passing from one model to the other these vectors certainly change but their inner product $\mathrm{x}^T \widehat{\beta}$, which define the **response surfaces**, remains the same (which is nice for **prediction**).

In other words, we trade direct interpretability of individual coefficients for numerical stability and more reliable statistical inference.

We may check this latter point (regarding prediction) for our models by comparing the mean responses…

```r
# Get the predicted values from both models
predictions_raw <- predict(model_full_raw, newdata = data_adv)
predictions_orth <- predict(model_poly_orth, newdata = data_adv)

# Check if the prediction vectors are numerically equal
all.equal(predictions_raw, predictions_orth)
```

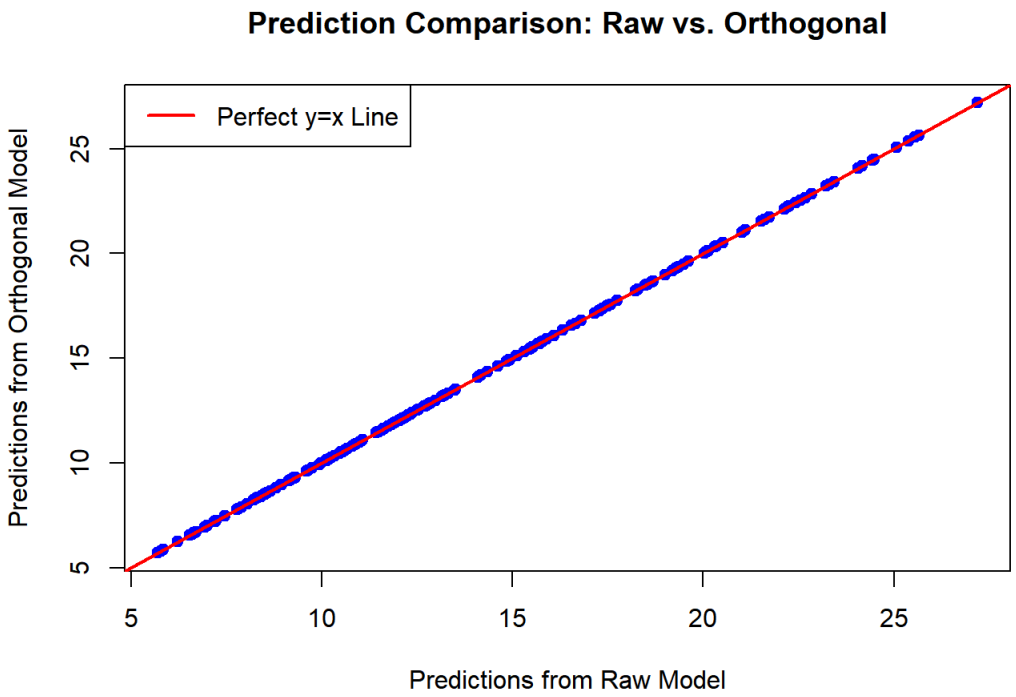
```
## [1] TRUE
```

```
#> [1] TRUE
```

…which can be graphically confirmed:

```r
pred_df <- data.frame(
  Raw_Predictions = predictions_raw,
  Orthogonal_Predictions = predictions_orth
)

# Plot the predictions
plot(
  pred_df$Raw_Predictions,
  pred_df$Orthogonal_Predictions,
  main = "Prediction Comparison: Raw vs. Orthogonal",
  xlab = "Predictions from Raw Model",
  ylab = "Predictions from Orthogonal Model",
  pch = 19,
  col = "blue"
)
```

```
# Add a perfect "y=x" line: if points are on this line, they are identical.
abline(a = 0, b = 1, col = "red", lwd = 2)

legend("topleft", legend = "Perfect y=x Line", col = "red", lty = 1, lwd = 2)
```



Now, the last summary…

```
print_coeffs_with_ci(model_poly_orth)
```

```
Coefficients:
               Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)   10.271740    0.129191   79.508     <2e-16    10.0169   10.5265 ***
poly(TV, 2)1  23.434934    1.208255   19.396     <2e-16    21.0519   25.8179 ***
poly(TV, 2)2  -9.990221    0.627776  -15.914     <2e-16   -11.2284   -8.7521 ***
poly(radio, 2)1   7.425769    1.239469    5.991      1e-08     4.9812    9.8703 ***
poly(radio, 2)2   0.493901    0.626077    0.789       0.43    -0.7409    1.7287
TV:radio       0.001075    0.000035   30.892     <2e-16     0.0010    0.0011 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…indicates that *radio^2* (equivalently, *poly(radio,2)2*) remains insignificant whereas the significance of *radio* (equivalently, *poly(radio,2)1*) gets restored, so we would like to enter into **backward selection** mode and remove *radio^2* from *model_poly_orth*. In order to validate this with ANOVA, we must fit the reduced, simpler model so as to pass to it after not being able to reject the null hypothesis. Instead of working directly with *model_poly_orth*, we prefer to manually generate all the polynomial terms, add them to a data frame as new columns, and then build the new version of the full model using these explicit column names.

```
# Generate the orthogonal polynomial matrix for TV and radio
poly_tv <- poly(data_adv$TV, 2)
colnames(poly_tv) <- c("TV_poly1", "TV_poly2")
poly_radio <- poly(data_adv$radio, 2)
colnames(poly_radio) <- c("radio_poly1", "radio_poly2")
# Create a new data frame with these columns
data_adv_poly <- cbind(data_adv, poly_tv, poly_radio)
```

For the sake of completeness, let us re-fit the full model to verify that our manual process correctly reproduces the original model *model_poly_orth*. Indeed…

```
model_poly_manual <- lm(sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1 + radio_poly2,
                        data = data_adv_poly)
summary_with_colors(model_poly_manual)
```

```
Call:
lm(formula = sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1 +
    radio_poly2, data = data_adv_poly)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.0027 -0.2859 -0.0062  0.3829  1.2100
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.027e+01  1.292e-01  79.508  < 2e-16 ***
TV_poly1     2.343e+01  1.208e+00  19.396  < 2e-16 ***
TV_poly2    -9.990e+00  6.278e-01 -15.914  < 2e-16 ***
radio_poly1  7.426e+00  1.239e+00   5.991 9.96e-09 ***
radio_poly2  4.939e-01  6.261e-01   0.789    0.431
TV:radio     1.075e-03  3.479e-05  30.892  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6244 on 194 degrees of freedom
Multiple R-squared:  0.986, Adjusted R-squared:  0.9857
F-statistic:  2740 on 5 and 194 DF,  p-value: < 2.2e-16
```

…thus yielding a summary which completely agrees with the one for *model_poly_orth*, as desired.

We may now fit the reduced model by simply omitting *radio_poly2* from *model_poly_manual*…

```
model_sub <- lm(sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1,
                data = data_adv_poly)
summary_with_colors(model_sub)
```

```
Call:
lm(formula = sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1,
    data = data_adv_poly)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-4.9949 -0.2969 -0.0066  0.3798  1.1686
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.026e+01  1.288e-01  79.716  < 2e-16 ***
TV_poly1     2.337e+01  1.204e+00  19.405  < 2e-16 ***
TV_poly2    -9.984e+00  6.271e-01 -15.920  < 2e-16 ***
radio_poly1  7.364e+00  1.236e+00   5.959 1.17e-08 ***
TV:radio     1.077e-03  3.466e-05  31.061  < 2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6238 on 195 degrees of freedom
Multiple R-squared:  0.986, Adjusted R-squared:  0.9857
F-statistic:  3432 on 4 and 195 DF,  p-value: < 2.2e-16
```

Let us display the *p*-values of the nested models in a single table…

```
model_list <- list(full_model = model_poly_manual, reduced_model = model_sub)
extract_pvals_simple <- function(model) {
  coef_table <- summary(model)$coefficients
 df <- data.frame(
  Regressor = rownames(coef_table),
    P_Value = coef_table[, "Pr(>|t|)"]
  )
rownames(df) <- NULL
return(df)
}
final_table <- map_dfr(model_list, extract_pvals_simple, .id = "Model")
final_table %>%
  # Conditionally format the P-Value column
  mutate(P_Value_Formatted = ifelse(P_Value < 0.001,
                                    sprintf("%.3e", P_Value),
                                    sprintf("%.4f", P_Value))) %>%
  select(Model, Regressor, P_Value_Formatted) %>%
  kable(format = "html",
        caption = "Model Comparison with P-Values",
        col.names = c("Model", "Regressor", "P-Value")) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  row_spec(which(final_table$Model == "full_model"),
           background = "#D9E3F4") %>%
  row_spec(which(final_table$Model == "reduced_model"),
           background = "#F4EAD9")
```

Model Comparison with P-Values

Model	Regressor	P-Value
full_model	(Intercept)	5.343e-150
full_model	TV_poly1	2.687e-47
full_model	TV_poly2	4.934e-37
full_model	radio_poly1	9.959e-09
full_model	radio_poly2	0.4311
full_model	TV:radio	7.710e-77
reduced_model	(Intercept)	9.121e-151
reduced_model	TV_poly1	2.050e-47
reduced_model	TV_poly2	4.120e-37
reduced_model	radio_poly1	1.167e-08
reduced_model	TV:radio	1.983e-77

…so we see that, as far as the standard stopping rule for **backward selection** goes, we may stop here (all *p*-values of the reduced model are much smaller that *0.05*). Let us additionally run the corresponding ANOVA:

```
anova(model_sub, model_poly_manual)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1
## Model 2: sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1 + radio_poly2
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    195 75.871
## 2    194 75.628  1   0.24261 0.6223 0.4311
```

Since the *p*-value is high (*~0.43*), the null hypothesis should **not** be rejected so there is no significant difference in the explanatory power of the two models. Therefore, we must switch to the simpler one *model_sub*, thus confirming that *poly(radio, 2)* was not a relevant predictor indeed.

And how this final quadratic model *model_sub* behaves in terms of statistical significance as compared to the previous linear model *model_partial*?

In this regard, we may run the corresponding ANOVA…

```
anova(model_partial,model_sub)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ TV + radio
## Model 2: sales ~ TV:radio + TV_poly1 + TV_poly2 + radio_poly1
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    197 556.91
## 2    195  75.87  2    481.04 618.18 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

…so the rather small *p*-value confirms that *model_partial* should be rejected indeed (in the presence of quadratic terms).

As for prediction, let us re-fit the quadratic model *model_sub* in a more straightforward manner:

```
model_sub_shortcut <- lm(sales ~ TV:radio + poly(TV, 2) + poly(radio, 1),
                         data = data_adv)
print_coeffs_with_ci(model_sub_shortcut)
```

```
Coefficients:
               Estimate  Std. Error  t value   Pr(>|t|)      2.5 %   97.5 %
(Intercept)   10.264807    0.128767   79.716     <2e-16    10.0109   10.5188 ***
poly(TV, 2)1  23.371803    1.204434   19.405     <2e-16    20.9964   25.7472 ***
poly(TV, 2)2  -9.983713    0.627114  -15.920     <2e-16   -11.2205   -8.7469 ***
poly(radio, 1)   7.364400    1.235827    5.959    1.2e-08     4.9271    9.8017 ***
TV:radio       0.001077    0.000035   31.061     <2e-16     0.0010    0.0011 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We now may use the function *int_conf_partial* above by just renaming it…

```
# The new function is essentially identical, just renamed for clarity.
int_conf_short <- function(new_TV, new_radio, model, conf_level = 0.95) {
  new_data_point <- data.frame(TV = new_TV, radio = new_radio)
  confidence_interval <- predict(
    model,
    newdata = new_data_point,
    interval = "confidence",
    level = conf_level
  )
  return(confidence_interval)
}
```

…and then compute the confidence interval for the prediction of the **mean response** evaluated at some chosen value for *(TV,radio)*:

```
int_conf_short(new_TV = 150, new_radio = 25 , model = model_sub_shortcut)
```

```
##        fit      lwr      upr
## 1 15.22219 15.08881 15.35557
```

Comparing this with the same computation for the strictly linear model *model_partial*

```
int_conf_partial(new_TV = 150, new_radio = 25 , model = model_partial)
```

```
##        fit      lwr      upr
## 1 14.48418 14.24802 14.72034
```

…we see that the introduction of quadratic terms has increased the prediction of *sales* in about *6%*!

Much more ahead, however. Indeed, if we **extrapolate** the prediction range by requiring that *TV+radio=350* and evaluate at some choices of *(TV,radio)* we find that…

```
results_table_quad <- get_confidence_table(new_data = my_new_points, model = model_sub_shortcut)
print(results_table_quad)
```

```
##    TV radio       fit       lwr      upr
## 1 350     0  9.515448  8.861573 10.16932
## 2 340    10 13.775571 13.250704 14.30044
## 3 290    60 31.517130 31.047812 31.98645
## 4 280    70 34.353631 33.821698 34.88556
## 5  10   340 21.250950 17.754903 24.74700
## 6   0   350 17.443880 13.648214 21.23955
```

…so that, differently from what happened with the same computation for the strictly linear model *model_partial*…

```
print(results_table_s)
```

```
##    TV radio      fit      lwr      upr
## 1 350     0 18.93529 18.21195 19.65862
## 2 340    10 20.35768 19.73213 20.98323
## 3 290    60 27.46965 26.74656 28.19274
## 4 280    70 28.89204 28.05099 29.73310
## 5  10   340 67.29669 62.23472 72.35865
## 6   0   350 68.71908 63.49562 73.94254
```

…the **mean response** is **not** monotone anymore but instead displays the typical quadratic behavior, thus reaching a **maximum value** somewhere. We may illustrate this by imposing *TV+radio=175*, which lies in a **interpolation range** covered by the data…

```
new_spend <- 175
TV <- c(25,60,65,70,75,80,85,90,95,100,105,110,115,120,150)
radio <- new_spend - TV
my_new_points_short <- data.frame(TV,radio)
results_table_short <- get_confidence_table(new_data = my_new_points_short, model=model_sub_shortcut)
print(results_table_short)
```

```
##     TV radio      fit      lwr      upr
## 1   25   150 15.65311 14.35552 16.95070
## 2   60   115 19.26946 18.49003 20.04889
## 3   65   110 19.54881 18.83028 20.26734
## 4   70   105 19.76884 19.10805 20.42964
## 5   75   100 19.92956 19.32336 20.53576
## 6   80    95 20.03096 19.47628 20.58564
## 7   85    90 20.07304 19.56686 20.57922
## 8   90    85 20.05580 19.59518 20.51642
## 9   95    80 19.97925 19.56132 20.39718
```

```
## 10 100    75 19.84338 19.46535 20.22140
## 11 105    70 19.64819 19.30736 19.98902
## 12 110    65 19.39368 19.08742 19.69994
## 13 115    60 19.07986 18.80560 19.35411
## 14 120    55 18.70671 18.46193 18.95150
## 15 150    25 15.22219 15.08881 15.35557
```

…to check that, apparently, the maximum for *sales* (on average) is about *20.0* and takes place somewhere around *(TV,radio)=(85,90)*. This output clearly confirms the well-known **trade-off** between interpretability and prediction when climbing up in hierarchy of regression models: by passing to the quadratic model, we have been able to substantially improve the prediction at the expense of turning it much less interpretable (Lou, Caruana, and Gehrke 2012).

We may complement this rather satisfactory analysis in many ways:

- we may estimate the **response** itself for prescribed values of *TV+radio* (by taking *TV+radio=175* we expect to find confidence intervals much wider than those in the previous table);
- we may insert **higher order** terms in the regression model, starting with those of third order, and check whether or not the resulting model, after being submitted to **backward selection**, is more significant than the quadratic one previously considered;
- we may then repeat the whole predictive analysis for such higher order models to see what happens;
- finally, delving deeper into the suggestions in (James et al. 2013, vol. 112, chap. 7), we may dispense altogether with monomials (or their orthogonal transforms) as regressors by adopting more flexible basis functions (piecewise polynomials, splines, etc.)…

…but we will stop here.

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

Lima, Levi de. 2023. “Probability, Estimation and Stochastic Calculus.” https://drive.google.com/file/d/1kgsn95hrqW-zVhsVfMvmMP0aabRCtCeP/view.

Lou, Yin, Rich Caruana, and Johannes Gehrke. 2012. “Intelligible Models for Classification and Regression.” In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 150–58.

---

1. We will challenge this linearity assumption later on when we discuss the statistical significance of introducing quadratic terms in the model.↩
2. Recall that the residual variance is an *unbiased* estimator for the unknown error variance.↩